INSTITUTO TECNOLOGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY
CAMPUS MONTERREY
PROGRAMA DE GRADUADOS EN ELECTRONICA,
COMPUTACION, INFORMACION Y COMUNICACIONES

ESTRATEGIA PARA INTEGRAR BASES DE DATOS BASADA EN EL REGISTRO Y EXTRACCION DE CORRESPONDENCIAS SEMANTICAS

T E S I S

PRESENTADA COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO ACADEMICO DE

MAESTRO EN CIENCIAS EN
TECNOLOGIA INFORMATICA

P O R :

MANUEL ENRIQUE GARCIA FIERRO

MONTERREY, N. L. NOVIEMBRE DE 2004

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

ESTRATEGIA PARA INTEGRAR BASES DE DATOS BASADA EN EL REGISTRO Y EXTRACCIÓN DE CORRESPONDENCIAS SEMÁNTICAS

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER
EL GRADO ACADEMICO DE:

MAESTRO EN CIENCIAS EN TECNOLOGÍA INFORMÁTICA

POR:

MANUEL ENRIQUE GARCIA FIERRO

MONTERREY, N.L. NOVIEMBRE DE 2004.

ESTRATEGIA PARA INTEGRAR BASES DE DATOS BASADA EN EL REGISTRO Y EXTRACCIÓN DE CORRESPONDENCIAS SEMÁNTICAS

POR:

MANUEL ENRIQUE GARCIA FIERRO

TESIS

Presentada al Programa de Graduados en Electrónica, Computación, Información y Comunicaciones.

Este trabajo es requisito parcial para obtener el grado de
Maestro en Ciencias en Tecnología Informática

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

NOVIEMBRE 2004

Dedicatoria

A Dios...

Por darme motivación, fuerza y voluntad para luchar por mis metas.

A mis padres...

Por enseñarme que los sueños se pueden alcanzar cuando se lucha por ellos. Este éxito es para ustedes.

A mis hermanos...

Sus palabras de aliento y cariño han sido una gran motivación para terminar la maestría.

Agradecimientos

A mi asesora:

Sus conocimientos, puntos de vista y paciencia han apoyado en la terminación de este trabajo de tesis y también han hecho una gran aportación a mi desarrollo personal. Muchas gracias.

A mis sinodales:

Por el apoyo e interés mostrado en el desarrollo de este trabajo de tesis.

A mis profesores:

Los conocimientos que ustedes me ayudaron a obtener fueron de gran utilidad en el desarrollo de esta tesis.

A mis compañeros de la maestría:

Janet, Idolina, Raúl, Carlos, Alex, Ricci. Muchas gracias por los días y noches de estudio y trabajos en equipo. Sin ustedes, la maestría no habría sido tan amena. Francisco muchas gracias por tu interés y apoyo en el desarrollo de esta tesis.

Resumen

Actualmente en las empresas existen bases de datos aisladas que requieren ser interoperables, lo cual significa que dichas bases de datos deben ser compartidas, accesadas y manipuladas de manera uniforme. Una de las opciones para lograr la interoperabilidad de un conjunto de bases de datos locales es integrar mediante un Esquema Global (GS), pero los problemas de heterogeneidad son un obstáculo para lograr esta integración. La heterogeneidad se puede presentar al tener diferentes DBMS´s, o por las diferencias en semántica de las bases de datos locales.

La heterogeneidad por diferentes DBMS´s se puede resolver con una capa de middleware. Sin embargo, el resultado de esta solución es un GS formado por un conjunto de tablas que pueden ser redundantes, lo cual impide la interoperabilidad entre las bases de datos locales.

La heterogeneidad semántica no es posible resolverla usando alguna herramienta, esto es debido a las limitaciones que tienen las herramientas para integrar bases de datos de hoy en día. Dichas limitaciones son la falta de una solución para integrar tablas, atributos y datos. Ante estas limitaciones, en esta tesis se presenta una estrategia para integrar bases de datos basada en el registro de correspondencias semánticas, la cual se divide en tres etapas: la primera etapa consiste en crear un GS a partir de los esquemas de las bases de datos locales, la segunda etapa consiste en registrar las correspondencias semánticas entre el GS y las bases de datos locales en un Diccionario de Datos Global (GDD) y la tercera etapa consiste en proporcionar acceso a las correspondencias semánticas mediante una arquitectura de componentes de software. De esta manera se logra la interoperabilidad de las bases de datos locales, es decir, para los usuarios del GS es transparente el acceso a las bases de datos locales, como si se tratara de una única base de datos.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1. Introducción

Este capítulo presenta la introducción a este trabajo de tesis. En la sección 1 se presentan los antecedentes, en donde se habla de los diferentes enfoques que ha tenido la integración de las bases de datos, en la sección 2 se presentan los objetivos en donde se describe de manera general la estrategia para integrar bases de datos y el enfoque de este trabajo de tesis, la sección 3 presenta las principales aportaciones así como los limites de este trabajo de investigación, y en la sección 4 se presenta la organización del documento de tesis.

1.1 Antecedentes

Hoy en día existen en las empresas bases de datos aisladas que requieren ser *interoperables*, lo cual significa que dichas bases de datos deben tener capacidades como compartir, interpretar y manipular información de manera uniforme [Lakshmanan et. al., 2001]. Para lograr la interoperabilidad de las bases de datos, se han dado algunas soluciones, algunas de las cuales son las siguientes:

- Con esquema global lógico (**GS**). Este esquema es una vista virtual de la unión de un conjunto de bases de datos [Batini et. al. 1986]. Parent y Spaccapietra [Parent y Spaccapietra 1998] señalan que el esquema global soporta el acceso a las bases de datos locales, esto significa que las actualizaciones que se hagan en dicho esquema se deben reflejar en las bases de datos locales.
- Integrar las bases de datos con un *esquema global físico*. Este esquema además de consolidar al conjunto de bases de datos locales, almacena los datos, lo cual es una ventaja, ya que es posible leer los datos de una manera más rápida, pero se tiene la desventaja de que los datos tienen que ser movidos hasta dicho esquema, y además se requiere un mecanismo de actualización, de tal manera que los cambios hechos en dicho esquema se reflejen en las fuentes de datos y viceversa.
- Sin *esquema global*. Algunos autores [Litwin 1984, 1990, Litwin et. al. 1982, Litwin y Abdellatif, 1986] sostienen que el esquema global es difícil de mantener debido a la heterogeneidad de las bases de datos locales. La integración sin un esquema global consiste en crear un esquema conceptual mediante un lenguaje manejador de bases de datos múltiples. Sin embargo, el *estado del arte* muestra que no existen herramientas disponibles que permitan realizar esta actividad.
- Integrar las bases de datos por medio de un *data warehouse* [Johnson 1999, Anahory y Murray, 1997]. El data warehouse apoya en la toma de decisiones lo cual se logra mediante el almacenamiento y extracción de la información. Sin embargo, el data warehouse no permite el acceso a las bases de datos de las que se extrae la información, lo cual es un obstáculo para hacer interoperables las bases de datos.

El esquema global es el medio ideal para lograr la integración de un conjunto de bases de datos, sin embargo, las herramientas existentes son limitadas, es decir, sólo logran una vista global de las bases de datos locales, sin hacer integración de tablas, atributos y datos de las bases de datos.

La definición de un esquema global requiere la solución de la heterogeneidad de las bases de datos locales. La heterogeneidad se puede presentar en diferentes grados o niveles:

- A *nivel DBMS.* Se presenta cuando las bases de datos locales son manejadas por diferentes DBMS´s. Las diferencias en DBMS se pueden presentar a nivel de sistema o por el enfoque de las bases de datos que manejan. Las bases de datos pueden ser relacionales, objeto-relacional, orientadas a objetos, red, jerárquicas o deductivas. Las herramientas middleware, como ODBC/JDBC, CORBA, RMI y herramientas para crear sistemas de bases de datos federados, como IBM DB2 Information Integrator o Sybase ECDA muestran que es posible solucionar la heterogeneidad a nivel DBMS´s. Esta solución es posible aun cuando los DBMS´s tienen diferente enfoques como relacional u objeto-relacional. Sin embargo, el resultado de esta solución es un esquema global con un conjunto de tablas que pueden ser redundantes, es decir, tablas que almacenan la misma información o que son semánticamente equivalentes, por ejemplo: dos tablas que almacenen el catálogo de autores.
- A *nivel de semánticas.* Aun cuando las bases de datos locales son manejadas por el mismo DBMS, pueden tener diferencias en semánticas. Estas diferencias se dan cuando las bases de datos locales presentan diferencias en el almacenamiento e interpretación de los datos. Para la solución de la heterogeneidad semántica no existen herramientas que permitan realizar esta actividad. Sin embargo, es posible capturar esta heterogeneidad en un diccionario de datos global (GDD) a través de correspondencias semánticas y de esta manera solucionar la heterogeneidad semántica.

1.2 Objetivos

Los objetivos de este trabajo de tesis son:

1. Presentar las necesidades, soluciones, metodologías y herramientas comerciales y no comerciales para integrar bases de datos heterogéneas y autónomas.
2. Proponer una estrategia para integrar bases de datos basada en el registro y extracción de correspondencias semánticas. La estrategia consiste en integrar los esquemas locales por medio de un GS. Después registrar las correspondencias semánticas entre el GS y los esquemas de las bases de datos locales en un GDD. Y por medio de una arquitectura de componentes de software acceder a esta información semántica con el fin de transformar operaciones SQL globales en sub operaciones SQL, ejecutarlos en las bases de datos locales y conciliar el conjunto de resultados locales en un resultado global.

El enfoque que se sigue a lo largo de esta tesis, es que un sistema para integrar bases de datos debe permitir la lectura o escritura a las bases de datos locales a través de un esquema global (virtual). Sin embargo, no debe permitir la alteración de las estructuras de las bases de datos locales para respetar su autonomía, es decir, las operaciones de los usuarios globales no deben afectar las operaciones de los usuarios locales.

En general, la integración de las bases de datos requiere la solución de conflictos, de acuerdo a la clasificación de Kim y Seo [Kim y Seo 1991], en esta tesis se resuelven los siguientes conflictos:

Conflictos de esquema

- Nombres diferentes para tablas semánticamente equivalentes
- Nombres diferentes para atributos semánticamente equivalentes
- Nombres iguales para atributos semánticamente no equivalentes
- Nombres iguales para tablas semánticamente no equivalentes
- Atributos faltantes
- Atributos faltantes pero implícitos
- Tipos de datos diferentes
- Tablas Faltantes

Conflictos de datos

- Diferentes unidades

Estos conflictos son los que suceden con más frecuencia al integrar bases de datos. Sin embargo, la clasificación de conflictos de Kim y Seo también menciona otros conflictos, que en esta tesis no han sido resueltos, dichos conflictos son los siguientes:

Conflictos de Esquema:

- Restricciones de integridad
- Conflictos de tabla versus atributo

Conflictos de datos:

- Diferentes expresiones
- Datos obsoletos o incorrectos
- Diferentes precisiones

Los conflictos de restricciones de integridad no son soportados por la arquitectura, los conflictos de tabla versus atributo no se abordaron en el caso de estudio, pero es posible capturar en el GDD correspondencias semánticas que les den solución.

Los conflictos de diferentes expresiones y diferentes precisiones no fueron implementados en la arquitectura del prototipo, sin embargo, se menciona la manera en la que es posible adaptar la arquitectura para que se puedan solucionar.

Los datos obsoletos o incorrectos no fueron solucionados ya que falta un mecanismo para detectarlos. En general, para la solución de estos conflictos no existen reglas precisas a seguir.

Debido a que la implementación del prototipo y caso de estudio de esta tesis abordan bases de datos relacionales y objeto-relacional, la estrategia propuesta soporta estas bases de datos. Sin embargo existen otros tipos de bases de datos que no son soportadas como orientadas a objeto, red o jerárquicas que pudieran requerir integración.

1.3 Aportaciones

Las aportaciones de esta tesis son:

1. Una estrategia para integrar bases de datos basada en el registro de correspondencias semánticas;
2. Un diccionario de datos global (GDD) para describir las correspondencias semánticas a nivel de tabla, atributo y datos;
3. Un método intuitivo para crear un esquema global;
4. Un componente de software para la extracción de la información de las correspondencias semánticas;
5. Proponer el uso de reglas de transformación para la solución de conflictos de datos, y un componente de software para codificar dichas reglas;
6. La solución de conflictos mediante el uso de correspondencias semánticas y reglas de transformación.

1.4 Organización del documento de tesis

La organización de este documento de tesis es la siguiente: En el capítulo 2 "Trabajo Relacionado", se presentan los diferentes enfoques que puede tener la integración de bases de datos, así como diferentes extensiones de SQL que se han realizado con el fin de integrar bases de datos, también se presenta el tema de heterogeneidad y algunos trabajos que han aportado diversas ideas para la solución de la heterogeneidad semántica. En el capítulo 3 "Herramientas de integración", se presentan las herramientas para integrar bases de datos, las cuales se clasifican en tres tipos: middleware, data warehouse y FDBMS´s. Por cada tipo de herramienta presentada se habla de sus ventajas y limitaciones. En el capítulo 4 "Integración semántica de datos" se presenta un método para integrar los esquemas de las bases de datos locales en un esquema global basado en correspondencias semánticas, dicho método se aplica a un caso de estudio que también es descrito en el capítulo 4. En el capítulo 5 "Arquitectura del prototipo", se presenta la arquitectura del prototipo para integrar bases de datos propuesto en esta tesis, se describen cada uno de los componentes, se presenta la forma de registrar las correspondencias semánticas en el GDD, así como la secuencia de operaciones que sigue la arquitectura. En el capítulo 6 "Implementación del prototipo" se presenta la implementación de la arquitectura propuesta en el capítulo 5 la cual se aplica al caso de estudio. Y por último, en el capítulo 7 "Conclusiones", se presentan las conclusiones generales de este trabajo de tesis.

Capítulo 2. Trabajo Relacionado

Actualmente varios autores han abordado el tema de integración de bases de datos, los cuales han presentado diversas soluciones siguiendo enfoques diferentes, lo cual da como resultado una serie de trabajos que utilizan diferente terminología, lo cual hace más difícil la compresión de este tema. Este capítulo presenta los trabajos relacionados, los cuales son clasificados de acuerdo a los enfoques que han dado para integrar bases de datos. Asimismo, se presenta el tema de heterogeneidad de las bases de datos, y también se presentan las arquitecturas que se pueden utilizar para integrar bases de datos.

La organización de este capítulo es la siguiente: en la sección 1 se presentan los diferentes enfoques para integrar bases de datos, en la sección 2 se presenta el tema de heterogeneidad de las bases de datos, en la sección 3 se presentan las arquitecturas para integrar bases de datos, por último, en la sección 4 se presenta la conclusión de este capítulo.

2.1 Diferentes enfoques de la integración de bases de datos

Cuando se habla de *integración de bases de datos* existen diferentes enfoques:

- Esquema global (virtual)
- Esquema global físico
- Sin esquema global
- Integración de aplicaciones empresariales (EAI)
- Integración con XML
- Data warehouse

La tabla 2.1 presenta una descripción de estos enfoques, sus ventajas y desventajas así como las publicaciones que los soportan.

2.1.1 Esquema Global (virtual)

Un enfoque es la integración de bases de datos o *esquemas externos* mediante un GS, donde los esquemas externos son partes de las base de datos a integrar y el esquema global es una vista virtual de la unión de un conjunto de bases de datos [Batini et. al. 1986]. [Parent y Spaccapietra 1998] señalan que el GS soporta el acceso a las bases de datos locales, esto significa que las actualizaciones que se hagan en el esquema global, se deben reflejar en las bases de datos locales. Este esquema proporciona interoperabilidad entre las bases de datos ya que usa una misma semántica para el conjunto de bases de datos y permite ser visto por los usuarios como una única base de datos a la cual se le podrán

realizar consultas y cambios. El GS es virtual porque debe ser creado de manera lógica. Sin embargo, también es posible que el GS sea físico, lo cual significa que almacena los datos de las bases de datos locales, por esta razón este esquema proporciona las mismas ventajas que el GS, pero requiere un mecanismo de actualización porque los datos tienen que ser movidos desde las fuentes de datos hasta dicho esquema.

Enfoque	¿Cómo se logra la integración?	Ventajas	Desventajas	Publicaciones
Esquema Global (virtual)	Mediante un Esquema virtual que representa a los esquemas del conjunto de bases de datos locales.	Una vista integrada de las bases de datos locales. Soluciona la heterogeneidad semántica.	El mantenimiento del Esquema Global.	Batini et. al., 1986; Parent y Spaccapietra ,1998
Esquema Global Físico	Mediante un Esquema Global que almacena los datos de las bases de datos locales.	La lectura de datos es rápida. Las ventajas de tener un esquema global.	Requiere un mecanismo de actualización. El mantenimiento del Esquema Global.	Hull y Zhou ,1996
Sin Esquema Global	Mediante sentencias de lenguaje manejador de bases de datos múltiples.	No requiere el mantenimiento de un esquema global. Facilita la escalabilidad.	No existen lenguajes manejadores de bases de datos múltiples comerciales. La creación de un lenguaje para manejar bases de datos múltiple es una actividad difícil.	Lee y Baik, 1999; Grant et. al, 1993; Litwin, 1990; Litwin et. al., 1989; Litwin, 1984 ; Litwin, et. al. 1982; Litwin y Abdellatif.1986.
EAI	Mediante middleware como los objetos distribuidos.	Proporciona transparencia de DBMS.	No soluciona la heterogeneidad semántica, por lo que se tiene que hacer trabajo adicional en dicha solución.	Linthicum, 2000.
Integración con XML	Mediante el formato de datos común que proporciona el lenguaje XML.	Proporciona transparencia de DBMS. Un formato común. La mayoría de los DBMS´s actuales importan o exportan documentos XML.	Se tiene que hacer un estudio para la solución de la heterogeneidad semántica.	Seligman y Rosenthal 2001.
Data warehouse	Mediante un repositorio de datos y herramientas ETL(Extraction, Transformation and Loading).	Facilita la toma de decisiones.	No permite el acceso a las fuentes de datos. No soluciona la heterogeneidad semántica.	Johnson, 1999. Anahory y Murray 1997.

Tabla 2.1: Principales enfoques de integración de bases de datos.

Un ejemplo de integración de bases de datos con el uso de un esquema global es la extensión de SQL SchemaSQL [Lakshmanan et. al., 2001]. Las extensiones SQL

proporcionan funciones adicionales para integrar bases de datos que el lenguaje SQL no proporciona. A diferencia de las extensiones MSQL y SemQL, presentadas en la sección 2.1.3, las cuales son enfocadas para sistemas sin esquema global o no federados, SchemaSQL es para sistemas de bases de datos federados. Los sistemas con SchemaSQL se caracterizan por tener un servidor SchemaSQL, el cual contiene en una tabla federada la información semántica de las bases de datos locales. Este servidor interactúa con una máquina de SQL y permite hacer operaciones a varias bases de datos. Las principales características de SchemaSQL son las siguientes:

- Resuelve los conflictos sintácticos entre diferentes bases de datos;
- Tiene un poder expresivo el cual es independiente del esquema específico con el cual una base de datos es estructurada;
- Ayuda a la reestructuración, entre las diferencias de datos o esquemas;
- Ayuda a desarrollar complejas agregaciones de bloques.

Por ejemplo: Se tienen dos bases de datos, univ-A y univ-B. De las cuales se desea listar los departamentos de univ-A que pagan mejor salario a sus técnicos comparados con los técnicos de la univ-B.

La consulta queda de la siguiente manera:

```
SELECT A.dept
FROM  univ-A::salInfo A, univ-B::salInfo B,
WHERE  A.category  =  "technician" and
                B.category  =  "technician" and
                A.salFloor >  B.AttB
```

Donde:

A) A.dept, son los departamentos de la universidad A.

B) univ-A::salInfo A, univ-B::salInfo B, es la asignación de las tablas de salarios a las variables A y B respectivamente.

C) A.category, B.category, es el tipo de trabajo, en este caso se trata de técnicos.

D) A.salFloor, B.AttB, son los salarios de los trabajadores técnicos.

El enfoque mediante un GS es el ideal de integración de bases de datos porque proporciona transparencia de DBMS y de semánticas, y además no requiere de un lenguaje manejador de bases de datos múltiples el cual es difícil de crear; sin embargo, no existen herramientas que logren lo que en teoría un esquema global debe proporcionar. Se habla más de las herramientas para integrar bases de datos en el siguiente capítulo.

2.1.2 Esquema Global (físico)

El *esquema global físico* es un esquema global, el cual además de representar al conjunto de esquemas de las bases datos, tiene la capacidad de almacenar sus datos [Hull and Zhou 1996]. Este esquema presenta las mismas ventajas que un esquema global (virtual), y además la ventaja de contener los datos de las bases de datos locales, lo cual significa que el acceso a los datos es más rápido, sin embargo este esquema tiene la

desventaja de requerir un mecanismo de actualización, ya que los cambios hechos en las bases de datos locales se deben reflejar en dicho esquema y viceversa.

2.1.3 Sin un esquema global

[Litwin 1984, 1990, Litwin et. al. 1982, 1986] sostienen que el esquema global es difícil de mantener debido a la heterogeneidad de las bases de datos locales. Litwin [Litwin 1990] afirma que la heterogeneidad semántica debe ser solucionada mediante un lenguaje para manejar bases de datos múltiples. Dicho lenguaje es el sustituto del esquema global. La integración mediante un lenguaje consiste en crear un esquema conceptual mediante las sentencias de dicho lenguaje [Litwin 1990].Dos ejemplos de integración sin un esquema global son las extensiones de SQL MSQL [Grant et. al, 1993, Litwin et. al. 1989], y SemQL[Lee y Baik 1999]. A continuación se presenta la descripción de estas dos extensiones de SQL.

MSQL

Las principales características de MSQL son las siguientes:

- Creación, modificación o alteración de tablas con comandos simples en cualquier número de bases de datos y la importación de bases de datos, tablas, o columnas esquemas.
- Recuperación o modificación incluyendo la unión de datos en diferentes esquemas de bases de datos.
- La transformación dinámica de los significados de los atributos, unidades de medida, etc. a tipos de valores definidos por el usuario que pueden ser recuperados o actualizados (algunas restricciones pueden ser aplicadas a las actualizaciones).
- Consultas entre varias bases de datos para flujo de datos entre las bases de datos.
- Agregación dinámica de datos de diferentes bases de datos usando varios estándares.
- Creación de vistas de y bases de datos virtuales sobre tales vistas.
- La creación de objetos auxiliares como triggers, procedimientos almacenados y transacciones (procedimientos).

Por ejemplo: Se desea crear una base de datos múltiple los bancos Bital, Banamex y City Bank.

1. Los comandos en MSQL para crear la base de datos múltiple quedarían de la siguiente forma:

```
CREATE MULTIDATABASE  Banks (Bital Banamex CityBank)
```

2. Después City Bank desea darse de baja en la base de datos múltiple mientras que Bancomer desea unirse:

```
ALTER Banks
INCLUDE Bancomer
REMOVE CityBank
```

3. Abonar (como premio) 500 pesos a los clientes de Bital quienes tengan más dinero que en su cuenta de City Bank:

```
USE (Bital b) CityBank
UPDATE account
SET account.balance = account.balance + 500
WHERE account.balance > acc.balance
AND b.client.clname = CityBank.client.cname AND b.client.street =
CityBank.client.street
```

Donde:

A) *USE (Bital b) CityBank,* define las bases de datos que participaran en la operación SQL.
B) Bital b, es el nombre de la base de datos del banco "Bital" y *b* es su alias.
C) CityBank, es el nombre de la base de datos del banco "CityBank".
D) account, es el nombre de la tabla que contiene las cuentas de los clientes de Bital.
E) acc, es el nombre de la tabla que contiene las cuentas del banco "CityBank".
F) account.balance, b.client.clname, b.client.street, son los atributos de la base de datos "Bital".
G) CityBank.client.cname, CityBank.client.street, son atributos de la base de datos "CityBank.
H) b.client.clname=CityBank.client.cname AND b.client.street =CityBank.client.street, es la condición sobre los atributos.

Se puede notar en esta operación que se utiliza un alias *b* para la base de datos *Bital,* esto es porque contiene tablas como *client,* cuya tabla equivalente en la base de datos *CityBank* tiene el mismo nombre, y por esta razón en la condición se utiliza el alias *b* en vez de el nombre de la base de datos. Se puede notar que existe una condición donde se verifica que el cliente viva en la misma calle, de esta manera se asegura que se trate del mismo cliente, porque pueden existir clientes con el mismo nombre.

SemQL

Los lenguajes para bases de datos múltiples no sólo deben ser enfocados a solucionar los conflictos entre las estructuras de los esquemas de las bases de datos componentes también deben ser enfocados a solucionar conflictos entre los datos, con este propósito SemQL [Lee y Baik 1999] soluciona la mayor parte de los problemas semánticos entre las bases de datos utilizando una red semántica, tal como se muestra en la figura 2.1, la red semántica contiene las relaciones entre las bases de datos a nivel entidad y atributo, de esta manera el procesador semántico de SemQL interactúa con la red semántica para reformular las consultas de los usuarios y las divide en varias sub consultas. Con la ayuda

de la red semántica SemQL elimina la necesidad de un esquema global. Una consulta en SemQL se divide en varias sub consultas, una por cada base de datos local. Los resultados de cada consulta se consolidan mediante un integrador.

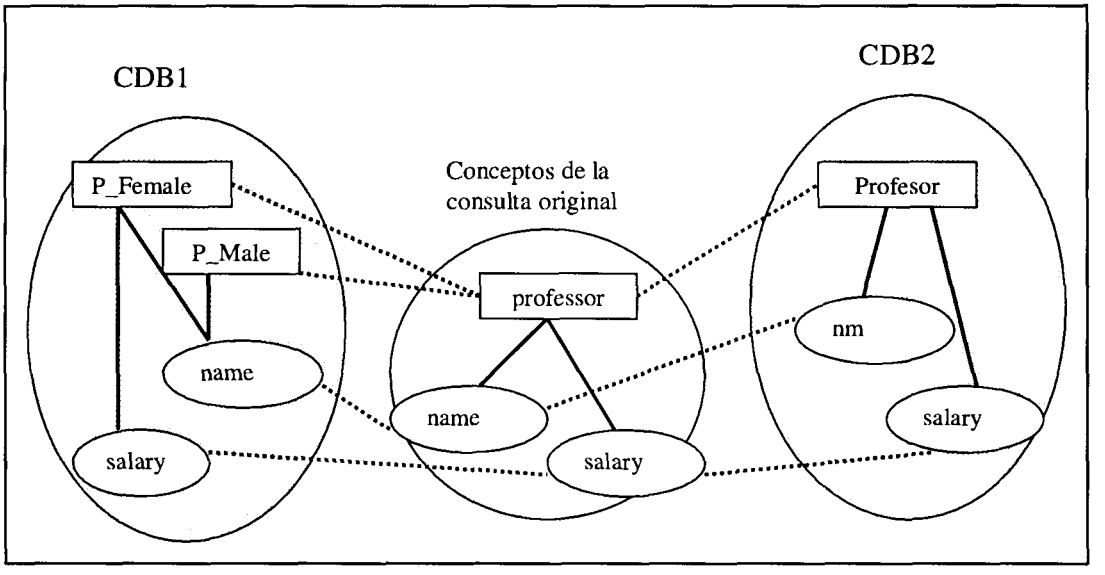

Figura 2.1: Ejemplo de una red semántica. [Lee y Baik 1999]

Por ejemplo: Dada la red semántica presentada en la figura 2.1 de los esquemas de dos bases de datos. Se desea hacer una consulta que regrese los profesores cuyo salario sea más de $ 50, 000.

1. Los comandos de la consulta original quedan de la siguiente manera:

```
SELECT profesor.name
FROM profesor
WHERE profesor.salary> 50, 000
```

2. Un procesador de consultas divide la consulta en dos: una para CDB1 y otra para CDB2.
   La consulta para CDB1:

```
SELECT name
FROM P_Male
WHERE salary > 50,000
UNION
SELECT name
FROM P_Female
WHERE salary > 50,000
```

   La consulta para CDB2:

```
SELECT nm
FROM Professor
WHERE salary > 50,000
```

2.1.4 Integración de aplicaciones empresariales (EAI)

Linthicum [Linthicum 2000] argumenta que las empresas deben integrar sus aplicaciones aisladas utilizando alguno de los cuatro niveles o estrategias de *Integración de Aplicaciones Empresariales* (**EAI**):

1. *Datos:* Son los procesos, técnicas y tecnologías para mover datos entre las bases de datos, es decir, extraer información desde una base de datos a otra, y quizá procesar esa información como se necesite. En una o más compañías esto puede involucrar cientos de tablas de datos, las cuales pueden ser heterogéneas y autónomas.
2. *Interfaz de aplicación:* Es enlazar las interfaces de aplicaciones para acceder a los procesos del negocio o información. De esta manera, es posible unir muchas aplicaciones. El reto de integrar las aplicaciones mediante este nivel son las características y funciones especificas de las interfaces de aplicaciones.
3. *Método:* Hay dos estrategias básicas para integrar mediante este nivel, en la primera se puede crear una serie de servidores de aplicación, los cuales son mejor conocidos como *objetos distribuidos*, de los cuales se habla más en la sección 3.2 que existen en un servidor físico compartido. En la segunda se pueden compartir métodos existentes dentro de las aplicaciones usando tecnología para compartir métodos distribuidos.
4. *Interfaz de usuario:* Es un nivel más primitivo. Mediante este nivel de integración se permite unir aplicaciones usando las interfaces de usuarios como un punto común de integración.

Se puede notar que Linthicum ubica la integración de bases de datos en el nivel de *datos*. Linthicum argumenta que las herramientas middleware como CORBA o RMI facilita la integración mediante alguno de estos niveles.

El enfoque de EAI está orientado a lograr la comunicación y transmisión de información de una aplicación a otra. Este enfoque se apega muy bien cuando las bases de datos no presentan heterogeneidad semántica, pero cuando se presenta este tipo de heterogeneidad es necesario hacer un análisis de las semánticas utilizadas por las bases de datos locales.

2.1.5 Integración de bases de datos con XML

Desde 1998, han surgido nuevos avances para la integración de bases de datos utilizando el lenguaje XML (*Extensible Markup Language*), el cual es un sucesor de HTML, pero con la diferencia que este se enfoca en el significado que tiene la información en Internet, en vez de la presentación de los documentos. XML proporciona un formato común para expresar la información en Internet, la cual pueden ser documentos de todos

los tipos incluyendo bases de datos. [Seligman and Rosenthal 2001] presentan las ventajas que tiene XML en la integración de información y bases de datos sobre sistemas autónomos y heterogéneos, algunas de las ventajas de XML que han contribuido a tener dicha aceptación:

- *Distribución Geográfica:* Los datos pueden estar en una amplia distribución geográfica. Esto es posible, porque los documentos XML pueden ser enviados a través de Internet.
- *Estructuras de datos y lenguajes heterogéneos:* Las bases de datos pueden tener diferencias en las estructuras así como también en los lenguajes de manipulación para las que fueron creadas, por ejemplo: bases de datos para SQL, lenguajes propietarios o archivos de datos sin lenguaje de manipulación.
- *Representación y semántica heterogénea de los atributos*: Los integradores reconcilian las diferentes representaciones de los mismos datos, por ejemplo la altitud en una base de datos puede estar representada en millas mientras que en otra base de datos puede estar representada en kilómetros, además el tipo de dato puede ser entero (datatype=integer, unit=miles). Esto se puede hacer definiendo un esquema global que sea descrito con un documento XML. Dicho esquema puede representar al conjunto de esquemas de las bases de datos locales, de tal manera que se puedan hacer los mapeos entre el esquema global y los esquemas locales.
- *Esquemas Heterogéneos:* Son las diferentes formas de organizar las tablas de las base de datos, por ejemplo la nómina de un cliente puede estar representada en un esquema mientras que en otra base de datos puede estar en dos o más. Varias comunidades han expresado su interés en los esquemas XML para proveer un modelo neutral.
- *Identificación de objetos:* Los avances en la descripción de la representación de los atributos y las semánticas pueden quitar el problema de objetos no identificados, por ejemplo: la fecha de pago de un documento, ¿se encuentra en formato americano o europeo?
- *Reconciliación de valores de datos:* Como los datos en XML describen los tipos de datos, es fácil hacer conversiones para conciliarlos.

Con estas ventajas, XML se proyecta como un enfoque para integrar bases de datos. Sin embargo, XML es sólo un formato común que puede describir las semánticas de las bases de datos y que permite compartir información en texto, y para lograr la integración de bases de datos se tendría que combinar con componentes de software que permitan obtener las semánticas de las bases de datos locales para hacer los mapeos necesarios. En esta tesis no se utilizó XML porque las semánticas de las bases de datos locales se almacenan en un diccionario de datos global, el cual es una base de datos relacional, lo cual facilita la ejecución de operaciones SQL y de esta manera se permite extraer la información semántica de las bases de datos locales.

2.1.6 Integración de bases de datos utilizando un *data warehouse*

Un *data warehouse* [Johnson 1999, Anahory y Murray 1997] es un repositorio de datos global alimentado con datos de diferentes bases de datos y documentos de una empresa que permite la consolidación de los datos para la toma de decisiones. Para la consolidación de datos los data warehouses utilizan herramientas ETL (Extraction, Transformation and Loading), de tal manera que los usuarios puedan hacer consultas a estos datos para tener una visión integrada de los negocios y de esta manera faciliten la toma de decisiones. Una de las características del data warehouse es que los datos existen en el repositorio de datos para ser consultados, pero no modificados. Las herramientas ETL permiten que la información viaje en un solo sentido (hacia el repositorio de datos), por esta razón los data warehouses tienen la limitante de no permitir el acceso directo a las bases de datos locales que lo alimentan.

2.2 Heterogeneidad de las bases de datos

De acuerdo la clasificación de heterogeneidad de Sheth [Sheth y Larson 1990], la heterogeneidad de las bases de datos se puede presentar en los siguientes aspectos:

1. Heterogeneidad en los DBMS´s
2. Heterogeneidad semántica

A continuación se presentan estos dos aspectos de la heterogeneidad.

2.2.1 Heterogeneidad en los DBMS's

La *heterogeneidad en los DBMS's* son las diferencias que existen en los modelos de datos o a nivel de sistema [Sheth y Larson 1990].

- *La heterogeneidad en los modelos de datos* es el resultado de los diferentes enfoques que puede tener los DBMS´s para almacenar los datos, es decir los DBMS´s pueden ser Relaciónales, Objeto-Relacional, Orientados a Objetos, Red, Jerárquicos y Deductivos. Para dar solución a las diferencias en los modelos de datos Chung y Mah [Chung y Mah 1995] proponen integrar los modelos de datos mapeándolos en un *modelo unificado*, el cual es un modelo relacional con características orientadas a objetos, lo cual facilita el mapeo entre los modelos de datos.
- *La heterogeneidad de nivel de sistema* se puede presentar cuando los DBMS´s tienen diferentes primitivas para manejar el control de concurrencia, manejo de transacciones, etc.

Hoy en día es posible lograr una transparencia en el DBMS que se utiliza en las bases de datos locales integrándolos con una capa de Middleware para bases de datos. Sin embargo esta solución sólo permite la creación de un GS, sin hacer integración de esquemas y datos.

2.2.2 Heterogeneidad de Semántica

La *semántica* de las bases de datos es el significado que tienen los datos para sus usuarios. La *heterogeneidad semántica* se presenta cuando hay un desacuerdo en el significado, interpretación o el uso de los datos [Sheth y Larson 1990]. Para solucionar la heterogeneidad semántica se requiere una integración semántica. Tal como se muestra en la tabla 2.2, en la literatura varios autores han abordado el tema de integración semántica de datos [Bright et. al 1994, Johannesson 1994, Parent y Spaccapietra 1998, Bergamaschi 1999, Hakimpour y Geppert 2001, Rishe et. al 2000], cuya solución más empleada es la utilización de un GS. Sin embargo, algunos autores [Litwin 1984, Litwin et. al. 1982, 1986] sostienen que dicho esquema es difícil de mantener ya que los cambios hechos en las fuentes de datos deben ser reflejados en el esquema global.

Publicación	**Descripción de la solución**
Hakimpour y Geppert, 2001	Solución basada en *ontologías*, donde los esquemas son de diferentes comunidades y cada comunidad usa su propia ontología, las ontologías son unidas, finalmente la ontología unida sirve de base para formar el esquema global.
Parent y Spaccapietra, 1998	Parent y Spaccapietra proponen soluciones para los conflictos entre las bases de datos locales y además dividen el proceso de integración en tres pasos: pre-*integración*, donde los esquemas son transformados en para hacerlos lo más homogéneos posibles; *identificación de correspondencias*, el cual es la identificación y descripción de las relaciones entre los esquemas; e *integración*, el cual el paso final donde se resuelven los conflictos entre esquemas y se unifican en un esquema global.
Rishe, et al, 2000	Esta solución se basa en correspondencias semánticas para la integración de esquemas.
Johannesson, 1994	Un método heurístico para identificar las correspondencias entre los esquemas. El proceso de integración es dividido en tres pasos comparación de esquemas, conformación de esquemas y unión de esquemas.
Bright, 1994	Esta solución utiliza un SSM (*Sumary Schemas Model*) para ayudar en la identificación de las semánticas utilizando una estructura de datos global para abstraer la información disponible en los sistemas de bases de datos múltiples. Con esta forma ayuda a los usuarios para usar sus propios términos cuando accedan a los datos, en lugar de que usen los términos especificados por el sistema.

Tabla 2.2: Trabajos enfocados en la solución de le heterogeneidad semántica.

El principal reto de la solución de la heterogeneidad semántica es hacer que el conjunto de esquemas heterogéneos se vea como un esquema global el cual debe proporcionar transparencia de semánticas porque utiliza una misma semántica para el conjunto de bases de datos locales. Para la creación de este esquema global, se requiere la solución de conflictos entre los esquemas locales a integrar. Los conflictos entre las bases de datos son el resultado de las diferentes representaciones simbólicas en las bases de datos locales [Kim y Seo 1991]. De acuerdo a la clasificación de conflictos de Kim y Seo [Kim y Seo 1991] presentada en la figura 2.2., los conflictos se pueden clasificar en conflictos de esquema y de datos. Los conflictos de esquema son las diferencias entre tablas y atributos, y los conflictos de datos son las diferentes formas de almacenar el mismo valor.

I.- Conflictos de Esquema

- A. Conflictos de tabla versus tabla
  - 1. Conflictos de relaciones uno a uno
    - a. Conflictos de los nombres entre las tablas
      - 1) Diferentes nombres para tablas equivalentes
      - 2) Mismo nombre para diferentes tablas
    - b. Conflictos entre las estructuras de las tablas
      - 1) Atributos faltantes
      - 2) Atributos faltantes pero implícitos
    - c. Conflictos entre las restricciones de las tablas
  - 2. Conflictos de relaciones mucho a muchos
- B. Conflictos de atributo versus atributo
  - 1. Conflictos de atributo uno a uno
    - a. Conflictos de los nombres entre los atributos
      - 1) Diferentes nombres para atributos equivalentes
      - 2) Mismo nombre para diferentes atributos
    - b. Conflictos de los valores por default
    - c. Conflictos entre las restricciones de los atributos
      - 1) Conflictos de los tipos de datos
      - 2) Conflictos de las restricciones de integridad de los atributos
  - 2. Conflictos de atributos mucho a muchos
- C. Conflictos de Tabla versus Atributo

II. Conflictos de los datos

- A. Datos Incorrectos
  - 1. Datos de entrada incorrectos
  - 2. Datos Obsoletos
- B. Diferentes representaciones de los mismos datos (misma representación de los diferentes datos)
  - 1. Diferentes expresiones
  - 2. Diferentes unidades
  - 3. Diferentes precisiones

Figura 2.2: Clasificación de conflictos de Kim y Seo. [Kim y Seo 1991]

2.3 Arquitecturas para integración de bases de datos

Las arquitecturas para integración de bases de datos son las bases de datos múltiples. La figura 2.3 presenta la clasificación de los *sistemas manejadores de bases de datos múltiples* (**MDBMS**) de Sheth [Sheth y Larson 1990], en donde se puede notar que estos sistemas se caracterizan porque soportan la operación de varios sistemas de bases de datos (**DBS**) donde cada **DBS** es manejado por un DBMS. Éstos se clasifican en dos tipos: federado y no federado. Los MDBMS no federados son una integración de sistemas de bases de datos distribuidos no autónomos. Los *MDBMS federados* (**FDBMS**) se componen de sistemas de bases de datos que son autónomos y permiten compartir de manera controlada sólo una parte de sus datos. Un FDBMS federado puede ser *débilmente acoplado* o *fuertemente acoplado*. Los FDBMS fuertemente acoplados pueden ser de una federación o de múltiples federaciones.

Los trabajos de Litwin [Litwin 1984, Litwin et. al. 1982, 1986] presentan una arquitectura de bases de datos múltiples, (MDBMS no federado), esta arquitectura propone utilizar un *lenguaje manejador de bases de datos múltiples* en vez de un esquema global. Mediante esta arquitectura, la integración se realiza con dicho lenguaje mediante sentencias [Litwin 1990], por esta razón el lenguaje es el componente esencial de un sistema de bases de datos mútiples [Litwin 1988, Litwin 1990]. Por ejemplo: para integrar las bases de datos A, B y C la sentencia sería de esta manera: `CREATE MULTIDATABASE (A,B,C).`

Litwin argumenta que el esquema global no debe ser utilizado para integrar bases de datos porque es difícil de mantener, ya que los cambios realizados en las bases de datos locales se deben reflejar en dicho esquema y viceversa, además requiere manejar la autonomía y la redundancia entre los datos.

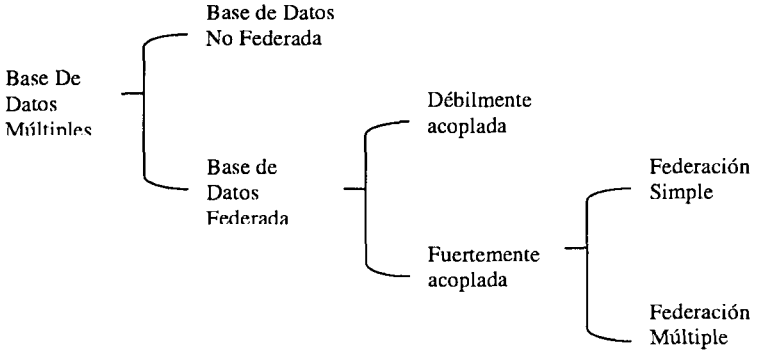

Figura 2.3: Clasificación de Sheth de Bases de Datos Múltiples. [Sheth y Larson 1990]

De las arquitecturas para integrar bases de datos, el FDBMS es una excelente opción ya que debe proporcionar transparencia de semánticas y DBMS, además no requiere de una extensión del lenguaje SQL para manejar las bases de datos locales. Sin embargo las herramientas para existentes para crear un FDBMS son limitadas, es decir no hacen integración semántica lo cual impide la interoperabilidad de las bases de datos locales. Se habla más de las herramientas para integrar bases de datos en el siguiente capítulo.

2.4 Conclusión

Las investigaciones presentadas en este capítulo presentan evidencia de que existen varios enfoques para la integración de bases de datos: esquema global(virtual), esquema global físico, sin esquema global, data warehouse, EAI e integración de bases de datos con XML. Un análisis de estos enfoques revela que la solución ideal consiste en la creación de un esquema global, ya que un esquema de este tipo no requiere de un lenguaje para manejar bases de datos múltiples y proporciona transparencia de DBMS y de semánticas. Se ha visto que la creación del esquema global requiere la solución del problema de la heterogeneidad entre las bases de datos locales, la cual se puede presentar por dos motivos: DBMS´s y semánticas de las bases de datos locales. La heterogeneidad de diferentes DBMS´s se puede solucionar con una capa de middleware, sin embargo para la solución del problema de la heterogeneidad semántica no existen herramientas, lo cual es un obstáculo para crear el esquema global. Para dar solución a este problema se han creado metodologías para integrar un conjunto de esquemas. Dichas metodologías tienen la desventaja de requerir un lenguaje extensión del lenguaje SQL, el cual debe tener capacidades para interoperar un conjunto de bases de datos. La creación de un lenguaje de este tipo es una actividad difícil.

Por lo tanto, en este capítulo se concluye que la integración de bases de datos es un problema al que no se le ha dado una solución que permita transparencia de DBMS, semánticas de las bases de datos y que además no requiera de un lenguaje extensión al lenguaje SQL. En este trabajo de tesis se integran las bases de datos locales mediante una arquitectura FDBMS, la heterogeneidad semántica se soluciona a partir de la creación de un esquema global y el registro de correspondencias semánticas con los esquemas de las bases de datos locales.

Capítulo 3. Herramientas de integración

Ya se han presentado los trabajos relacionados a la integración de bases de datos, en este capítulo se presentan las herramientas que se pueden utilizar para integrar bases de datos, estas herramientas se presentan en tres partes: middleware, data warehouse, y federación. Por cada herramienta que se presenta, se habla de sus ventajas, desventajas y limitaciones. Este capítulo también hace una comparación de las herramientas tomando como base los requerimientos que cada herramienta puede satisfacer. La organización de este capítulo es la siguiente: la sección 1 presenta la introducción a este capítulo, donde se habla de los requerimientos de integración y su relación con las herramientas para integrar bases de datos, en la sección 2 se presentan las herramientas de middleware, en la sección 3 se presentan las herramientas para crear data warehouses, en la sección 4 se presentan las herramientas para crear una federación de bases de datos, la sección 5 se presenta una comparación de las herramientas de acuerdo a los requerimientos que satisfacen y por ultimo, en la sección 6 se presenta la conclusión de este capítulo.

3.1 Introducción a las herramientas para integrar bases de datos

Tal como se vio en la sección 1.1, la interoperabilidad de las bases de datos significa que se debe tener la capacidad de *compartir*, *acceder* y *manipular* las fuentes de datos de manera *uniforme* [Lakshmanan et. al., 2001].

- *Compartir* significa que cada base de datos local debe tener la capacidad de publicar su información para que entidades externas las utilicen.
- *Acceder* significa que los usuarios globales deben tener la capacidad de ver la información de las bases de datos locales.
- *Manipular* es la capacidad de que los usuarios globales puedan cambiar o borrar los datos de las bases de datos locales.

Cuando se utiliza el término "uniforme", se entiende que las capacidades anteriores deben de utilizar una misma *semántica* para el conjunto de bases de datos locales, lo cual significa que se debe proporcionar una vista global del conjunto de bases de datos.

Con el fin de hacer un análisis más profundo de las herramientas, se analizará el término "uniforme" desde tres aspectos:

1. Si la herramienta tiene la capacidad de crear un esquema global (virtual), el cual puede contener o no tablas redundantes se utilizará el término: *Creación de un GS*
2. Si la herramienta tiene la capacidad de crear un esquema global (virtual), integrando también las tablas y atributos, se utilizará el término: *Integración de Esquemas.*
3. Si la herramienta no sólo integra los esquemas, sino también los valores de las instancias de las bases de datos locales, entonces se utilizará el término: *Integración de datos.*

Estas capacidades son los principales requerimientos de integración que una buena herramienta para integrar bases de datos debe satisfacer. Con base en estos requerimientos, en la sección 3.5 de este capítulo se presenta una comparación de las herramientas para integrar bases de datos.

La figura 3.1 presenta las principales herramientas para integrar bases de datos, clasificadas en middleware, data warehouse y federación. En esta figura se puede notar que los usuarios globales pueden ver los datos de las bases de datos fuentes a través de una herramienta FDBMS o data warehouse. Estas herramientas se comunican con las bases de datos fuente mediante una combinación de herramientas middleware y ODBC/JDBC, donde la capa de middleware proporciona comunicación y ODBC/JDBC permite el acceso transparente al DBMS de la base de datos local. Se puede notar en este esquema que las bases de datos locales alimentan al data warehouse, pero este no proporciona acceso a dichas bases de datos. A diferencia del data warehouse, el FDBMS si permite el acceso a las fuentes de datos, pero no almacena los datos, es decir, proporciona una vista global unificada del conjunto de bases de datos a integrar a través de un esquema (virtual). Dicho esquema se almacena en el FDBMS, de esta manera los usuarios globales no tienen la necesidad de conocer los detalles de las bases de datos locales.

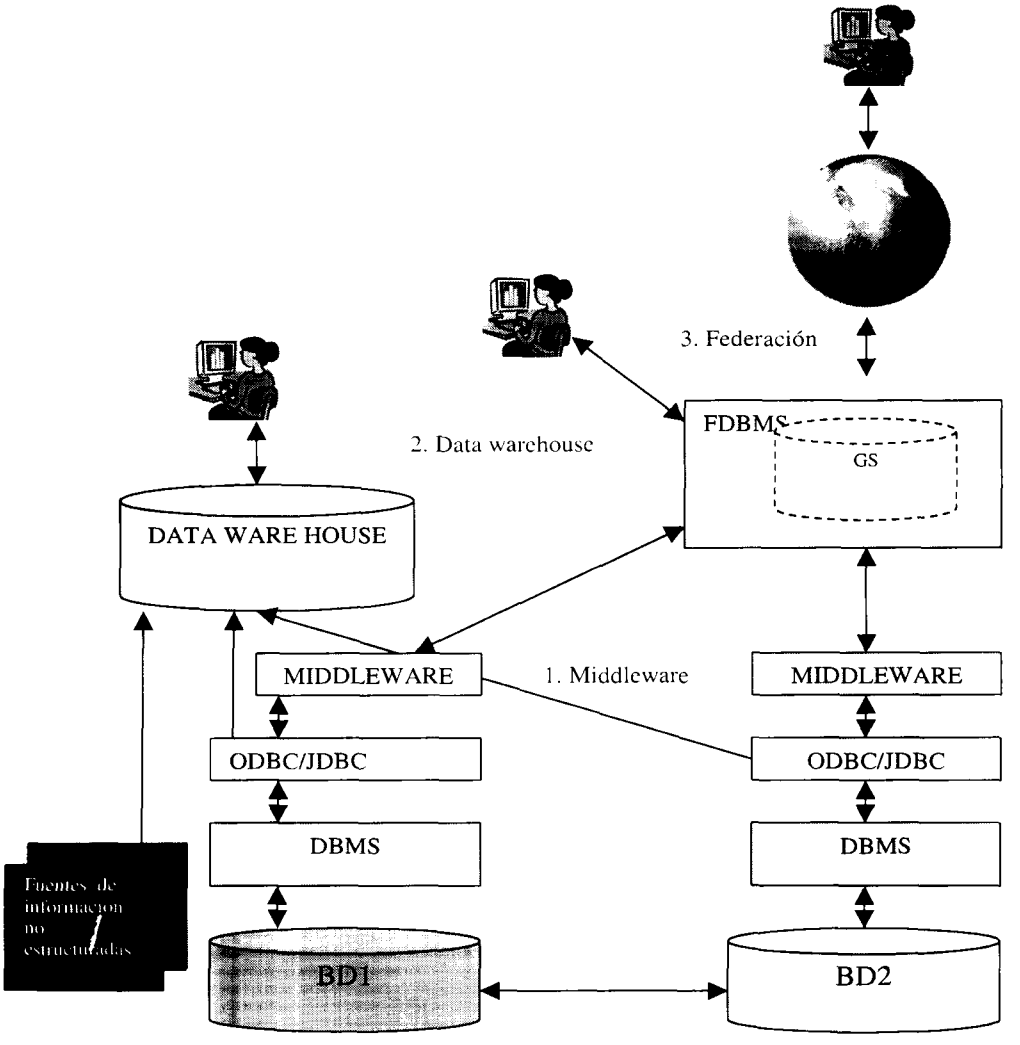

Figura 3.1: Principales herramientas para integración de bases de datos.

3.2 Middleware

El *middleware* es un tipo de software que facilita la comunicación entre dos o más sistemas de software [Linthicum 2000] y proporciona un método sencillo que permite a una entidad, ya sea una base de datos o aplicación comunicarse con otras entidades.

El middleware oculta las complejidades del sistema fuente y destino, de esta manera se libera a los desarrolladores de concentrarse en las API's de bajo nivel y los protocolos de redes y ayuda en enfocarse en el desarrollo de compartir información. El middleware es la mejor manera de trasladar información entre las aplicaciones y bases de datos. Desafortunadamente el middleware tiene muchas limitaciones en resolver los principales retos de integración de bases de datos que se presentan hoy en día, dejando la mayor parte del trabajo a los desarrolladores. Algunas de estas limitaciones son las siguientes:

- Están enfocados a integrar aplicaciones, pero no bases de datos.
- Es complicado integrar semánticamente bases de datos sólo con middleware.

Aunque el middleware no proporciona una solución completa para integrar bases de datos pueden ser utilizados en combinación con otras herramientas, en el capítulo 6 se muestra cómo se utilizó el middleware para apoyar en la integración de bases de datos.

El middleware evoluciona constantemente, y con las nuevas tecnologías surgen nuevos tipos de middleware. En esta tesis se presentan los tipos de middleware que pueden apoyar en el proceso de integración de bases de datos, los cuales son los siguientes:

- RPC (*Remote Procedure Call*)
- MOM (*Mesage Oriented Middleware*)
- Objetos distribuidos (*Distributed Objects*)
- Middleware orientado a bases de datos (*Database Oriented Middleware*)
- Web Services

Algunos de estos tipos no tienen una relación directa con la integración de bases de datos como los RPC's o MOM's, pero en esta tesis se presentan, ya que fueron la base de los siguientes tipos de middleware como objetos distribuidos, web services, o middleware orientado a bases de datos, lo cuales si pueden apoyar en la integración de bases de datos.

A continuación se presenta una descripción de cada uno de estos tipos de middleware.

3.2.1 RPC (*Remote Procedure Call*)

Los RPC's son el tipo de middleware más antiguo, también son los más fáciles de comprender y usar. Básicamente, los RPC's proporcionan a los desarrolladores la capacidad de invocar a una función dentro de un programa que se encuentra en otra máquina remota, para el usuario dicha función aparenta ser ejecutada localmente.

Los RPC´s son síncronos, es decir, detienen la ejecución del programa para poder ejecutar la llamada al procedimiento remoto, por esta razón los RPC´s son conocidos como *middleware de bloqueo*. Debido a que ejecutar una llamada a un procedimiento remoto produce demasiado *overhead*, los RPC´s requieren más ancho de banda que otros tipos de productos middleware.

Actualmente muchos sistemas UNIX incluyen librerías para RPC´s y herramientas como parte de sus sistemas operativos básicos. El mejor tipo de RPC conocido es el DCE (*Distributed Computing Environment*) de la OSF (*Open Software Fundation*). DCE proporciona un mecanismo muy sofisticado, el cual contiene muchas capas de servicio, tales como de seguridad, de directorios, y la capacidad para mantener la integridad de las aplicaciones.

Shoffner [Shoffner 2000] clasifica a CORBA y RMI como RPC, pero también se les puede clasificar como middleware de objetos distribuidos.

Una de las desventajas de este tipo de middleware es la escalabilidad, ya que la integración entre aplicaciones es entre dos procesos, lo cual hace imposible agregar nuevos procesos a la arquitectura. Y la otra desventaja es que requieren una gran capacidad de procesamiento debido a la comunicación síncrona.

Debido a que este tipo de middleware integra sólo dos procesos, es muy difícil hacer una arquitectura para integrar bases de datos, donde se requiere acceder a un número variable de bases de datos. Sin embargo, este tipo de middleware es la base de los siguientes tipos, los cuales sí pueden apoyar al proceso de integración de bases de datos.

3.2.2 MOM (*Message Oriented Middleware)*

Todo middleware que utiliza mensajes se dice que es un MOM, este tipo de middleware fue creado para cubrir las deficiencias de los RPC a través del uso de los mensajes, pues con esta característica no se requiere una sincronía entre los procesos que se integran.

En contraste a los RPC´s, los MOMs no modelan los mensajes como llamadas a métodos, en lugar de esto, los MOM´s modelan los mensajes como eventos [Shoffner 2000].

Mediante el uso de MOM´s los mensajes pueden ser enviados a través de la red sin esperar una respuesta instantánea de la aplicación que recibe el mensaje, por lo que es posible continuar con el funcionamiento de la aplicación que envió el mensaje. De este modo se dice que MOM es una aplicación asíncrona. El empleo de este paradigma es más conveniente para los desarrolladores ya que no bloquea la aplicación desde el punto de vista de procesamiento, sin embargo el modelo es más complejo que el modelo síncrono.

La ventaja de los mensajes puede verse en que son pequeñas unidades de información y son fáciles de manipular. Estos mensajes tienen una estructura y contenido, en pocas palabras pueden describirse como bases de datos de un registro que se mueven entre las aplicaciones a través de mecanismos de envío de mensajes. Existen dos modelos soportados por MOM: punto-a-punto y MQ (*message queuing*), aunque en últimas fechas se ha estado desarrollando el modelo de publicación/suscripción (*pub/sub*).

El más empleado de estos dos modelos es el segundo (MQ), ya que permite a cada programa realizar todos sus procesos sin interrupción alguna al poner los mensajes en una cola de espera. Otra ventaja que se puede apreciar es que una aplicación puede enviar el mensaje remotamente a otras aplicaciones sin esperar que se encuentren corriendo o funcionando.

La desventaja que tiene este tipo de middleware es que la manera de programar con MOM es nueva lo cual representa una desventaja para los desarrolladores quienes se tienen que familiarizar con este tipo de herramientas.

3.2.3 Objetos Distribuidos (*Distributed Objects*)

Los *objetos distribuidos* también son considerados middleware porque facilitan la comunicación entre aplicaciones.

Los objetos distribuidos son pequeños programas de aplicación que usan interfaces estándar y protocolos para la comunicación [Linthicum 2000]. Por ejemplo, los desarrolladores pueden crear objetos distribuidos de CORBA (*Common Request Object Broker Architecture*) que trabajan sobre un servidor UNIX y otros que trabajen sobre un servidor NT. Ya que ambos objetos son creados usando un estándar (CORBA) y ambos objetos utilizan el mismo protocolo de comunicaciones, (Inter-ORB), entonces los objetos son capaces de intercambiar información y utilizar funciones que se encuentran en máquinas remotas.

Actualmente existen varios tipos de objetos distribuidos en el mercado, algunos de los cuales son los siguientes:

- CORBA fue creada por la OMG en 1991, la cual es más estándar que tecnología. Ésta provee las especificaciones sobre las reglas que los desarrolladores deben seguir para crear un objeto distribuido de CORBA. CORBA y Java en conjunto se pueden utilizar para integrar bases de datos. Un ejemplo de este tipo de trabajos es [Xuequn 1999], el cual presenta un proyecto realizado por Telekom en el cual diseñaron una arquitectura llamada VHDBMS. Otro ejemplo [Bouguettaya et. al. 1999], donde también se propone una arquitectura para la integración e interoperabilidad de bases de datos con la ayuda de Java y CORBA, pero en este caso es para trabajar sobre Internet.
- COM y DCOM son estándares de Microsoft [Microsoft 1996]. De la misma manera de CORBA, COM y DCOM proporcionan las reglas para crear objetos

distribuidos de COM. Estas reglas incluyen estándares para las interfaces y protocolos de comunicación. COM y DCOM son más nativos a los sistemas operativos de Windows por lo que son más homogéneos en naturaleza, lo cual es una limitante. COM ayuda la comunicación entre componentes en una misma máquina, y su extensión es DCOM que proporciona la capacidad para la comunicación entre procesos que están en diferentes maquinas a través de un protocolo estándar de red DCOM. Es posible utilizar la plataforma .NET [Microsoft 2003] en conjunto con DCOM para desarrollar aplicaciones distribuidas, donde .NET soluciona la lógica de la aplicación local y DCOM la integración con otras aplicaciones.

- RMI es otra alternativa de objetos distribuidos comparable a CORBA, pero RMI es nativo del lenguaje Java, lo cual es una limitante ya que frecuentemente se requiere trabajar en ambientes heterogéneos. Aunque RMI y CORBA brindan soluciones similares, en la practica la implementación de objetos distribuidos con RMI es más fácil que con CORBA, ya que es fácil de usar y escribir [Sun 2002]. Otras de las ventajas de RMI es que proporciona la seguridad y portabilidad de Java en el cómputo distribuido [Sun 2002].

3.2.4 Middleware orientado a bases de datos (*Database Oriented Middleware)*

Para Linthicum [Linthicum 2000] el *middleware orientado a bases de datos* es un middleware que facilita la comunicación con una base de datos, ya sea desde una aplicación o entre bases de datos. Los desarrolladores típicamente usan este tipo de middleware como un mecanismo para extraer información ya sea de bases de datos remotas o locales.

Los primeros desarrollos de este tipo de tecnología proveían un mecanismo propietario para acceder a los datos; a este tipo de tecnología se le denominaba middleware nativo de bases de datos. La principal desventaja es que sólo podía comunicarse con un tipo de dato.

Sin embargo, en los últimos años los mecanismos empleados con el acceso a datos se realizan a través de estándares como ODBC (*Open Database Connectivity*) o JDBC (*Java Database Connectivity*). Este tipo de middleware, denominado Interfaz de Nivel de Llamada (*Call Level Interface*) ofrece una solución de integración dentro del nivel de datos ya que permite acceder directamente a diferentes fuentes de datos o a las bases de datos sin profundizar en la funcionalidad o las interfaces de presentación de una aplicación.

3.2.5 Web Services

Los *web services* son componentes de software diseñados para soportar la interacción de máquina a máquina en una red [W3C 2004]. Debido a que los web services están basados en Web, son asíncronos, facilitan la escalabilidad de las aplicaciones. Con base en el concepto de EAI [Linthicum 2000], los web services pueden hacer una integración a nivel método, dicho nivel de integración también se puede lograr con el middleware de objetos distribuidos como CORBA o RMI. Por lo tanto, es posible implementar arquitecturas basadas en CORBA o RMI con web services. La diferencia que tienen los web services con el middleware de objetos distribuidos es que los web services están basados en protocolos estándar Web como HTTP, XML y SOAP.

Actualmente cada DBMS pueden estar formado por un conjunto de web services [Gray 2004], por lo tanto se puede lograr integración de bases de datos por medio de estos, por ejemplo: si se tiene una arquitectura de bases de datos donde cada DBMS local puede ser accedido por web services se puede proporcionar transparencia de DBMS´s. Sin embargo, se tiene la limitante de no solucionar la heterogeneidad semántica lo cual es un obstáculo para integrar un conjunto de bases de datos.

3.3 Data warehouse

El data warehouse también se puede utilizar como un medio para integrar bases de datos. Con base en [Inmon 1996] las características de un data warehouse son:

- *Integrado*: Los datos almacenados en el data warehouse deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas. La información suele estructurarse también en distintos niveles de detalle para adecuarse a las distintas necesidades de los usuarios.
- *Temático:* Sólo los datos necesarios para el proceso de generación del conocimiento del negocio se integran desde el entorno operacional. Los datos se organizan por temas para facilitar su acceso y entendimiento por parte de los usuarios finales. Por ejemplo, todos los datos sobre clientes pueden ser consolidados en una única tabla del data warehouse. De esta forma, las peticiones de información sobre clientes serán más fáciles de responder dado que toda la información reside en el mismo lugar.
- *Histórico:* El tiempo es parte implícita de la información contenida en un data warehouse. En los sistemas operacionales, los datos siempre reflejan el estado de la actividad del negocio en el momento presente. Por el contrario, la información almacenada en el data warehouse sirve, entre otras cosas, para realizar análisis de tendencias. Por lo tanto, se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.
- *No volátil:* El almacén de información de un data warehouse existe para ser leído, y no modificado. La información es por tanto permanente, significando la actualización del data warehouse la incorporación de los últimos valores que

tomaron las distintas variables contenidas en él sin ningún tipo de acción sobre lo que ya existía.

Actualmente existen varias herramientas para crear data warehouses, dos de las cuales son las siguientes:

- *Oracle Warehouse Builder* (**OWB**) [Oracle 2003]: Esta herramienta permite a los diseñadores y desarrolladores diseñar, instanciar, manejar y mantener los data warehouses de una empresa. Desde un enfoque de integración de bases de datos, OWB brinda las principales ventajas de los data warehouses, es decir, la extracción, transformación y carga de los datos (**ETL**) en un repositorio de datos. Estas características sólo son unas de las necesidades de integración de bases de datos que las compañías pueden tener, ya que no solamente se deben extraer los datos sino que en muchos casos se requiere la modificación de las fuentes heterogéneas de datos, lo cual no es posible mediante un data warehouse.
- *Microsoft SQL Server 2005* [Microsoft 2004]: Proporciona una plataforma para inteligencia de negocios, la cual incluye herramientas ETL, lo cual permite ver la información de la empresa integrada en un data warehouse, a través del cual es posible organizar la información por cada departamento de las organizaciones.

El data warehouse no proporciona herramientas para la solución de la heterogeneidad semántica. Debido a que uno de los objetivos de este trabajo de tesis es solucionar la heterogeneidad semántica, no se optó por implementar un data warehouse. Sin embargo, un data warehouse contiene la información de la empresa integrada, es por esta razón que se le considera una opción para integrar bases de datos.

3.4 Federación

En esta sección se presenta un análisis de los sistemas de bases de datos federados los cuales permiten ver al conjunto de bases de datos de una manera integrada y permiten el acceso a las bases de datos locales.

Un *sistema de bases de datos federado* (**FDBMS**) es una colección de DBMS´s autónomos y posiblemente heterogéneos cooperando entre sí [Sheth y Larson 1990]. Una federación de bases de datos se hace necesaria cuando se necesitan ver a las fuentes de datos como un sólo sistema de bases de datos, pero al mismo tiempo con la capacidad de poder acceder dichas fuentes para modificarlas o actualizarlas.

Actualmente existen herramientas en el mercado para crear sistemas de bases de datos federados, en esta sección se describen las siguientes:

- ECDA (*Enterprise Connect Data Access*)
- IBM DB2 Information Integrator

En [Hayes 2003] se muestran las características que un FDBMS debe tener, las cuales son las siguientes:

- *Transparencia:* Si un sistema federado es transparente, el usuario no puede ver las diferencias, idiosincrasias, e implementaciones de las fuentes de datos. Idealmente, éste hace que el conjunto de DBMS´s federados se vea como un solo sistema. El usuario no necesita conocer dónde se almacenan los datos (transparencia de localización), el lenguaje de programación de la interfase (transparencia de invocación), si se usa SQL, qué dialecto de SQL soportan las fuentes de datos (transparencia de dialecto), cómo son almacenados los datos, o si son particionados o replicados (transparencia de fragmentación y replicación). El usuario sólo debe de ver una interfase simple y uniforme. A esta interfase también se le conoce como esquema global.
- *Heterogeneidad:* La heterogeneidad es el grado de diferenciación de las fuentes de datos, como se mencionó anteriormente pueden existir diferentes tipos de heterogeneidad desde hardware, software, DBMS, modelos de datos, etc. Un buen FDBMS debe soportar diferentes tipos de heterogeneidad.
- *Apertura:* Todos los sistemas necesitan evolucionar con el tiempo. En los sistemas federados se requiere que tengan la capacidad de agregar fuentes de datos nuevas. Por ejemplo en una federación de datos sobre líneas aéreas y sus vuelos, podría iniciar con cierto número de líneas pero con el tiempo otras nuevas líneas aéreas se crearán y necesitarán agregar sus bases de datos a la federación para que los usuarios puedan conocer los vuelos así como reservar lugares en los vuelos.
- *Autonomía:* Típicamente una fuente de datos tiene aplicaciones y usuarios. Es importante que las operaciones que se realicen en el sistema federado no deben afectarlos, es decir, la implementación de un FDBMS no debe requerir cambios en las aplicaciones de las fuentes de datos.

3.4.1 ECDA (Enterprise Connect Data Access)

ECDA es una herramienta para crear un FDBMS de Sybase. La arquitectura de esta herramienta permite acceder a bases de datos heterogéneas. Los componentes de ECDA ayudan a las aplicaciones a interactuar con bases de datos Sybase y otras, para acceder, consolidar o actualizar los datos. Es posible implementar esta herramienta cuando la heterogeneidad de las bases de datos sea sólo por DBMS. Para solucionar la heterogeneidad semántica no incluye algún componente que apoye en este proceso. Esta herramienta también permite desarrollar aplicaciones adicionales usando una interfase común. Los principales componentes de ECDA constan de un producto base y una serie de componentes.

Las características o componentes de ECDA son las siguientes:

- *Enterprise Connect Data Access Base:* Proporciona los bloques para conectividad entre clientes basados en LAN y fuentes de datos empresariales. Este producto permite acceder transparentemente a fuentes de datos múltiples como si los datos fueran almacenados en una sola base de datos y ejecuta *joins* heterogéneos entre múltiples plataformas.
- *Component Integration Services (ASE/CIS, formalmente conocido como OmniConnect):* Trabajando con DirectConnect ayuda a las aplicaciones a obtener y cambiar datos en múltiples servidores de datos heterogéneos. ASE/CIS proporciona una vista consolidada de los datos empresariales sin la necesidad de conocer la localización y el tipo de estos. Con una consulta se puede acceder y combinar datos desde múltiples fuentes de datos a través de la empresa como si los datos fueran almacenados en una base de datos simple.
- *jConnect for JDBC:* Con el uso del protocolo Sybase´s Native Tabular Data Stream (TDS) [Sybase 2001] se proporciona a los desarrolladores de Java un alto desempeño en el acceso directo a la familia de productos de Sybase.
- *Open Client:* Proporciona las interfaces que las aplicaciones necesitan para comunicarse con Adaptive.
- *Open Server:* Proporciona las librerías, herramientas e interfaces que simplifican el diseño e implementación de aplicaciones servidoras en un ambiente cliente-servidor.

Arquitectura de ECDA

Sybase proporciona varias opciones en el acceso a los datos para satisfacer un amplio rango de requerimientos desde simples hasta más elaborados. En [Sybase 2001] se presentan las opciones más importantes que Sybase propone para la integración de bases de datos:

- *Two-Tiered Data Access:* La mayor parte de este tipo de soluciones involucran uno o más controladores ODBC o JDBC. Este tipo de solución es importante cuando sólo existe una o dos bases de datos destino. Sin embargo, cuando se necesita movimiento y acceso transparente de los datos, estos controladores resultan insuficientes ya que no proporcionan funcionalidad adicional.
- *Multi-Tiered Gateway Data Access (Direct Connect):* Cuando se agregan más bases de datos locales se requiere un nivel mayor de control y transparencia de datos, lo cual no es posible con la solución Two-Tiered, ya que mantener muchos controladores ODBC/JDBC de muchas fuentes de datos destino es costoso y tardado. Para solucionar esto se requiere de una capa de transparencia entre los diferentes controladores ODBC/JDBC para reducir la complejidad. En la figura 3.2 se muestra la arquitectura en la cual se puede notar como ECDA soporta esta simplificación utilizando sólo un controlador para el acceso a todas las bases de datos.

- *Transparent Data Access:* Cuando se usa DirectConnect o JDBC/ODBC, cada cliente debe conocer la base de datos destino. Cuando una aplicación cliente necesita fácilmente encontrar datos sin conocer su localización o plataforma, tal como se muestra en la figura 3.3, Sybase Adaptive Server Enterprise (ASE) usada con uno o más componentes de DirectConnect es una solución. Este mejora el desempeño y proporciona acceso fácil (especialmente si las bases de datos están cambiando) sin requerir cambios en la aplicación ya que todos los datos aparecen en la aplicación como si estos residieran en la base de datos ASE. Esta base de datos tiene una característica llamada Component Integration Services (CIS), lo cual habilita la definición de "Proxy tables", las cuales son mapeadas a tablas actualizadas o vistas en bases de datos heterogéneas.

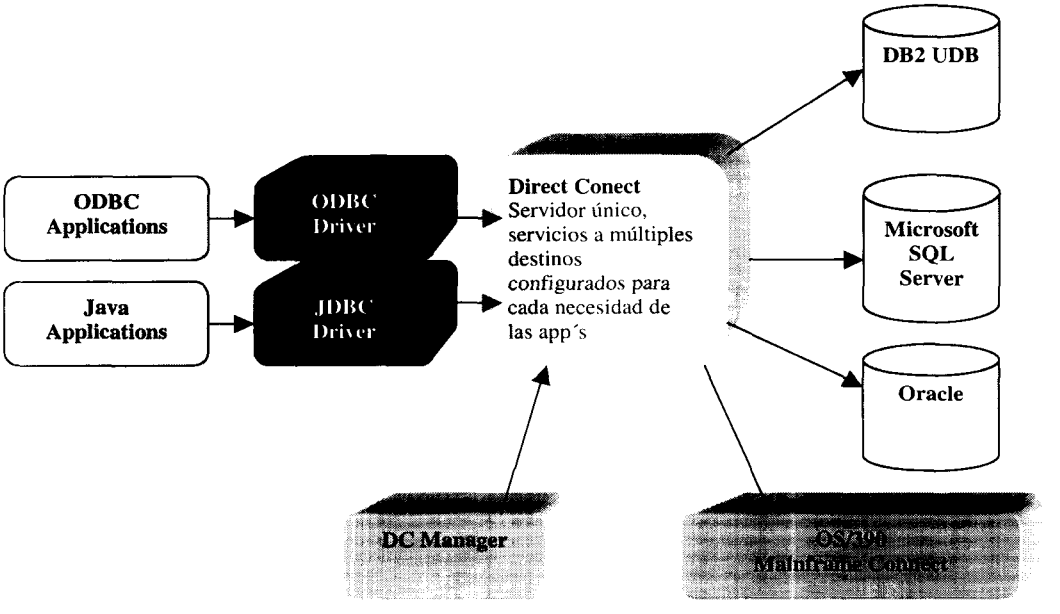

Figura 3.2: Multi-tiered Gateway Data Access con Direct Connect [Sybase 2001]

Disponibilidad de ECDA

Aunque actualmente no existe una sucursal Sybase en México, se pueden obtener los productos mencionados en esta sección así como soporte técnico a través de su sitio de Internet:http://www.sybase.com/products/middleware/enterpriseconnectdataaccess. Existen diferentes versiones de estos productos para las plataformas de Windows, IBM, Sun Solaris y otras.

Limitaciones de ECDA

Enterprise Connect Data Acces de Sybase es una de las herramientas para integración de bases de datos más completas que existen en el mercado ya que permite:

- Acceso transparente a las bases de datos, es decir, los datos aparentan residir en la base de datos ASE.
- Capacidad para actualizar las fuentes de datos.
- Federación de las bases de datos.
- Replicación entre bases de datos heterogéneas.

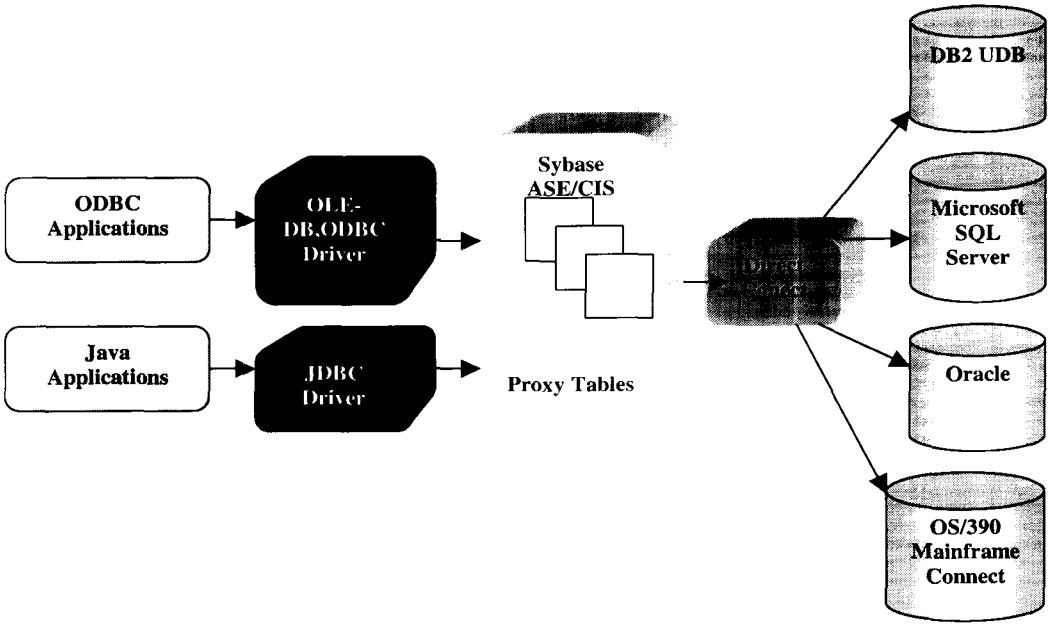

Figura 3.3: ASE/CIS proporcionando acceso transparente a fuentes de datos heterogéneas.[Sybase 2001]

Sin embargo, ECDA no incluye herramientas automáticas para la solución de conflictos de esquema o de datos como diferentes expresiones o unidades. En el prototipo implementado en este trabajo de tesis, se permite la solución de este tipo de conflictos mediante el uso de un esquema global y reglas globales de transformación.

3.4.2 IBM DB2 Information Integrator

IBM DB2 Information Integrator es una herramienta para crear FDBMS´s. Los beneficios o ventajas de IBM DB2 Information Integrator para integrar bases de datos [Hayes 2003] son:

- *Una sola vista para los datos:* Ya que es posible ver a los datos de las bases de datos como si existieran de manera local.
- *Acceso flexible:* Es posible que los clientes accedan a las bases de datos mediante ODBC, JDBC, Web services, interfaces de clientes nativas o de clientes asíncronos, y es posible utilizar los lenguajes para consultas:
    a. Lenguaje de consultas estructurado SQL.
    b. Xquery, el estándar para acceder a datos XML.
    c. Interfaces de aplicación orientadas a objetos IBM DB2 que soportan el ciclo de vida de la administración del contenido.
- *Federación:* La federación de IBM, cuya arquitectura está ilustrada en la figura 3.4, provee una federación sobre fuentes de datos diversas de tal manera que pueden ser vistas y manipuladas como si estuvieran en una sola fuente pero respetando la autonomía e integridad de las bases de datos.
- *Búsquedas:* La infraestructura de IBM permitirá búsquedas y consultas incluyendo la capacidad para rastrear la Web, indexar documentos, federar los resultados desde máquinas de búsqueda múltiples, categorizar y resumir los documentos textuales para un acceso inteligente y entender las semánticas.

Arquitectura

Haas en su trabajo [Haas y Lin 2002] muestra que un sistema federado de IBM se crea instalando una máquina federada y después configurándolo para que se pueda "entender" con las fuentes de datos.

Tal como se muestra en la figura 3.4 el sistema federado consta de los siguientes componentes:

- *Wrapper:* Es un programa para desplegar datos de XML, para la fuente de datos. Para crearlo se tiene que decir a la máquina federada de IBM dónde encontrarlo lo cual es hecho por una sentencia `CREATE WRAPPER`. En caso de que se requieran muchas fuentes de datos del mismo tipo, sólo se requiere un wrapper. Por ejemplo, aun si el sistema federado incluirá cinco bases de datos de Oracle, posiblemente en diferentes máquinas, sólo se requiere un wrapper para Oracle.
- *Servidores:* Cada fuente de datos debe ser identificada mediante la sentencia `CREATE SERVER`. Si hay cinco bases de datos de Oracle entonces se requieren cinco sentencias `CREATE SERVER`.
- *Middleware federado:* Los datos de las fuentes remotas deben ser descritos desde el punto de vista del modelo de datos del middleware federado. Debido a que el sistema federado de IBM soporta el modelo de datos objeto relacional, cada

colección de datos de una fuente externa debe ser descrita a la máquina federada como tabla con columnas de los tipos apropiados.

- *Nicknames:* Es una colección de datos externos modelados como una tabla y sus nombres de tabla y columnas son usados por las aplicaciones de la federación. Los nicknames son identificados con la sentencia CREATE NICKNAME. Por ejemplo para crear un nickname de información relacionada con el clima se realiza la siguiente sentencia de SQL:

```
CREATE NICKNAME weather
        (zone integer, climate varchar(10),  yearly_rainfall float)
           SERVER weather_server OPTIONS(Query_METHOD 'GET')
```

Disponibilidad

Se puede obtener una copia de IBM DB2 II así como el manejador de bases de datos DB2 en la página de Internet del sitio de IBM:
http://www-3.ibm.com/software/swprod/swprod.nsf/(BuildHTBPage)?OpenAgent&DocID=DBLH-5MPKJD.

En esta página también se encuentran los contactos para recibir asesoría personalizada del personal de IBM México. También es posible obtener el software de prueba en el sitio de Internet:
http://www7b.software.ibm.com/dmdd/library/demos/0203xperanto/0203xperanto.html

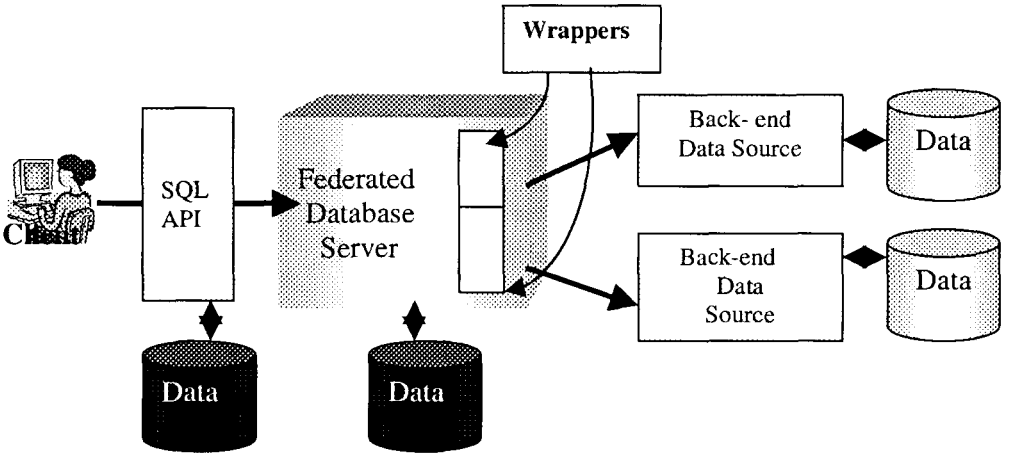

Figura 3.4: Arquitectura del sistema federado de IBM DB2 II. [Haas y Lin 2002]

Limitaciones

Al igual que las herramientas de Sybase, para la integración de bases de datos, IBM DB2 II proporciona ventajas como el acceso a las fuentes heterogéneas de datos, actualización, cambios, replicación y federación. Sin embargo, la federación de IBM DB2 Information Integrator no une los datos del conjunto de fuentes heterogéneas, sino que los maneja en tablas separadas de datos, es decir, cada nickname es una tabla de datos de sólo una fuente de datos. De esta forma el sistema federado de IBM muestra tablas redundantes, las cuales deberían manejarse en el sistema federado como una sola tabla.

3.5 Comparación de las herramientas para integrar bases de datos de acuerdo a los requerimientos que satisfacen

En la sección 3.1 ya se han presentado los requerimientos para integrar bases de datos, también ya se han presentado las herramientas para integrar bases de datos. Esta sección presenta una comparación de las herramientas de integración de bases de datos de acuerdo a los requerimientos que satisfacen. La tabla 3.1 presenta dicha comparación, cada columna en la tabla representa una herramienta y cada renglón representa un requerimiento. En esta tabla se puede notar que las herramientas permiten tener una vista global de las bases de datos, pero no permiten la solución de la heterogeneidad semántica.

De acuerdo a la comparación presentada en la tabla 3.1, se puede notar que IBM DB2 Information Integrator se acerca más al concepto de FDBMS, pero no proporciona integración de esquemas ni de datos, lo cual significa que para implementar esta herramienta como una solución de integración se debe hacer un análisis previo de las semánticas de las bases de datos locales y además se debe agregar algún componente de software que permita la integración de los esquemas y datos mediante las semánticas analizadas.

Herramienta / Requerimiento		MIDLEWARE (ODBC/JDBC + RMI/CORBA)	Oracle Warehouse Builder	Microsoft SQL Server Data Warehouse	Sybase Enterprise Connect Data Access	IBM DB2 Information Integrator
Acceso a los datos a través de un esquema global		Si	No	No	Si	Si
Integración de DBMS´s Heterogéneos		Si	Si	Si	Si	Si
Almacenamiento		No	Si	Si	No	No
Uniformidad (Solución de la heterogeneidad semántica)	Creación de un Esquema Global	No	No	No	Si	Si
	Integración de esquemas	No	No	No	No	No
	Integración de datos	No	No	No	No	No

Tabla 3.1: Comparación de las herramientas para integrar bases de datos.

3.6 Conclusión

En este capítulo se han presentado las principales herramientas que pueden satisfacer los principales requerimientos de integración. Finalmente se ha hecho una comparación de las herramientas de acuerdo a los requerimientos que satisfacen. En este capítulo se concluye que aunque existen herramientas que permiten la comunicación, el acceso, la integración de los diferentes DBMS´s, la formación de un esquema global y el almacén de los datos de las bases de datos locales, aun no existen herramientas para integrar los esquemas y sus datos. Además, no existen herramientas que permitan extraer la heterogeneidad semántica de las fuentes de datos para la formación de un esquema global o metadatos que represente al conjunto de bases de datos locales. Para cubrir la falta de una herramienta que permita solucionar estas limitantes, en esta tesis se toma el reto de integrar las bases de datos a partir de un esquema global tomando como entrada el conjunto de los esquemas de las bases de datos locales. Dicho esquema proporciona una vista global y soluciona las redundancias e inconsistencias que se pueden presentar al integrar bases de datos utilizando alguna de las herramientas presentadas en este capítulo. En el siguiente capítulo se presenta el método que se sigue para la creación de este esquema.

Capítulo 4. Integración semántica de datos

Cuando se hace una integración de bases de datos se deben solucionar una serie de problemas relacionados con la heterogeneidad de las bases de datos. El más importante de estos problemas es la *heterogeneidad semántica* de los esquemas de las fuentes de datos a integrar. Las *semánticas* de los datos son los significados que tienen los datos para sus usuarios [Hakimpour y Geppert, 2001]. La *integración semántica* de los datos es la actividad de unir un conjunto de esquemas de bases datos, posiblemente en un esquema global, el cual representa al conjunto de bases de datos, y proporciona a los usuarios una vista global. Los beneficios de esta integración es que los usuarios no tienen que conocer las diferencias en cuanto a estructuras de las bases de datos y sus datos.

En este trabajo de tesis se propone integrar los esquemas de tres bases de datos locales mediante un GS. La descripción de los esquemas de las bases de datos locales y el GS, son descritas en este capítulo. Asimismo, se describe el método a seguir para integrar dichos esquemas.

La organización de este capítulo es la siguiente: en la sección 1 se presentan los requerimientos de integración. En la sección 2 se presentan tres opciones existentes para la integración semántica de datos: integración mediante un esquema global, esquema global físico y sin esquema global. La sección 3 presenta un caso de estudio para un sistema bibliográfico, en el cual se presentan los esquemas de las bases de datos locales, así como los conflictos entre dichos esquemas. En la sección 4 se presenta el esquema global para el caso de estudio. Una introducción a las correspondencias semánticas se presenta en la sección 5. En la sección 6 se propone una metodología para integrar esquemas basada en identificación de correspondencias semánticas, después se presentan los conflictos existentes entre los esquemas locales, y por último, en la se sección 7 se presenta la conclusión de este capítulo.

4.1 Requerimientos de integración

En general, los requerimientos de integración de las bases de datos son los siguientes:

1. Los usuarios necesitan ver al conjunto de bases de datos como si se tratara de una única base de datos.
2. Se requiere solucionar la heterogeneidad semántica a nivel de tabla atributos y datos.
3. Los usuarios globales no sólo deben tener la capacidad de hacer consultas, sino también la capacidad para actualizar los registros de las bases de datos locales.

4. Los usuarios globales requieren información actualizada. Por ejemplo, estos usuarios pueden acceder al precio de un artículo, ese precio debe ser actualizado. No es correcto presentar un precio no vigente.
5. Los puntos anteriores se deben cumplir respetando la autonomía de las bases de datos locales, lo cual significa que estas bases de datos no deben ser cambiadas en cuanto a su estructura original.

4.2 Análisis de opciones de integración

Las principales opciones de integración semántica existentes son las siguientes:
1. *Esquema global (virtual)*
2. *Esquema global físico*
3. Sin *esquema global*

A continuación se presenta la descripción de cada una de estas opciones así como sus ventajas y desventajas.

4.2.1 Esquema global (virtual)

El *esquema global* es una vista virtual del conjunto de esquemas de las bases de datos a integrar [Batini et. al. 1986, Parent y Spaccapietra 1998], lo cual significa que no almacena datos físicamente, pero permite el acceso a estos.

Las ventajas de integrar por medio de un esquema global son las siguientes:
1. *Proporciona transparencia de semántica a los usuarios:* lo cual significa que dichos usuarios no tienen la necesidad de conocer las estructuras de los esquemas de las bases de datos locales.
2. *Los conflictos entre n-esquemas son reducidos a conflictos entre 2-esquemas:* el esquema global y el esquema de la base de datos local.
3. *Permite el acceso a los datos:* lo cual significa que es posible que un usuario global pueda hacer actualizaciones en las fuentes de datos por medio de este esquema.
4. *Respeta la autonomía de las fuentes de datos:* ya que no es necesario que las fuentes de datos tengan que ser cambiadas en cuanto a su estructura original.

Las desventajas en el uso de este esquema son las siguientes:
1. *El mantenimiento del esquema global es una actividad difícil de realizar:* ya que los cambios en las estructuras de los esquemas de las fuentes de datos locales deben ser reflejados en este esquema, desarrollar el mecanismo para ésto es una tarea compleja.

2. *El esquema global es un medio para establecer las correspondencias semánticas entre los esquemas locales heterogéneos*: Sin embargo, frecuentemente no sólo existen diferencias entre estos esquemas, sino también existe heterogeneidad entre los datos de las bases de datos locales. No es posible establecer esta heterogeneidad sólo con un esquema global. Para dar solución a la heterogeneidad de los datos, es necesario agregar componentes al esquema global, por ejemplo: procedimientos para hacer conversiones de datos, tipos de datos o expresiones, etc.

En este trabajo de tesis se integran semánticamente los esquemas de las bases de datos por medio de un esquema global. Sin embargo, se tiene la limitante de no solucionar la primera desventaja mencionada, ya que es necesario implementar un proceso para que los cambios en las estructuras de los esquemas de las bases de datos locales se reflejen en el esquema global. Sin embargo la segunda desventaja fue solucionada agregando un componente de software que contiene reglas para expresar las correspondencias semánticas a nivel de datos. En el capítulo 5 se describe a detalle el componente utilizado para dar solución a la heterogeneidad semántica de los datos causada por el uso de diferentes tipos, unidades y datos.

4.2.2 Esquema global físico

El *esquema global físico* es un esquema global, el cual además de representar al conjunto de esquemas de las bases datos, tiene la capacidad de almacenar sus datos.

Las principales ventajas de integrar con un esquema global físico son las siguientes:

1. *El esquema global físico representa una vista integrada del conjunto de bases de datos:* Debido a esto, el esquema global físico proporciona las ventajas del esquema global.
2. *Contiene a los datos de las bases de datos:* Significa que cada registro de datos de las bases de datos locales es almacenado en este esquema. Esto es una ventaja importante, ya que proporciona el acceso directo a los datos.
3. *El acceso a los datos* es más rápido debido a que los datos residen en dicho esquema.
4. *Los datos se almacenan en este esquema:* No es necesario desarrollar componentes adicionales para el acceso a los datos. Por ejemplo: si un usuario realiza una operación `SELECT`, la operación se ejecuta directamente en el esquema físico. La operación no tiene que ser trasformada a una llamada SQL en la base de datos fuente.

Las desventajas son las siguientes:

1. *El costo de actualizar los datos de las bases de datos:* Se requiere que los datos sean replicados o enviados desde las fuentes de datos a este esquema.

2. *Se requiere un mecanismo de actualización:* Como las operaciones se realizan directamente en el esquema físico, es probable que los usuarios no obtengan resultados vigentes porque los datos podrían no estar actualizados.
   Un ejemplo para esta desventaja es el siguiente: supóngase que existen tres bases de datos heterogéneas A, B y C las cuales replican su información a un esquema físico EF. EF tiene los datos actualizados de A y B, pero debido a problemas con la red de comunicaciones no tiene actualizado los datos de C. Cuando un usuario global realiza una operación al esquema físico EF dicho usuario no obtiene un resultado actualizado ya que los datos de C no han sido actualizados en EF.
3. *El mantenimiento de dicho esquema también es un problema*: Los cambios en las estructuras de los esquemas de las bases de datos locales deben ser reflejados en este esquema.

En esta tesis no se implementó un esquema físico ya que una de las necesidades de los usuarios globales es obtener la información actualizada y esta es una de las desventajas del esquema global físico.

4.2.3 Sin esquema global

Algunos autores [Litwin 1984, 1990, Litwin et. al. 1982, 1986] sostienen que el esquema global es difícil de mantener debido a la heterogeneidad de las bases de datos locales. Litwin [Litwin 1990] afirma que la heterogeneidad semántica debe ser solucionada mediante un *lenguaje para manejar bases de datos múltiples*. Dicho lenguaje es el sustituto del esquema global. La integración mediante un lenguaje consiste en crear un *esquema conceptual* mediante las sentencias de dicho lenguaje [Litwin 1990]. Por ejemplo: para integrar las bases de datos A, B y C la sentencia sería de esta manera: `CREATE MULTIDATABASE (A,B,C)`. De igual forma, el lenguaje para manejar bases de datos múltiples debe dar soporte para ejecutar operaciones SQL en este esquema, tales operaciones pueden ser tanto de definición del esquema conceptual como manipulación de datos. Las operaciones de definición consisten en agregar nuevas bases de datos o quitarlas. Las operaciones de manipulación de datos consisten en ejecutar llamadas SQL en el esquema conceptual.

Las ventajas de esta opción de integración son las siguientes:
1. *No requiere el mantenimiento de un esquema global.*
2. *Respeta la autonomía de las bases de datos locales.*
3. *Flexibilidad para trabajar con un número variable de bases de datos:* Debido a que no existe un esquema global, en cualquier momento es posible agregar o quitar bases de datos utilizando el lenguaje para manejar bases de datos múltiples.
4. *Esta opción es apta para integrar bases de datos en Internet*: ya que la información que se encuentra en esta red cambia constantemente y trabajar con un esquema integrado se vuelve una actividad demasiado complicada.

Las desventajas son las siguientes:

1. *Los lenguajes para trabajar con bases de datos múltiples comerciales que se han creado son muy limitados.*
2. *La creación de un lenguaje para manejar bases de datos múltiples es una actividad más compleja que la creación de un esquema global.*

4.3 Descripción del caso de estudio: Sistema Bibliográfico

En esta sección se presenta un caso de estudio que requiere integración de bases de datos, el cual consiste en crear un sistema bibliográfico con los artículos científicos relacionados con el área de computación. Estos artículos se encuentran en tres bases de datos heterogéneas: *Biblioteca1, Bibliteca2* y *Biblioteca3.*

La integración de estas tres bases de datos es necesaria ya que las bases de datos pertenecen a diferentes universidades las cuales requieren compartir sus artículos científicos. Al compartir estos documentos los alumnos de estas tres universidades pueden acceder a los artículos científicos de todas las bibliotecas.

Los artículos se encuentran registrados y almacenados en estas tres bases de datos. Sin embargo, las diferencias en estructuras de los esquemas y DBMS de estas bases de datos no permiten alguna solución típica, como la replicación de datos. Por lo tanto, en este trabajo de tesis, se propone que estas bases de datos se integren mediante un esquema global y componentes de software.

Los artículos pueden ser *journals* o *proceedings* de conferencia. Sin embargo, las estructuras de los esquemas de las bases de datos locales presentan diferencias, las cuales tienen diferentes causas:

1. *Las estructuras de los esquemas no representan la clasificación completa*: es decir, algunos esquemas no contienen las tablas para representar los artículos de journals mientras que en otros esquemas no se encuentran las tablas para representar los proceedings.
2. *Otra causa es la utilización de diferentes estructuras para representar la misma tabla*: por ejemplo diferentes nombres y número de atributos.

Para presentar con más detalle cada una de estas diferencias se presentan los esquemas de las bases de datos locales y los conflictos encontrados. La notación utilizada es diagramas de clases UML y relacional.

4.3.1 Esquema de la base de datos Biblioteca1

El esquema de la base de datos *Biblioteca1*, presentado en la figura 4.1, representa artículos de journals y conferencias así como los autores que escribieron estos artículos.

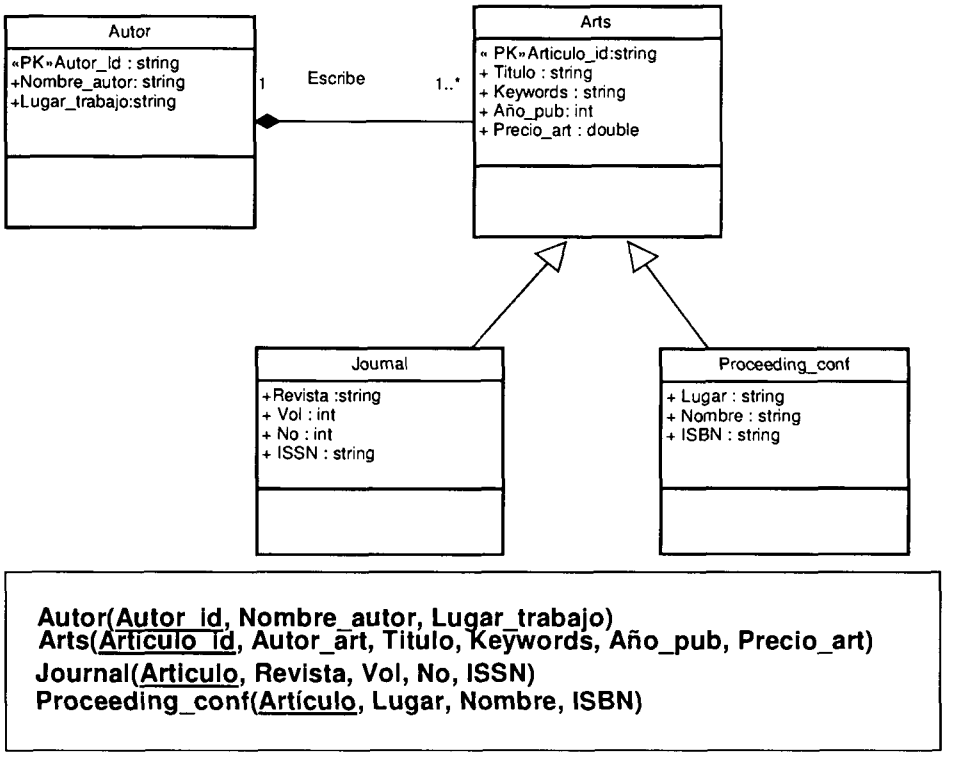

Figura 4.1 Esquema de la base de datos Biblioteca1.

La estructura en clases de este esquema es la siguiente:

1. La clase AUTOR representa a los autores principales de los artículos.
2. Existe una clase genérica ARTS para representar los artículos, la cual contiene dos subclases:
    a. JOURNAL, para representar los artículos journals
    b. PROCEEDING_CONF, para representar los artículos de conferencias
3. La composición entre las clases AUTOR y ARTS nos indican una relación fuerte, es decir, un artículo no puede existir si no esta asociado a un autor.
4. Existe una relación de herencia entre ARTS y sus subclases JOURNAL Y PROCEEDING_CONF.

Los atributos de las tablas relacionales de Biblioteca1 son los siguientes:

- AUTOR:
  - AUTOR_ID: Es la clave que identifica al autor de manera única.
  - NOMBRE_AUTOR: Contiene el nombre y apellido del autor principal de un artículo.
  - LUGAR_TRABAJO: Contiene la ubicación del lugar de trabajo del autor.
- ARTS:
  - ARTÍCULO_ID: Contiene la clave que identifica a cada artículo de manera única.
  - AUTOR_ART: Es la llave foránea del autor principal del artículo.
  - TITULO: Contiene el nombre del artículo.
  - KEYWORDS: Contiene las palabras que describen el tema principal del artículo.
  - AÑO_PUB: Contiene el año en el que fue publicado el artículo.
  - PRECIO_ART: Almacena el precio del artículo en pesos.
- JOURNAL:
  - ARTÍCULO: Es una llave foránea que hace referencia a los datos genéricos de artículo journal.
  - REVISTA: Almacena el nombre de la revista que publica al artículo de journal.
  - NO: Contiene el número de la revista que publica el artículo journal.
  - VOL: Almacena el número de volumen de la revista que publica al artículo journal.
  - ISSN: (Internacional Standard Serial Number), es un número internacional de ocho dígitos para identificar publicaciones periódicas.
- PROCEEDING_CONF:
  - ARTÍCULO: Es la llave foránea que hace referencia a los datos genéricos del artículo de proceeding.
  - LUGAR: Contiene el lugar donde fue presentada la conferencia.
  - NOMBRE: Contiene el título del proceeding.
  - ISBN: (International Standard Book Number) es un número internacional legible por máquinas y que identifica a cualquier libro de manera única. El atributo ISBN almacena dicho número.

Algunos ejemplos de instancias para Biblioteca1 se presentan la tabla 4.1.

Autor

Autor_Id	Nombre_autor	Lugar_trabajo
000001	Dongwon Lee	University of Arizona, Tucson, AZ
000003	Cheng Hian Goh	University of California, Los Angeles, CA

Arts

Artículo_id	Autor_art	Titulo	Keywords	Año_pub	Precio_art
000001	000001	Extracting semantic metadata and its visualization	Database, Metadata, semantics	2001	40
000002	000003	Languages for multi-database interoperability	Heterogeneous databases, metadata	1997	50

Journal

Artículo	Revista	Vol	No	ISSN
000001	ACM Crossroads	7	3	1047-8188

Proceeding_conf

Artículo	Lugar	Nombre	ISBN
000002	Tucson, Arizona, United States	Proceedings of the 1997ACM SIGMOD international conference on Management of data	0-89791-911-4

Tabla 4.1: Ejemplos de instancias para Biblioteca1.

4.3.2 Esquema de la base de datos Biblioteca2

La base de datos *Biblioteca2*, cuyo esquema es presentado en la figura 4.2 contiene sólo artículos de conferencias.

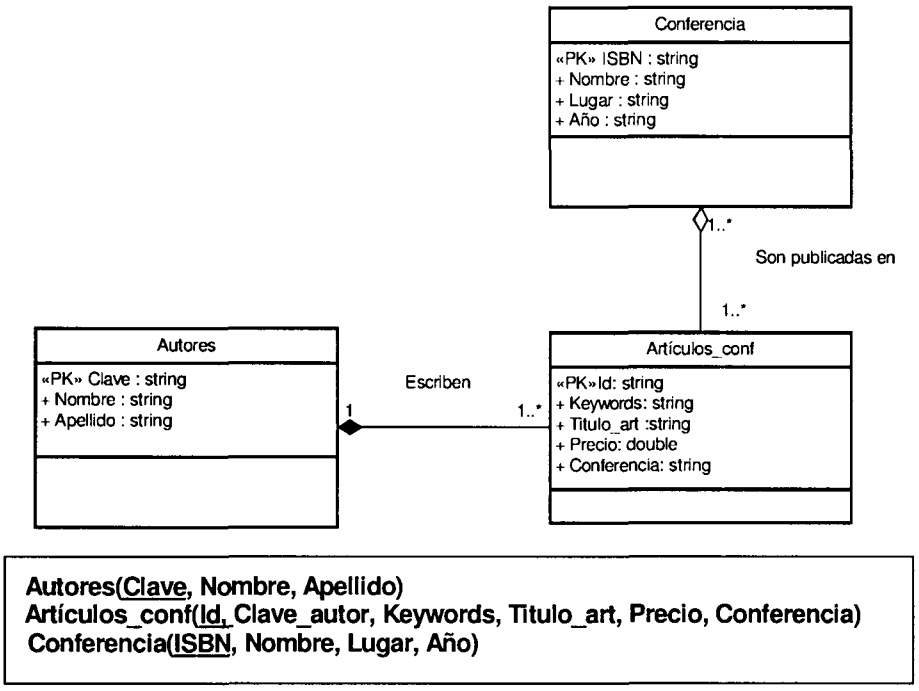

Figura 4.2 Esquema de la base de datos Biblioteca2.

La estructura de las clases de este esquema es el siguiente:

1. La clase CONFERENCIA representa las conferencias donde se presentaron los artículos almacenados en la base de datos Biblioteca2.
2. La clase ARTÍCULOS_CONF contiene proceedings de las conferencias.
3. La clase AUTORES representa a los autores principales de los artículos de proceedings.
4. La relación de composición entre las clases AUTORES y ARTICULOS_CONF indican que un proceeding no puede existir si no es escrito por un autor. Debido a esto, ARTICULOS_CONF contiene la llave foránea de AUTORES.
5. La relación de asociación entre CONFERENCIA Y ARTICULOS_CONF indica que los artículos pueden ser publicados en una o más conferencias y que las conferencias pueden publicar al menos un artículo.

Los atributos de las tablas relacionales de Biblioteca2 son los siguientes:

- AUTORES:
    - CLAVE: Es la llave que identifica al autor de manera única.

- NOMBRE: Contiene el nombre sin apellidos del autor quien escribió al menos un proceeding.
- APELLIDO: Almacena el apellido(sin nombre) del autor principal de un artículo.

- ARTICULOS_CONF:
  - ID: Es la clave para identificar los artículos de proceedings de manera única.
  - CLAVE_AUTOR: Contiene la llave foránea del autor quien escribió el artículo.
  - KEYWORDS: Son las palabras que describen el tema principal del artículo.
  - TITULO_ART: Almacena el título del artículo de proceeding.
  - PRECIO: Contiene el precio en dólares del artículo de proceeding.
  - CONFERENCIA: Es una llave foránea para hacer referencia a la conferencia donde se presentó el proceedings.
- CONFERENCIA:
  - NOMBRE: Almacena el título de la conferencia.
  - LUGAR: Contiene el lugar donde se presento la conferencia.
  - AÑO: Almacena el año en el que fue presentada la conferencia.
  - ISBN: (International Standard Book Number) es un número internacional legible por máquinas y que identifica a cualquier libro de manera única. El atributo ISBN contiene dicho número.

Algunos ejemplos de instancias para Biblitoeca2 se presentan en la tabla 4.2.

Autores

Clave	Nombre	Apellido
000001	Frédéric	Gingras
000006	Farshad	Hakimpour

Articulos_conf

Id	Clave_Autor	Keywords	Titulo_art	Precio	Conferencia
000001	000001	MSQL,interoperability, multi-database	Languages for multi-database interoperability	5	0-89791-911-4
000002	000006	semantics, integration	Resolving semantic heterogeneity in schema integration	9	1-58113-377-4

Conferencia

ISBN	Nombre	Lugar	Año
0-89791-911-4	International Conference on Management of Data and Symposium on Principles of Database	Tucson, Arizona, United States	1997
1-58113-377-4	Conference on Formal Ontology in Information Systems	Ogunquit, Maine, United States	2001

Tabla 4.2: Ejemplos de instancias para Biblioteca2.

4.3.3 Esquema de la base de datos Biblioteca3

La base de datos *Biblioteca3*, cuyo modelo de datos es presentado en la figura 4.3, contiene artículos de journal.

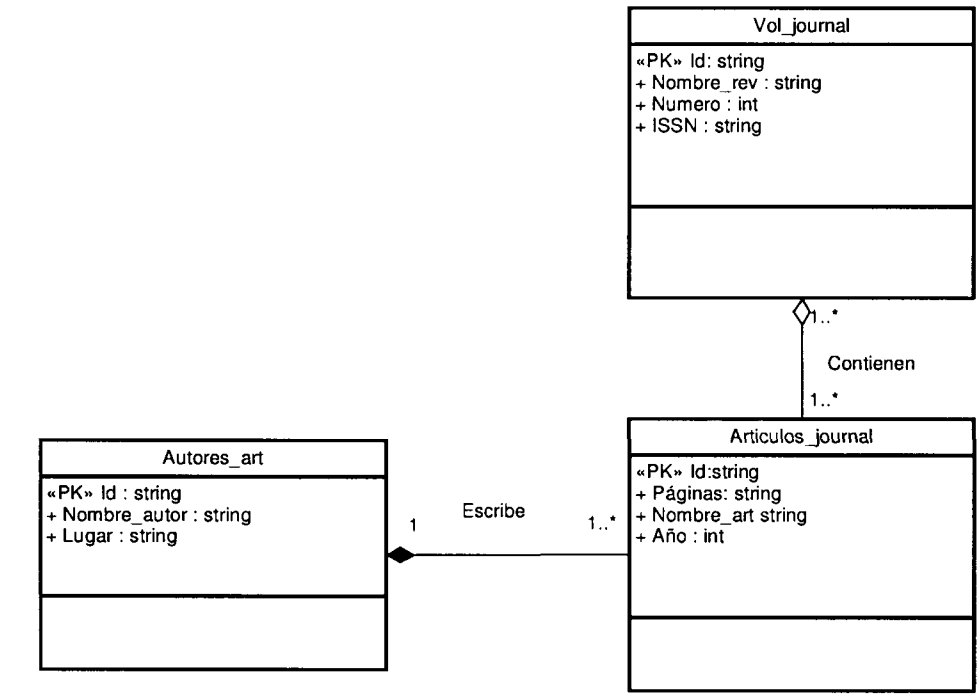

Figura 4.3: Esquema de la base de datos Biblioteca3.

La estructura de las clases del esquema de Biblioteca3 es la siguiente:

1. La clase VOL_JOURNAL representa los volúmenes de las revistas que publican los artículos de journal.
2. La clase ARTICULOS_JOURNAL contiene los artículos de journal publicados en los volúmenes de las revistas.
3. La clase AUTORES_ART representa a los autores principales de los artículos de journal.

Una relación de composición entre AUTORES_ART y ARTICULOS_JOURNAL indica una relación fuerte, es decir, los journals no pueden existir en la base de datos si no contienen una relación con un autor por lo menos. Por lo tanto, la clase ARTICULOS_JOURNAL contiene la llave foránea para referenciar a los autores.

La descripción de los atributos del esquema de Biblioteca3 es la siguiente:

- AUTORES_ARTS:
  - ID: Contiene un número identificador del autor.
  - NOMBRE_AUTOR: Contiene el nombre con apellidos de autor principal del artículo.
  - LUGAR: Almacena la procedencia del autor.
- ARTICULOS_JOURNAL:
  - PÁGINAS: Contiene las páginas de la revista en la que se encuentra el artículo journal.
  - NOMBRE_ART: Almacena el título del artículo journal.
  - AÑO: Contiene el año en el que se publicó el journal.
  - VOL_JOURNAL: Almacena la llave foránea para indicar el volumen donde se encuentra publicado el artículo.
  - AUTOR_ID: Contiene la llave foránea para indicar el autor principal del artículo.
  - ID: Contiene el número identificador del journal.
- VOL_JOURNAL:
  - ISSN: Contiene un número estándar internacional para identificar al volumen. Cada volumen contiene un número identificador ID.
  - NOMBRE_REV: Contiene el nombre de la revista a la que pertenece el volumen.
  - NUMERO: Contiene el número de volumen de la revista.

Algunos ejemplos de instancias para Biblioteca3 se presentan en la tabla 4.3.

Autores_art

Id	Nombre_autor	Lugar
000001	Dongwon Lee	University of Arizona, Tucson, Arizona
000003	Cheng Hian Goh	University of California, Los Angeles, California

Articulos_journal

Id	Nombre_art	Vol_journal	Autor_id	Paginas	Año
00001	Extracting semantic metadata and its visualization	001	000001	10,11	2001
00002	Context interchange: new features and formalisms for the intelligent integration of information	002	000003	1-20	1999

Vol_journal

Id	ISSN	Numero	Nombre_rev
001	1047-8188	7	ACM Crossroads
002	1046-8188	17	ACM Transacions on Information Systems

Tabla 4.3: Ejemplos de instancias de Biblioteca3.

4.3.4 Conflictos entre los esquemas de las bases de datos

Ya se han presentado los esquemas de las bases de datos locales. En esta sección se presenta la heterogeneidad encontrada entre estos esquemas. De acuerdo a la clasificación de conflictos de Kim y Seo [Kim y Seo 1991], los conflictos que se encontraron en las bases de datos locales son los siguientes:

Conflictos de esquema:

1. *Conflictos por diferentes nombres para tablas semánticamente equivalentes*: La tabla AUTOR de Biblioteca1 es la tabla semánticamente equivalente de AUTORES en Biblioteca2 y en Biblioteca3 de AUTORES_ART.
2. *Tablas faltantes*: En Biblioteca3 no se encuentra la tabla semánticamente equivalente a CONFERENCIA de Biblioteca2.
3. *Conflictos por diferentes nombres para atributos semánticamente equivalentes*: El atributo NOMBRE_AUTOR de Biblioteca1 tiene un atributo semánticamente equivalente en Biblioteca2 llamado NOMBRE y en Biblioteca3 es llamado NOMBRE_AUTOR.
4. *Conflictos por atributos faltantes pero implícitos*: La tabla AUTOR de Biblioteca1 tiene el atributo NOMBRE, cuyos atributos correspondientes son NOMBRE y APELLIDO en Biblioteca2, los cuales concatenados forman dicho atributo.
5. *Atributos faltantes*: Por ejemplo el atributo PRECIO_ART de Biblioteca1 no se encuentra en Biblioteca3.
6. *Diferentes tipos de datos:* El atributo AÑO de la tabla ARTICULOS_JOURNAL de Biblioteca3 es de tipo String, mientras que en la tabla ARTS de Biblioteca1 el atributo semánticamente equivalente a éste es de tipo Integer.

Conflictos de datos:

1. *Diferentes unidades*: en Biblioteca1 el campo PRECIO_ART utiliza el tipo de unidad "PESOS", pero en Biblioteca2 su campo semánticamente equivalente PRECIO utiliza el tipo de unidad "DÓLAR".

4.4 Esquema global para el caso de estudio

Supóngase que ya se ha construido un esquema global, cuyo modelo de datos y tablas relacionales son presentados en la figura 4.4.

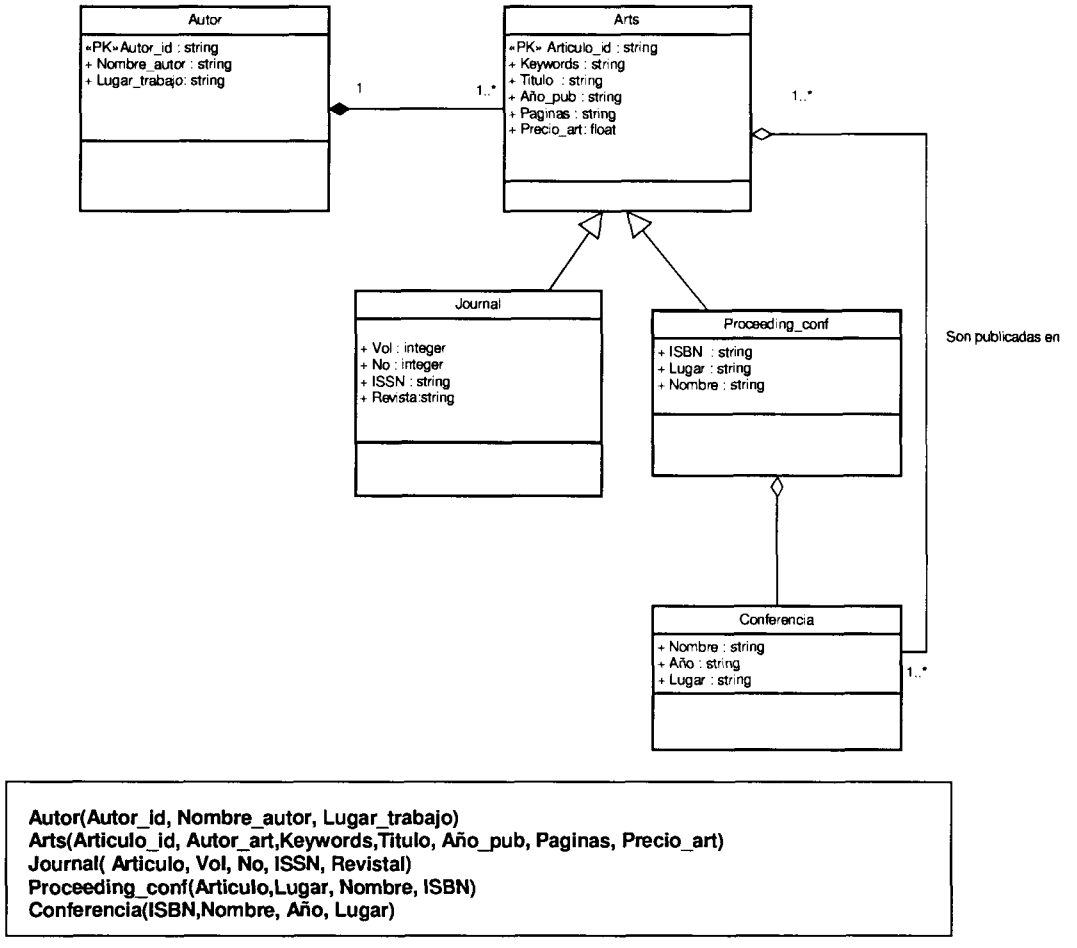

Figura 4.4: Esquema global de las bases de datos locales.

El esquema global es el resultado de la integración del conjunto de esquemas de las bases de datos locales y es la base para analizar la metodología y los pasos que se siguieron para generar su estructura.

A continuación se presenta la descripción de la estructura del esquema global.

El esquema global esta compuesto de las siguientes cinco tablas:

- AUTOR: Almacena el conjunto de los autores principales de los artículos de revistas o conferencias.
- ARTS: Contiene los datos genéricos del conjunto de artículos de revistas o conferencias escritos por los autores.
- JOURNAL: Contiene la descripción del conjunto de artículos de journal que escriben los autores.

- PROCEEDING_CONF: Almacena la descripción de los artículos de proceedings, los cuales son publicados en conferencias.
- CONFERENCIA: Contiene la descripción del conjunto de conferencias donde fueron publicados los proceedings.

A continuación se describen los atributos de las tablas relacionales.

- AUTOR
  - AUTOR_ID: Contiene la clave que identifica a los autores que escriben artículos journal o proceeding.
  - NOMBRE_AUTOR: Contiene el nombre y apellido de los autores de los artículos.
  - LUGAR_TRABAJO: Contiene la ubicación del lugar de trabajo del autor.
- ARTICULOS
  - ARTICULO_ID: Contiene la clave que identifica a los artículos de journal o conferencia.
  - AUTOR_ART: Contiene la llave foránea que hace referencia al autor principal del artículo.
  - KEYWORDS: Contiene las palabras que más se repiten en los articulos.
  - TITULO: Almacena el título del artículo de journal o proceedings.
  - AÑO_PUB: Contiene el año en el que fue publicado el artículo.
  - PAGINAS: Almacena las paginas de la revista donde se encuentran los artículos.
  - PRECIO_ART: Almacena el precio del artículo en dólares.
  - VOL_JOURNAL: Contiene número del volumen de un artículo journal.
- JOURNAL
  - ARTÍCULO: Contiene la llave foránea para hacer referencia a los atributos genéricos de los artículos de journal.
  - VOL: Almacena el número de volumen de la revista que contiene al artículo journal.
  - NO: Contiene el número de revista que contiene el artículo journal.
  - ISSN: (International Standard Serial Number), es un número de ocho dígitos para identificar publicaciones periódicas.
  - REVISTA: Contiene el nombre de la revista del artículo journal.
- PROCEEDING_CONF
  - ARTÍCULO: Contiene la llave foránea que hace referencia a los atributos genéricos al artículo proceeding
  - LUGAR: Almacena el lugar donde se presentó la conferencia del proceeding.
  - NOMBRE: Contiene el titulo del proceeding.
  - ISBN: (International Standard Book Number) es un número internacional legible por máquinas y que identifica a cualquier libro de manera única.
- CONFERENCIA
  - NOMBRE: Contiene el título del artículo donde se presentó la conferencia.
  - AÑO: Almacena el año en el que se presentó conferencia.
  - LUGAR: Contiene el lugar donde se presentó la conferencia.

- ISBN: (International Standard Book Number) es un número internacional legible por máquinas y que identifica a cualquier libro de manera única.

Aunque el esquema global es una vista virtual, es posible presentar ejemplos de instancias para el caso de estudio de este trabajo de tesis. Dichas instancias se presentan en la tabla 4.4. Se puede notar que en algunos registros, algunos de los campos contienen valor nulo, esto significa que dichos campos no existen en alguna de las bases de datos locales, tal es el caso de la tabla ARTS, en donde el campo PAGINAS tiene un registro con valor nulo.

Autor

Autor_id	Nombre_autor	Lugar_trabajo
000001	Dongwon Lee	University of Arizona, Tucson, Arizona
00003	Cheng Hian Goh	University of California, Los Angeles, California

Arts

Artículo_Id	Autor_Art	Keywords	Titulo	Año_Pub	Paginas	Precio_art	Vol_journal
000001	000001	Database, Metadata, semantics	Extracting semantic metadata and its visualization	2001	10,11	4	7
000002	000003	Heterogeneous databases, Metadata	Languages for multi-database interoperability	1997	Null	5	Null

Journal

Artículo	Vol	No	ISSN	Revista
000001	7	3	1047-8188	ACM Crossroads

Proceeding_conf

Artículo	Nombre	Lugar	ISBN
000002	Proceedings of the 1997ACM SIGMOD international conference on Management of data	Tucson, Arizona, United States	0-89791-911-4

Conferencia

ISBN	Nombre	Año	Lugar
0-89791-911-4	International Conference on Management of Data and Symposium on Principles of Database	1997	Tucson, Arizona, United States

Tabla 4.4: Ejemplos de instancias para el esquema global.

4.5 Correspondencias semánticas entre los esquemas de las bases de datos locales

Las *correspondencias semánticas* son las equivalencias de significados entre tablas o atributos [Parent y Spaccapietra 1988, Hakimpour y Geppert 2001]. En este trabajo de tesis se propone que se utilicen las correspondencias semánticas entre las bases de datos locales como un medio para resolver la heterogeneidad semántica entre las bases de datos locales. En esta sección se presenta una introducción a la clasificación y *registro de correspondencias semánticas*. Primeramente se presentan los tipos de correspondencias semánticas que se pueden encontrar en un conjunto de bases de datos locales, y después se presentan dos formas de registrarlas.

4.5.1 Tipos de correspondencias semánticas

Supóngase que un usuario desea ejecutar una consulta en el esquema global para conocer los autores de los artículos cuyo precio sea de "cinco dólares":

```
SELECT Autor.Nombre_autor, Arts.Titulo, Arts.Paginas
FROM Autor, Arts
WHERE Arts.Precio_art = 5
AND Autor.Autor_id = Arts.Autor_artículo
```

De acuerdo a las instancias del esquema global presentadas en la tabla 4.4 el resultado que arroja esta consulta global es:

Autor.Nombre_autor	Arts.Nombre_Artículo	Arts.Paginas
Cheng Hian Goh	Languages for multi-database interoperability	Null

Para lograr que esta operación se ejecute de manera transparente en las bases de datos locales y obtenga el resultado anteriormente presentado se tiene que traducir esta operación en operaciones SQL locales y se requiere hacer mapeos del esquema global con cada una de las bases de datos locales. Dichos mapeos requieren el conocimiento de las correspondencias semánticas entre el esquema global y los esquemas de las bases de datos locales.

En la figura 4.5 se puede notar que para hacer los mapeos es necesario que se conozcan los siguientes tipos de correspondencias semánticas:

1. *Correspondencias semánticas entre atributos:* Después de la cláusula `SELECT` se encuentran los atributos del esquema global que deben ser consultados. Pero no se conocen sus atributos semánticamente equivalentes en las bases de datos locales.
2. *Correspondencias semánticas entre tablas:* Porque después de la cláusula `FROM` se encuentran las tablas del esquema global sobre las que se realizará la consulta. Pero no se conocen sus tablas semánticamente equivalentes en las bases de datos locales.

3. *Correspondencias semánticas entre datos:* En la cláusula WHERE se encuentra una restricción a nivel dato, es decir, sólo se debe presentar en el resultado global los registros cuyo precio sea mayor a nueve dólares. En este caso se debe transformar el dato "cinco" de la unidad de medida "dolar" a la unidad de medida "peso".

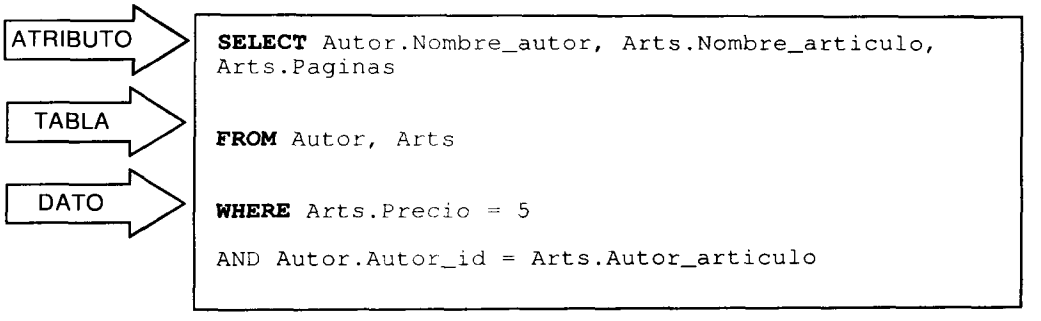

Figura 4.5: Tipos de correspondencias semánticas.

4.5.2 Registro de correspondencias semánticas

Es posible hacer el registro de correspondencias semánticas de dos formas:

1. ***Registrar las correspondencias semánticas sin crear tablas y atributos globales***
2. ***Registrar las correspondencias semánticas creando tablas y atributos globales***

A continuación se presenta la descripción de estas dos formas de registro.

4.5.2.1 Registrar las correspondencias semánticas sin crear tablas y atributos globales

Esta manera de registrar las correspondencias toma como entrada un conjunto de bases de datos y da como resultado una *red de correspondencias semánticas entre* las bases de datos, donde cada atributo y tabla del conjunto de bases de datos puede tener correspondencias semánticas con sus atributos equivalentes y tablas semánticamente equivalentes en el conjunto de bases de datos a integrar.

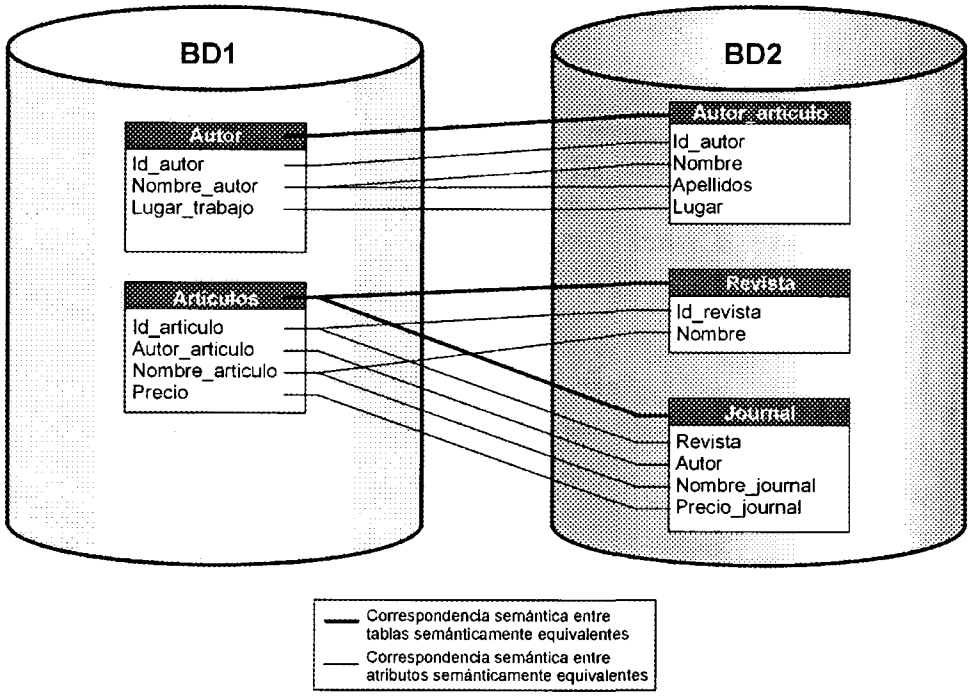

Figura 4.6: Registro de correspondencias semánticas sin crear tablas y atributos globales.

En la figura 4.6 se muestra la forma en la que se realiza este registro, en la cual se puede notar que existen dos tipos de correspondencias semánticas:

a) *Correspondencia semántica entre tablas semánticamente equivalentes:* en donde el registro de la correspondencia semántica se realiza de esta manera:
   i. BD1.Autor = BD2.Autor_articulo
   ii. BD1.Articulos = BD2.Revista
   iii. BD1.Articulos = BD2.Journal

b) *Correspondencia semántica entre atributos semánticamente equivalentes:* en donde el registro de las correspondencias semánticas se realiza de la siguiente manera:
   i. BD1Autor.Id_autor = BD2.Autor_artículo.Id_autor
   ii. BD1.Autor.Nombre_autor = BD2.Autor_articulo.Nombre
   iii. BD1.Autor.Lugar_trabajo = BD2.Autor_articulo.Lugar

Si este tipo de registro de correspondencias semánticas se realiza entre un número *n* bases de datos, se requiere que se registren correspondencias con los atributos de las *n* base de datos, lo cual es una desventaja por el tiempo y trabajo de registro. También se vuelve muy complicado y lento agregar nuevas bases de datos a este registro de correspondencias.

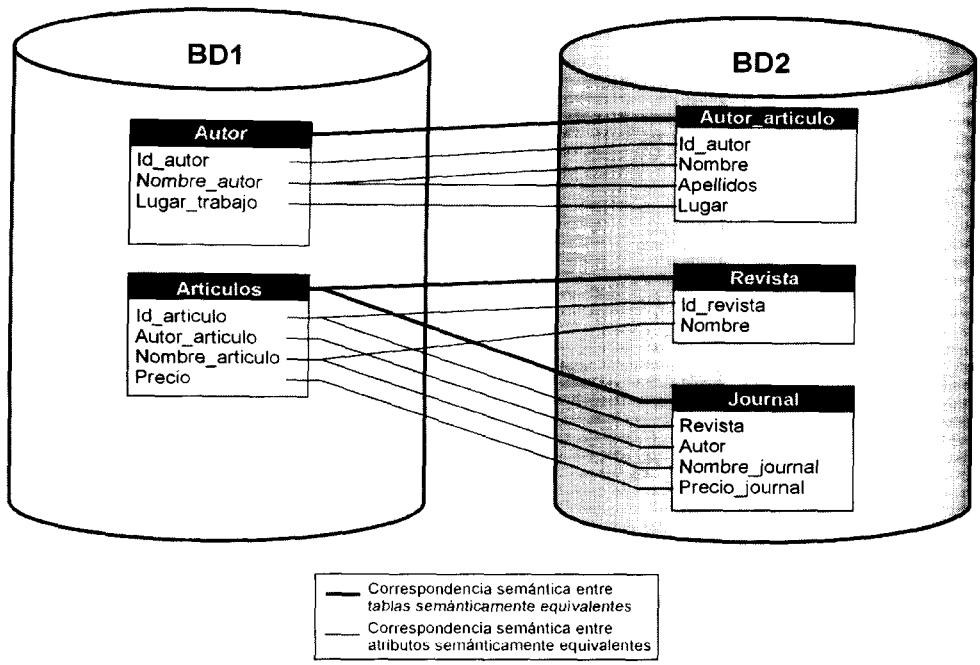

Figura 4.6: Registro de correspondencias semánticas sin crear tablas y atributos globales.

En la figura 4.6 se muestra la forma en la que se realiza este registro, en la cual se puede notar que existen dos tipos de correspondencias semánticas:

a) *Correspondencia semántica entre tablas semánticamente equivalentes:* en donde el registro de la correspondencia semántica se realiza de esta manera:
   i. BD1.Autor = BD2.Autor_articulo
   ii. BD1.Articulos = BD2.Revista
   iii. BD1.Articulos = BD2.Journal

b) *Correspondencia semántica entre atributos semánticamente equivalentes:* en donde el registro de las correspondencias semánticas se realiza de la siguiente manera:
   i. BD1Autor.Id_autor = BD2.Autor_artículo.Id_autor
   ii. BD1.Autor.Nombre_autor = BD2.Autor_articulo.Nombre
   iii. BD1.Autor.Lugar_trabajo = BD2.Autor_articulo.Lugar

Si este tipo de registro de correspondencias semánticas se realiza entre un número *n* bases de datos, se requiere que se registren correspondencias con los atributos de las *n* base de datos, lo cual es una desventaja por el tiempo y trabajo de registro. También se vuelve muy complicado y lento agregar nuevas bases de datos a este registro de correspondencias.

4.5.2.2 Registrar las correspondencias semánticas creando tablas y atributos globales

Esta forma de registro de las correspondencias semánticas tiene por objetivo tomar como entrada un conjunto de bases de datos locales y crear las tablas y atributos globales de acuerdo a las correspondencias encontradas. En la figura 4.7 se muestra este tipo de registro en la cual se puede notar que el resultado de esta actividad es un esquema global, ya que contiene las tablas y atributos que son semánticamente equivalentes a las tablas y atributos locales. A diferencia del registro sin crear tablas y atributos globales, donde se crean correspondencias entre un número *n* bases de datos, aquí se crean las correspondencias semánticas entre dos entidades: el esquema global y la base de datos local. Sin embargo, antes de crear las correspondencias se debe crear la tabla o atributo global con la cual se hace la correspondencia.

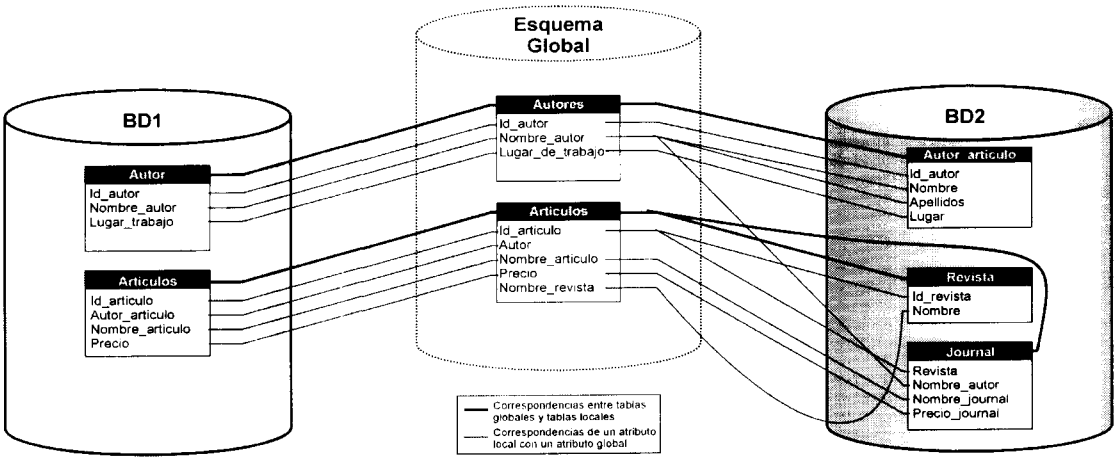

Figura 4.7: Registro de correspondencias semánticas creando tablas y atributos globales.

En la figura 4.7 se puede notar que existen dos tipos de correspondencias semánticas:

a) *Correspondencias entre tablas locales y tablas globales:* En este tipo de correspondencia se realiza una asociación entre una tabla del esquema global y una tabla de la base de datos local. La condición para que se pueda crear esa correspondencia es que la tabla global exista, si no existe se debe crear. El nombre que se asigna a la tabla global es decidido por la persona que realiza el registro. Por ejemplo: el registro de este tipo de correspondencias es de la siguiente manera:

   i. Autor = Autores
   ii. Articulos = Articulos
   iii. Autor_articulo =Autores
   iv. Revista = Articulos
   v. Journal = Articulos

No es necesario especificar el nombre de la base de datos, ya que la asociación siempre es entre una base de datos local con el esquema global. Con el fin de distinguir tablas con mismos nombres en diferentes bases de datos, es posible agregar el nombre de la base de datos local como prefijo, por ejemplo: BD1.Autor.

En el primer registro "Autor = Autores" se conoce que Autor (lado izquierdo de la igualdad) pertenece a la base de datos local y Autores (lado derecho de la igualdad) es una tabla del esquema global. Se puede notar que se crearon tablas globales llamadas "Autores" y "Artículos".

b) *Correspondencias de un atributo local con un atributo global:* En este tipo de correspondencia se establece una asociación entre el atributo de la base de datos local con el atributo del esquema global, la condición para que se pueda crear esa correspondencia es que el atributo global exista, si no existe se debe crear. Para crear el nombre del atributo global, se debe identificar la tabla global a la que pertenecerá ese atributo, y se debe asignar un nombre al atributo global dicho nombre es decisión de la persona que hace el registro de las correspondencias. De igual forma que en el registro de correspondencias entre tablas, la parte izquierda de la igualdad son las tablas y atributos de la base de datos local y el lado derecho son las tablas y atributos globales. Por ejemplo: el registro de este tipo de correspondencia es de la siguiente manera:

   i. Autor.Id_autor = Autores.Id_autor
   ii. Autor.Nombre_autor = Autores.Nombre_autor
   iii. Autor.Lugar_trabajo = Autores.Lugar_de_trabajo
   iv. Autor_articulo.Id_autor = Autores.Id_autor
   v. Autor_articulo.Nombre = Autores.Nombre_autor
   vi. Autor_articulo.Apellidos = Autores.Nombre_autor
   vii. Autor_articulo.Lugar = Autores.Lugar_de_trabajo

Aunque los registros son entre el esquema global y la base de datos local, es posible conocer las correspondencias semánticas, no sólo entre una base de datos local y el esquema global, sino también entre las bases de datos locales.

A continuación se presentan unos ejemplos que muestran la manera en la que es posible hacer esto.

1. Si se conoce que dos atributos en dos bases de datos locales contienen una correspondencia semántica con el mismo atributo global:
   a. BD1: *Autor.Lugar_trabajo = Autores.Lugar_de_trabajo*
   b. BD2: *Autor_artículo.Lugar = Autores.Lugar_de_trabajo*

   Entonces también existe una correspondencia semántica entre esos dos atributos locales:
   c. *Autor.Lugar_trabajo = Autor_articulo.Lugar*

2. Si se conoce que en una base de datos BD1 existe una correspondencia semántica entre un atributo local y un atributo global:
   a. *Autor.Nombre_autor = Autores.Nombre_autor*

   Y en otra base de datos BD2 existe una correspondencia de dos atributos locales con el mismo atributo global:
   b. *Autor_articulo.Nombre = Autores.Nombre_autor*
   c. *Autor_articulo.Apellido = Autores.Nombre_autor*

   Entonces se puede deducir que el atributo local de BD1 es correspondiente a la concatenación de los dos atributos locales de BD2:
   d. *Autor.Nombre_autor = Autor_articulo.Nombre + Autor_articulo.Apellido*

Sin embargo, no es necesario registrar estas correspondencias porque se pueden obtener con el registro existente entre las bases de datos locales con el esquema global. De esta manera se simplifica el registro, y se tiene la ventaja de reducir el tiempo de registrar todas las correspondencias semánticas.

3. Si se conoce que en una base de datos BD1 existe una correspondencia semántica entre un atributo local y un atributo global:
   a. *Autor.Nombre_autor = Autores.Nombre_autor*

   Luego, en otra base de datos BD2 existe una correspondencia semántica entre una tabla local y una tabla global:
   b. *Journal = Articulos*

   Y en la misma tabla *Journal* de BD2 existe un atributo *Nombre_autor* cuyo atributo global correspondiente no se encuentra en su tabla global correspondiente *Articulos,* sino en la tabla global *Autores.* En este caso el registro se realiza de la siguiente manera:
   c. *Journal.Nombre_autor = Autores.Nombre_autor*

Con este ejemplo se puede notar que no necesariamente las tablas de atributos correspondientes tienen que ser también correspondientes.

4.6 Metodología para integrar esquemas locales en un esquema global

Actualmente existen metodologías para integrar esquemas locales a un esquema global. Estas metodologías se pueden clasificar en metodologías que se basan en *ontologías* para crear un esquema global [Bergamaschi 1999, Hakimpour y Geppert 2001], y las metodologías que utilizan las correspondencias entre los esquemas [Bright et. al 1994, Johannesson 1994, Parent y Spaccapietra 1998, Rishe et. al 2000], estas metodologías requieren las solución de conflictos para resolver la heterogeneidad semántica.

En este trabajo de tesis se propone utilizar las correspondencias semánticas encontradas en un conjunto de bases de datos locales para la creación de un esquema global, el cual también funciona como un diccionario de datos estandarizado. Para lograr este resultado es necesario por una parte analizar las semánticas entre las bases de datos locales y por otra parte encontrar las correspondencias semánticas entre estas bases de datos.

Para obtener las correspondencias semánticas entre el conjunto de bases de datos locales es necesaria una investigación de las estructuras de las bases de datos locales y de sus datos. En este capítulo se propone una metodología para crear un esquema global partiendo de las correspondencias semánticas encontradas en el conjunto de bases de datos a integrar. El objetivo de esta metodología es obtener el conjunto de tablas y atributos que formarán el esquema global, los cuales son los registros del diccionario de datos de la arquitectura para integrar bases de datos propuesta. Se habla más de dicha arquitectura en el siguiente capítulo.

Los pasos de esta metodología son los siguientes:

1. *Análisis de las tablas y atributos a integrar;*
2. *Análisis y registro de correspondencias semánticas a nivel tabla;*
3. *Análisis y registro de correspondencias semánticas a nivel atributo;*
4. *Análisis y registro de correspondencias semánticas a nivel de dato;*
    a. *Registro de diferentes tipos de datos;*
    b. *Registro de diferentes unidades de medida;*
    c. *Registro de diferentes expresiones.*

Cada uno de estos pasos es aplicado al caso de estudio propuesto en este capítulo, donde se muestra a detalle el análisis que se realizó en las bases de datos locales para obtener el esquema global presentado en la sección 4.4.

A continuación se describen cada uno de estos pasos.

4.6.1 Paso 1: Análisis de las tablas y atributos a integrar

Las tablas y atributos a integrar son un subconjunto de las tablas y atributos de los esquemas locales. Debido a la autonomía de las bases de datos locales, en la mayoría de los casos, las bases de datos locales no comparten toda la base de datos, sólo las partes que se requieren integrar. Por esta razón se deben registrar las tablas y atributos que cada base de datos compartirá.

En esta actividad se debe formar una vista por cada base de datos local. Cada esquema local esta formado por *m* tablas y cada tabla por *n* atributos. La vista local resultante representa las tablas y atributos que la base de datos local permite compartir. En muchos casos la vista local, es igual al esquema de la bases de datos local, esto sucede cuando la base de datos local comparte todas sus tablas y atributos. Para formar las vistas locales se deben tomar en cuenta los siguientes puntos:

A. *La base de datos comparte tablas que contienen más de un atributo:* En este caso, la vista local toma el nombre de la tabla del esquema local y los atributos que se compartirán. El resto de los atributos los ignora.
B. *La base de datos local contiene tablas que comparten sólo uno de sus atributos:* En este caso, la vista local contendrá la tabla local sólo con el atributo que se comparte.
C. *La base de datos local contiene tablas que no comparten ningún atributo:* En este caso, el esquema de la vista local no comparte la tabla local.

4.6.2 Paso 2: Análisis y registro de correspondencias semánticas a nivel tabla

Después de definir las tablas y atributos que se desean integrar es necesario analizar las correspondencias semánticas utilizadas en cada una de las bases de datos. Aunque no existe herramienta para analizar las semánticas de las tablas, es posible obtener dichas semánticas del diccionario de datos de las bases de datos locales. El diccionario de datos describe el contenido de las bases de datos a nivel, tabla atributo y datos. También es posible conocer el tipo de dato y el formato, con estos elementos es posible crear tablas globales que sean semánticamente equivalentes a las tablas locales. Por lo tanto, también es posible registrar las correspondencias semánticas entre las tablas locales y las tablas globales.

Para poder crear tablas globales es necesario conocer la semántica de las tablas locales, es decir las partes del "mundo real" que representan dichas tablas. Es posible conocer esta semántica o significado partiendo de las siguientes propiedades:

- *La descripción de la tabla en el diccionario de datos:* La descripción del contenido de la tabla proporciona la información necesaria para el registro de las semánticas.
- *El nombre de la tabla:* Si una tabla tiene por nombre CONFERENCIA, lo más probable es que almacene registros cuyos atributos describan a las conferencias. Sin embargo, esta propiedad de la tabla no siempre describe su significado. Por ejemplo: podría suceder que una tabla llamada CONFERENCIA almacene registros que en vez de describir a las conferencias, describan a los proceedings. Cuando no sea clara la parte "mundo real" que representa el nombre de la tabla es necesario analizar su contenido.
- *El contenido de la tabla*: Cuando no es suficiente el nombre para definir la semántica de una tabla es necesario analizar su contenido, lo cual es una actividad que requiere más tiempo, sin embargo, es más confiable definir el significado basándose en el contenido que por el nombre. Por ejemplo, después de analizar el contenido de una tabla llamada CONFERENCIA se encontró que sus atributos describían a proceedings.

Registro de las correspondencias semánticas entre tablas

Durante este registro se debe definir por lo menos una correspondencia de cada tabla de las bases de datos locales debe con una tabla del esquema global. En la figura 4.8 se presenta el método propuesto.

Como se puede notar en la figura 4.8, el método para el registro de las correspondencias semánticas entre tablas consiste en tomar como entrada una tabla global, analizar su semántica, verificar si existe una tabla global correspondiente a dicha tabla, en caso de no existir se crea una tabla global cuyo nombre es igual a la tabla local. Por ultimo, Se registra una correspondencia semántica entre la tabla local y la tabla global.

Como se mencionó en la sección 4.6.2, el registro de una correspondencia semántica indica que las tablas o atributos que participan en el registro son semánticamente equivalentes.

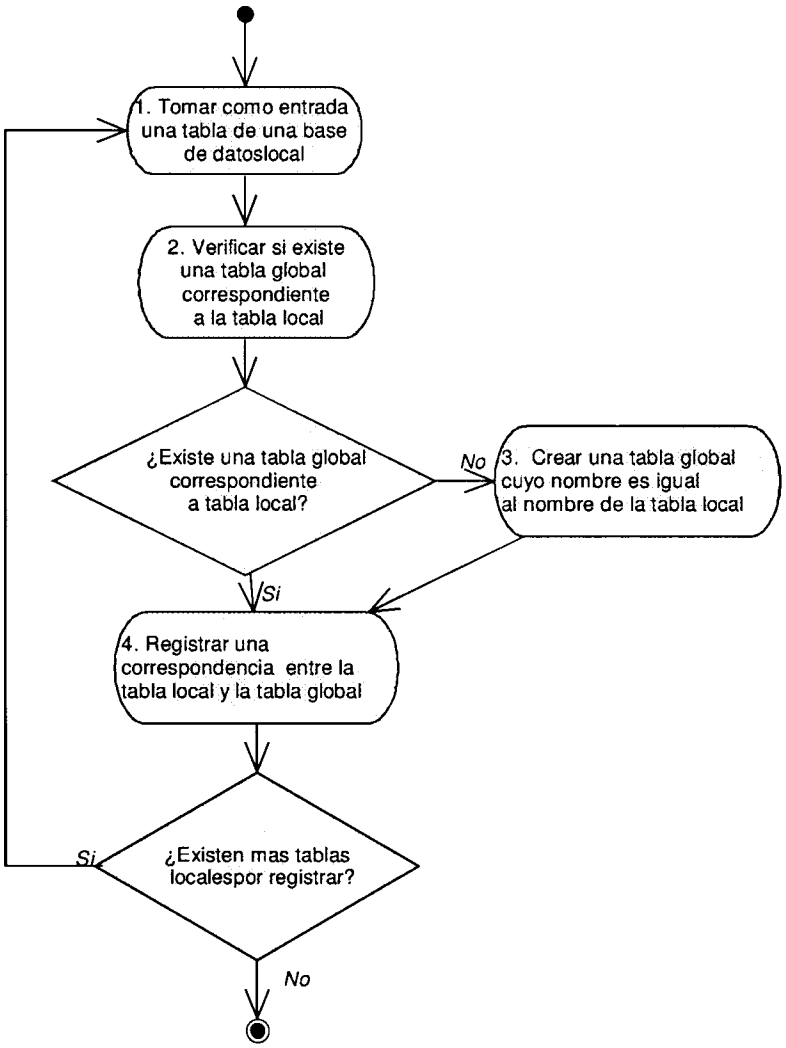

Figura 4.8: Método para el registro de correspondencias semánticas entre tablas.

En la tabla 4.5 se presenta el método para el registro de correspondencias semánticas entre tablas aplicado al caso de estudio. Primeramente se muestra el análisis de Biblioteca1, después de Biblioteca2 y por ultimo de Biblioteca3. Se puede notar que se crearon cinco tablas globales, las cuales son parte del esquema global: Autor, Arts, Journal, Proceeding_conf y conferencia. Las primeras cuatro tablas toman el nombre de las tablas locales de Biblitoeca1 y la última tabla toma el nombre de Biblioteca3.

	1. Tomar como entrada una tabla local	2. Verificar si existe una tabla global correspondiente	3. Crear una tabla global	4. Registrar una correspondencia semántica entre la tabla local y la tabla global	
				Tabla Local	Tabla Global
IBLIOTECA1	Autor	No	Autor	Autor	Autor
	Arts	No	Arts	Arts	Arts
	Journal	No	Journal	Journal	Journal
	Proceeding_conf	No	Proceeding_conf	Proceeding_conf	Proceeding_conf
IBLIOTECA2	Autores	Si	-	Autores	Autor
	Articulos_Conf	Si	-	Articulos_Conf	Arts
	Conferencia	No	Conferencia	Conferencia	Conferencia
IBLIOTECA3	Autores_Art	Si	-	Autores_Art	Autor
	Articulos_Journal	Si	-	Articulos_Journal	Arts
	Vol_journal	Si	-	Vol_journal	Journal

Tabla 4.5: Método para el registro de correspondencias semánticas entre tablas.

4.6.3 Paso 3: Análisis y registro de correspondencias semánticas a nivel atributo

Después de analizar y registrar las correspondencias semánticas a nivel tabla es necesario analizar y registrar las correspondencias semánticas a nivel atributo. De igual manera que en el análisis de semánticas a nivel tabla es necesario que se analicen los siguientes tres aspectos de los atributos:

- *La descripción del atributo en el diccionario de la base de datos local:* La descripción del atributo proporciona la información más importante para identificar la semántica de un atributo.
- *El nombre del atributo:* En la mayor parte de las veces, el nombre del atributo define su semántica, sin embargo existen excepciones. Por ejemplo, un atributo llamado NOMBRE_AUTOR, se espera que almacene el nombre del autor, sin embargo, esta no es regla general, porque pueden existir atributos que tengan ese nombre y almacenen otro tipo de información.
- *Las instancias de los atributos*: Cuando no sea clara la semántica que representa el nombre de un atributo, se pueden analizar sus instancias, las instancias de los atributos proporcionan información sobre la semántica. Por ejemplo, si un atributo

llamado LUGAR de una tabla AUTORES contiene una instancia "University of Arizona, Phoenix", se puede deducir que su semántica es "lugar de trabajo".

En la figura 4.9 se muestra el método propuesto para el registro de correspondencias semánticas a nivel de atributo. De la misma manera que en el método para el registro de correspondencias semánticas a nivel de tabla, primeramente se toma como entrada un atributo local, se analiza su semántica, después se verifica si existe un atributo global que sea semánticamente equivalente al atributo local, en caso de no existir el atributo global, se crea tomando el nombre del atributo local. Por último se realiza un registro de la correspondencia semántica entre el atributo local y el atributo global.

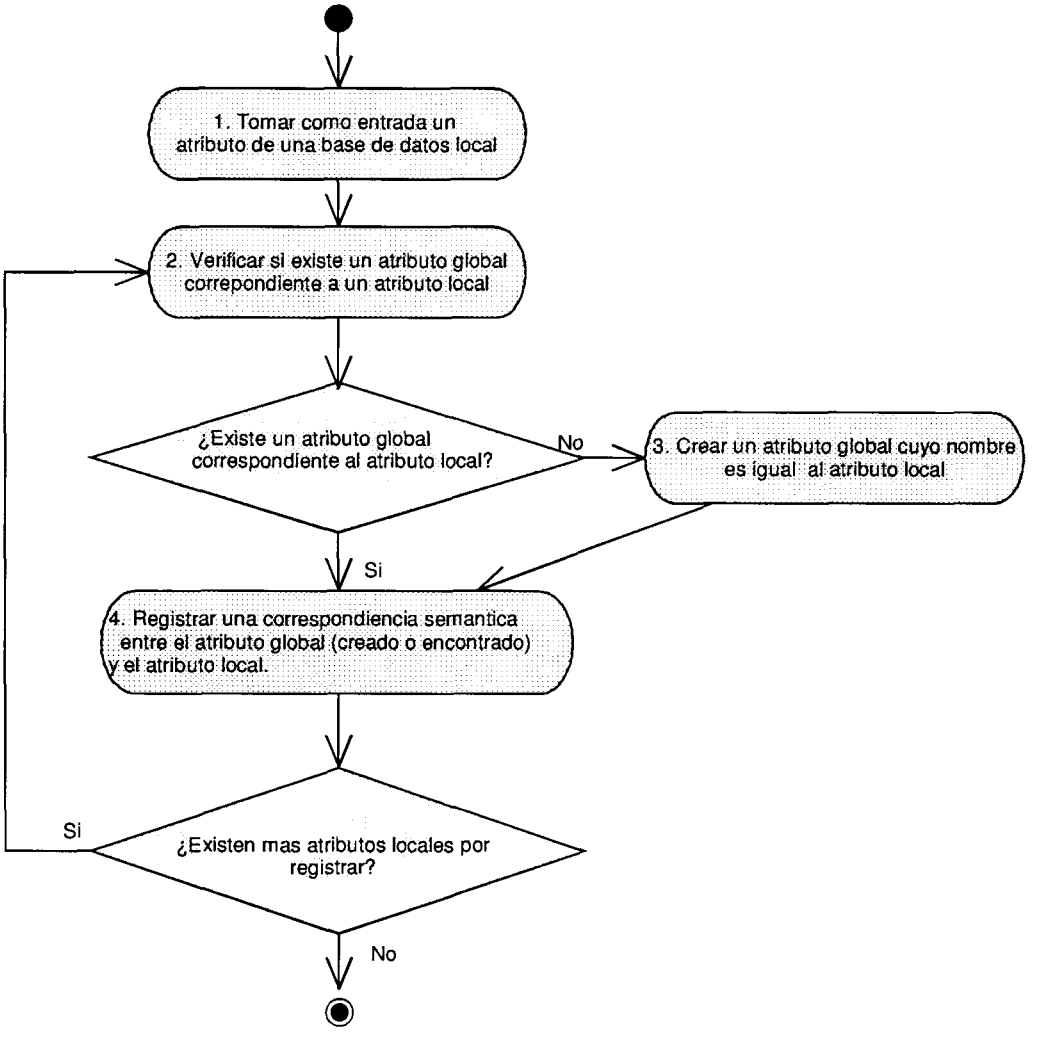

Figura 4.9: Método para el registro de correspondencias semánticas entre atributos.

	1. Tomar como entrada un Atributo Local	2. Verificar si existe un atributo global correspondiente	3. Crear un atributo global	4. Registrar una correspondencia semántica entre el atributo local y el atributo global	
				Atributo Local	Atributo global
BIBLIOTECA1	Autor.Autor_Id	No	Autor.Autor_Id	Autor.Autor_Id	Autor.Autor_Id
	Autor.Nombre_autor	No	Autor.Nombre_autor	Autor.Nombre_autor	Autor.Nombre_autor
	Autor.Lugar_trabajo	No	Autor.Lugar_trabajo	Autor.Lugar_trabajo	Autor.Lugar_trabajo
	Arts.Artículo_Id	No	Arts.Artículo_Id	Arts.Artículo_Id	Arts.Artículo_Id
	Arts.Keywords	No	Arts.Keywords	Arts.Keywords	Arts.Keywords
	Arts.Titulo	No	Arts.Titulo	Arts.Titulo	Arts.Titulo
	Arts.Año_pub	No	Arts.Año_pub	Arts.Año_pub	Arts.Año_pub
	Arts.Precio_art	No	Arts.Precio_art	Arts.Precio_art	Arts.Precio_art
	Arts.Autor_art	No	Arts.Autor_art	Arts.Autor_art	Arts.Autor_art
	Journal.Vol	No	Journal.Vol	Journal.Vol	Journal.Vol
	Journal.No	No	Journal.No	Journal.No	Journal.No
	Journal.ISSN	No	Journal.ISSN	Journal.ISSN	Journal.ISSN
	Journal.Revista	No	Journal.Revista	Journal.Revista	Journal.Revista
	Journal.Artículo	No	Journal.Artículo	Journal.Artículo	Journal.Artículo
	Proceeding_conf.Artículo	No	Proceeding_conf.Artículo	Proceeding_conf.Artículo	Proceeding_conf.Artículo
	Proceeding_conf.ISBN	No	Proceeding_conf.ISBN	Proceeding_conf.ISBN	Proceeding_conf.ISBN
	Proceeding_conf.Lugar	No	Proceeding_conf.Lugar	Proceeding_conf.Lugar	Proceeding_conf.Lugar
	Proceeding_conf.Nombre	No	Proceeding_conf.Nombre	Proceeding_conf.Nombre	Proceeding_conf.Nombre
BIBLIOTECA2	Autores.Clave	SI	-	Autores.Clave	Autor.Id
	Autores.Nombre	SI	-	Autores.Nombre	Autor.Nombre_autor
	Autores.Apellido	SI	-	Autores.Apellido	Autor.Nombre_autor
	Articulos_Conf.Id	SI	-	Articulos_Conf.Id	Arts.Artículo_Id
	Articulos_Conf.Titulo_art	SI	-	Articulos_Conf.Titulo_art	Arts.Titulo
	Articulos_Conf.Precio	SI	-	Articulos_Conf.Precio	Arts.Precio_art
	Articulos_Conf.Clave_autor	SI	-	Articulos_Conf.Clave_autor	Arts.Autor_art
	Conferencia.Nombre	No	Conferencia.Nombre	Conferencia.Nombre	Conferencia.Nombre
	Conferencia.Año	No	Conferencia.Año	Conferencia.Año	Conferencia.Año
	Conferencia.Lugar	No	Conferencia.Lugar	Conferencia.Lugar	Conferencia.Lugar
	Conferencia.ISBN	No	Conferencia.ISBN	Conferencia.ISBN	Conferencia.ISBN
BIBLIOTECA3	Autores_art.Id	Si	-	Autores_art.Id	Autor.Autor_Id
	Autores_art.Nombre_autor	Si	-	Autores_art.Nombre_autor	Autor.Nombre_autor
	Autores_art.Lugar	Si	-	Autores_art.Lugar	Autor.Lugar_trabajo
	Articulos_journal.Id	Si	-	Articulos_journal.Id	Arts.Artículo_Id
	Articulos_journal.Nombre_art	Si	-	Articulos_journal.Nombre_art	Arts.Titulo
	Articulos_journal.Año	Si	-	Articulos_journal.Año	Arts.Año_pub
	Articulos_journal.Autor_id	Si	-	Articulos_journal.Autor_id	Arts.Autor_art
	Articulos_journal.Paginas	No	Arts.Paginas	Articulos_journal.Paginas	Arts.Paginas
	Articulos_journal.Vol_journal	No	Arts.Vol_journal	Articulos_journal.Vol_journal	Arts.Vol_journal
	Vol_journal.Id	Si	-	Vol_journal.Id	Journal.Vol
	Vol_journal.Número	Si	-	Vol_journal.Número	Journal.No
	Vol_journal.ISSN	Si	-	Vol_journal.ISSN	Journal.ISSN
	Vol_journal.Nombre_rev	SI	-	Vol_journal.Nombre_rev	Journal.Revista

Tabla 4.6: Método para el registro de correspondencias semánticas a nivel atributo.

Como se puede notar en la tabla 4.6 el resultado de este análisis son los atributos del esquema global del caso de estudio propuesto en este capítulo. Con estos atributos y las tablas identificadas en la sección 4.6.3 se forma el esquema global. Se puede notar que todos los nombres de los atributos de la base de datos Biblioteca1 son iguales a los del esquema global, esto es porque Biblioteca1 es la primera base de datos analizada y debido a que sus atributos globales correspondientes no existían, fueron creados. Esto no sucede en las bases de datos Biblioteca2 y Biblioteca3 donde la mayoría de sus atributos ya tenían un atributo global correspondiente.

4.6.4 Paso 4: Análisis y registro de correspondencias semánticas a nivel de dato

Aun después de haber registrado las correspondencias semánticas a nivel atributo y tabla, existen diferencias en cuanto al formato de los datos que también deben ser registradas. Dichas diferencias son las siguientes:

1. Diferentes tipos de datos
2. Diferentes unidades
3. Diferentes expresiones

4.6.4.1 Diferentes tipos de datos

En la figura 4.10 se presenta el método para el registro de diferentes tipos de datos.

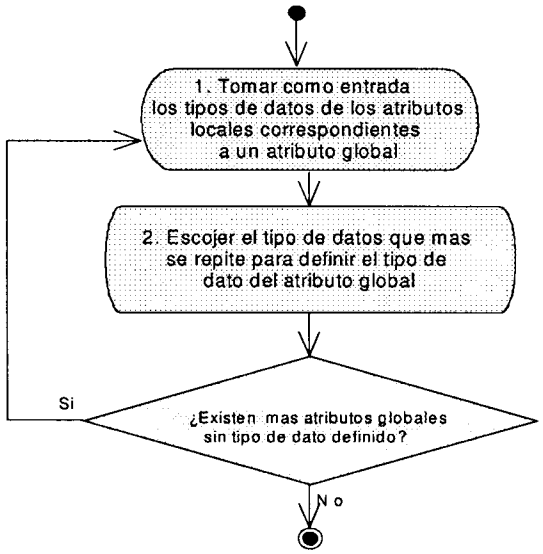

Figura 4.10: Método para el registro de diferentes tipos de datos.

En este paso se requiere que se registren los tipos de datos de todas las bases de datos y además se definan los tipos de datos para las tablas y atributos globales. Este

registro es necesario porque se debe manejar un tipo de dato global, al cual los atributos de las bases de datos locales deben mapearse.

En este paso se toma como entrada los tipos de datos de los atributos locales correspondientes a un atributo global y se escoge como el tipo de dato del atributo global el que más se repite.

De igual manera que en los pasos de la metodología presentados anteriormente, este método ha sido aplicado al caso de estudio propuesto en este capítulo. En la tabla 4.7 se presenta el análisis realizado. Como se puede notar en esta tabla las bases de datos utilizan tres tipos de datos: STRING, INTEGER y FLOAT. La mayor parte de los atributos locales utilizan el tipo de dato STRING. El tipo de dato INTEGER es utilizado solamente por los atributos AÑO_PUB, VOL, NO. El tipo de dato FLOAT solamente es utilizado por el atributo PRECIO_ART y su atributo semánticamente equivalente PRECIO.

En la tabla 4.7 también se puede notar que los tipos de datos que resultaron en el esquema global en su mayoría son de tipo STRING. El tipo de dato INTEGER en VOL y NO. El tipo de dato FLOAT solamente se definió en el atributo PRECIO_ART.

Las tres bases de datos coinciden en los tipos de datos en la mayor parte de sus atributos semánticamente equivalentes a excepción del atributo AÑO_PUB de BIBLIOTECA1 y AÑO de BIBLIOTECA3, en donde AÑO_PUB es INTEGER y AÑO es STRING. Como en este caso hay empate, el tipo de dato seleccionado para el esquema global es de tipo STRING, pero también es posible que sea INTEGER, el usuario puede elegir el tipo de dato.

Se puede notar que sigue existiendo el conflicto de tipos de datos diferentes entre el esquema global y BIBLIOTECA1. En el siguiente capítulo se muestra la forma en la que este tipo de conflictos son solucionados mediante componentes de software.

1. Tomar como entrada los tipos de datos de los atributos locales correspondientes a un atributo global								2. Escoger el tipo de dato que más se repite para definir el tipo de dato del atributo global		
Biblioteca1			Biblioteca2			Biblioteca3		Esquema global		
Tabla y Atributo		Tipode dato	Tabla y atributo		Tipo De Dato	Tabla y Atributo	Tipo de dato	Tabla y atributo		Tipo de dato
Autor	**Autor_id**	STRING	Autores	**Clave**	STRING	Autores_Art **Id**	STRING	Autor	**Autor_Id**	STRING
	Nombre_autor	STRING		Nombre	STRING	Nombre_autor	STRING		Nombre_autor	STRING
	Lugar_trabajo	STRING		Apellido	STRING	Lugar	STRING		Lugar_trabajo	STRING
Arts	**Artículo_Id**	STRING	Articulos_Conf	**Id**	STRING	Articulos_Journal **Id**	STRING	Arts	**Artículo_Id**	STRING
	Keywords	STRING							Keywords	STRING
	Titulo	STRING		Titulo_art	STRING	Nombre_art	STRING		Titulo	STRING
	Año_pub	INTEGER				Año	STRING		Año_pub	STRING
	Precio_art	FLOAT		Precio	FLOAT				Precio_art	FLOAT
	Autor_art	STRING		**Clave_autor**	STRING	**Autor_id**	STRING		**Autor_art**	STRING
						Paginas	STRING		Paginas	STRING
						Vol_journal	STRING		**Vol_journal**	STRING
Journal	**Vol**	INTEGER				Vol_journal **Id**	STRING	Journal	**Vol**	INTEGER
	No	INTEGER				Número	STRING		No	INTEGER
	ISSN	STRING				ISSN	STRING		ISSN	STRING
	Revista	STRING				Nombre_rev	STRING		Revista	STRING
	Artículo	STRING							**Artículo**	STRING
Proceeding_conf	**Artículo**	STRING						Proceeding_conf	**Artículo**	STRING
	ISBN	STRING							ISBN	STRING
	Lugar	STRING							Lugar	STRING
	Nombre	STRING							Nombre	STRING
			Conferencia	Nombre	STRING			Conferencia	Nombre	STRING
				Año	STRING				Año	STRING
				Lugar	STRING				Lugar	STRING
				ISBN	STRING				ISBN	STRING

Tabla 4.7: Método para el registro de diferentes tipos de datos.

4.6.4.2 Diferentes Unidades

En la figura 4.11 se muestra el método para el registro de diferentes unidades propuesto.

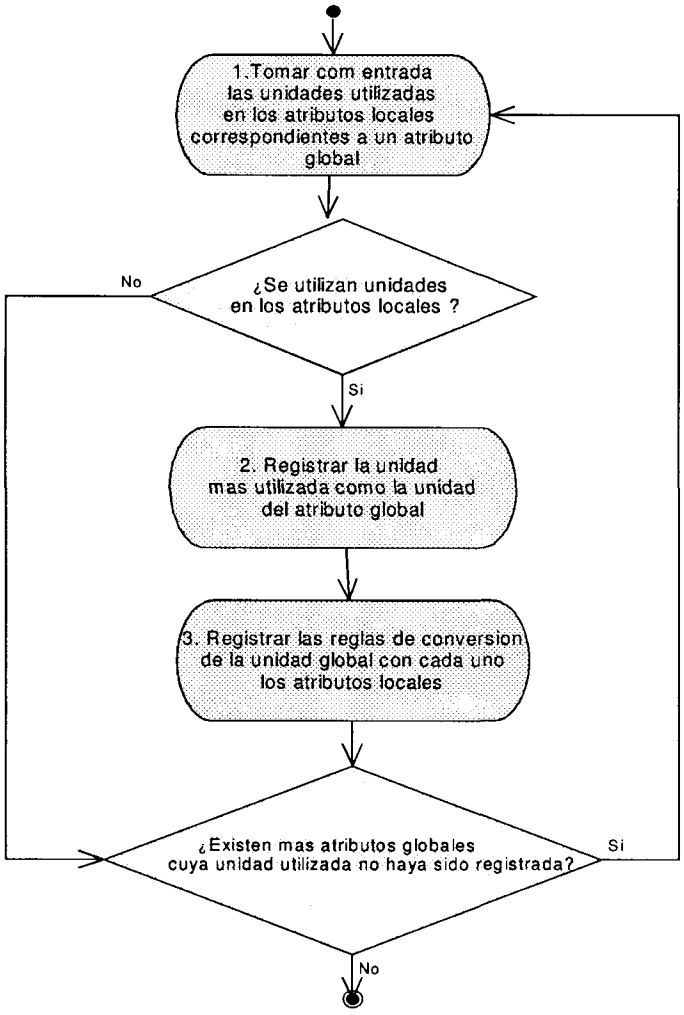

Figura 4.11: Método para el registro de diferentes unidades.

Para hacer el registro de diferentes unidades, es necesario por una parte registrar las unidades utilizadas en los atributos locales, la unidad definida en los atributos globales y por otra parte registrar la regla de conversión entre la unidad del atributo local y la unidad del atributo global. Este registro es necesario porque se debe mostrar al usuario global una única unidad, la cual se debe elegir del conjunto de unidades utilizadas en las bases de

datos locales. Después de haber elegido dicha unidad, se debe hacer el registro de la regla de conversión en los atributos locales, cuando la unidad utilizada sea diferente a la unidad global. Para identificar las unidades utilizadas es necesario analizar el diccionario de datos de las bases de datos locales.

En este registro se toma como entrada los atributos locales correspondientes a un atributo global, se verifica si se utilizan unidades en los atributos locales, se registra la unidad más utilizada como la unidad del atributo global y por último se registra la regla de conversión entre los atributos locales. La regla de conversión es necesaria porque aun cuando se conozca la unidad utilizada en los atributos locales y globales, se requiere tener la información para hacer las conversiones entre dichas unidades.

En las tablas 4.8 y 4.9 se presenta el método para el registro de diferentes unidades. En la tabla 4.8 (tabla A), se encuentran los pasos 1 y 2, los cuales consisten en tomar como entrada los atributos locales correspondientes a un atributo local y registrar la unidad más utilizada por los atributos locales. En la tabla 4.9 (tabla B), se encuentra el paso 3, el cual consiste en registrar las reglas de conversión entre las unidades.

En las tablas 4.8 y 4.9 se puede notar que en este caso de estudio sólo existen dos campos semánticamente equivalentes que utilizan unidades, los cuales son PRECIO_ART en BIBLIOTECA1 y PRECIO en BIBLIOTECA2. Las unidades utilizadas por estos dos campos son "Peso" en PRECIO_ART y "Dólar" en el campo PRECIO. Para el atributo global que es correspondiente a estos dos campos se seleccionó la unidad "Dólar".

En la tabla 4.9 se encuentran las reglas de conversión entre la unidad "Peso" y "Dolar". Existe una regla por cada atributo local. En BIBLIOTECA1 la regla tiene por objetivo obtener la cantidad en dólares, tomando como entrada el dato en pesos. En BIBLIOTECA2 la regla de conversión indica que tanto el atributo local como el global utilizan la misma unidad. En caso que el tipo de cambio de dólares a pesos cambie, también se debe cambiar la regla. En esta tesis se tiene la limitante de que el esquema global y sus reglas de conversión no se actualizan de acuerdo a los cambios. Esta limitante se tiene como una de las líneas de trabajos futuros.

1. Tomar como entrada los atributos locales correspondientes a un atributo global										2. Registrar la unidad más utilizada por los atributos locales		
Biblioteca1			Biblioteca2			Biblioteca3				Esquema global		
Tabla y Atributo		Unidad	Tabla y atributo		Unidad	Tabla y atributo		Unidad		Tabla y atributo		Unidad
Autor	**Autor_id**		Autores	**Clave**		Autores_Art	**Id**			Autor	**Autor_Id**	
	Nombre_autor			Nombre			Nombre_autor				Nombre_autor	
				Apellido								
	Lugar_trabajo						Lugar				Lugar_trabajo	
Arts	**Artículo_Id**		Articulos_Conf	**Id**		Articulos_Journal	**Id**			Arts	**Artículo_Id**	
	Keywords										Keywords	
	Titulo			Titulo_art			Nombre_art				Titulo	
	Año_pub						Año				Año_pub	
	Precio_art	Peso		Precio	Dolar						Precio_art	Dolar
	Autor_art			**Clave_autor**			**Autor_id**				**Autor_art**	
							Paginas				Paginas	
							Vol_journal				**Vol_journal**	
Journal	**Vol**					Vol_journal	**Id**			Journal	**Vol**	
	No						Número				No	
	ISSN						ISSN				ISSN	
	Revista						Nombre_rev				Revista	
	Artículo										**Artículo**	
Proceeding_conf	**Artículo**									Proceeding_conf	**Artículo**	
	ISBN										**ISBN**	
	Lugar										Lugar	
	Nombre										Nombre	
			Conferencia	Nombre						Conferencia	Nombre	
				Año							Año	
				Lugar							Lugar	
				ISBN							**ISBN**	

Tabla 4.8: Método propuesto para el registro de diferentes unidades. Tabla A.

3. Registrar las reglas de Conversión entre los atributos globales y los atributos locales				
Biblioteca1	Biblioteca2	Biblioteca3	Esquema global	
Regla de Conversión	Regla de Conversión	Regla de Conversión	Tabla y atributo	Unidad
Arts.Precio_art= Arts.Precio_art * 10	Articulos_conf.Precio = Arts.Precio_art	-	Arts.Precio_art	dolar

Tabla 4.9: Método para el registro de diferentes unidades. Tabla B.

4.6.4.3 Diferentes Expresiones

A nivel de datos, las bases de datos generalmente utilizan diferentes expresiones para almacenar sus datos. Las expresiones son las diferentes formas de registrar un mismo dato en atributos que son semánticamente equivalentes. Existen diferentes aspectos de las instancias de los atributos que pueden generar conflictos de diferentes expresiones, algunos de los cuales son los siguientes:

- Usar palabras diferentes que significan lo mismo (sinónimos), por ejemplo: "ubicación", "dirección".
- Usar abreviaturas, por ejemplo: "Texas", "Tx.".
- Usar diferentes lenguajes humanos para el dato, por ejemplo: "Street", "Calle"
- Usar diferente formato para los campos fecha, por ejemplo: "Sunday, Agust 08, 2004", "08-08-2004".

Una manera de dar solución a conflictos de diferentes expresiones, es generar un diccionario de sinónimos global. Este diccionario contiene las palabras que se utilizan en los datos "virtuales" del esquema global y sus sinónimos en las bases de datos locales. Una aplicación que se le puede dar al diccionario de sinónimos es la traducción de consultas, por ejemplo: Supóngase que se tienen dos bases de datos locales BD1 y BD2, y un esquema global, y un usuario quisiera conocer los autores cuya ciudad de procedencia sea D.F., se ejecutaría la siguiente consulta global:

```
SELECT  Autores.Nombre_autor
FROM Autores
WHERE Autores.Lugar_trabajo LIKE "% D.F.%"
```

Al traducir esta consulta en las bases de datos locales, se buscan los sinónimos de la palabra después del operador LIKE, y después se usan dichos sinónimos para hacer la transformación de la consulta. De esta manera, la traducción de la cláusula WHERE queda de la siguiente manera:

- BD1:
  ```
  WHERE Autores.Lugar_trabajo LIKE "%D.F.%"
  ```
- BD2:
  ```
  WHERE Autores.Lugar LIKE "%Distrito Federal%"
  ```

Para formar un diccionario de sinónimos es posible seguir el método presentado en la figura 4.12. Se puede notar que se encuentra dividido en cuatro pasos. En el paso uno se toma como entrada los atributos locales correspondientes a un atributo global. Después, en el paso dos, en caso de existir palabras diferentes que signifiquen lo mismo, se selecciona una palabra para utilizarla en las instancias "virtuales" del esquema global, las instancias son virtuales porque no existen físicamente en el esquema global, pero es necesario que los usuarios de este esquema "sientan" que las instancias se encuentran en dicho esquema. En el paso tres, en caso de existir abreviaturas, se selecciona una abreviatura o palabra que se utilizará en las instancias del esquema global. Por ultimo, en el paso cuatro, en caso de existir palabras de diferentes lenguajes que signifiquen lo mismo, se selecciona una palabra para las instancias del esquema global.

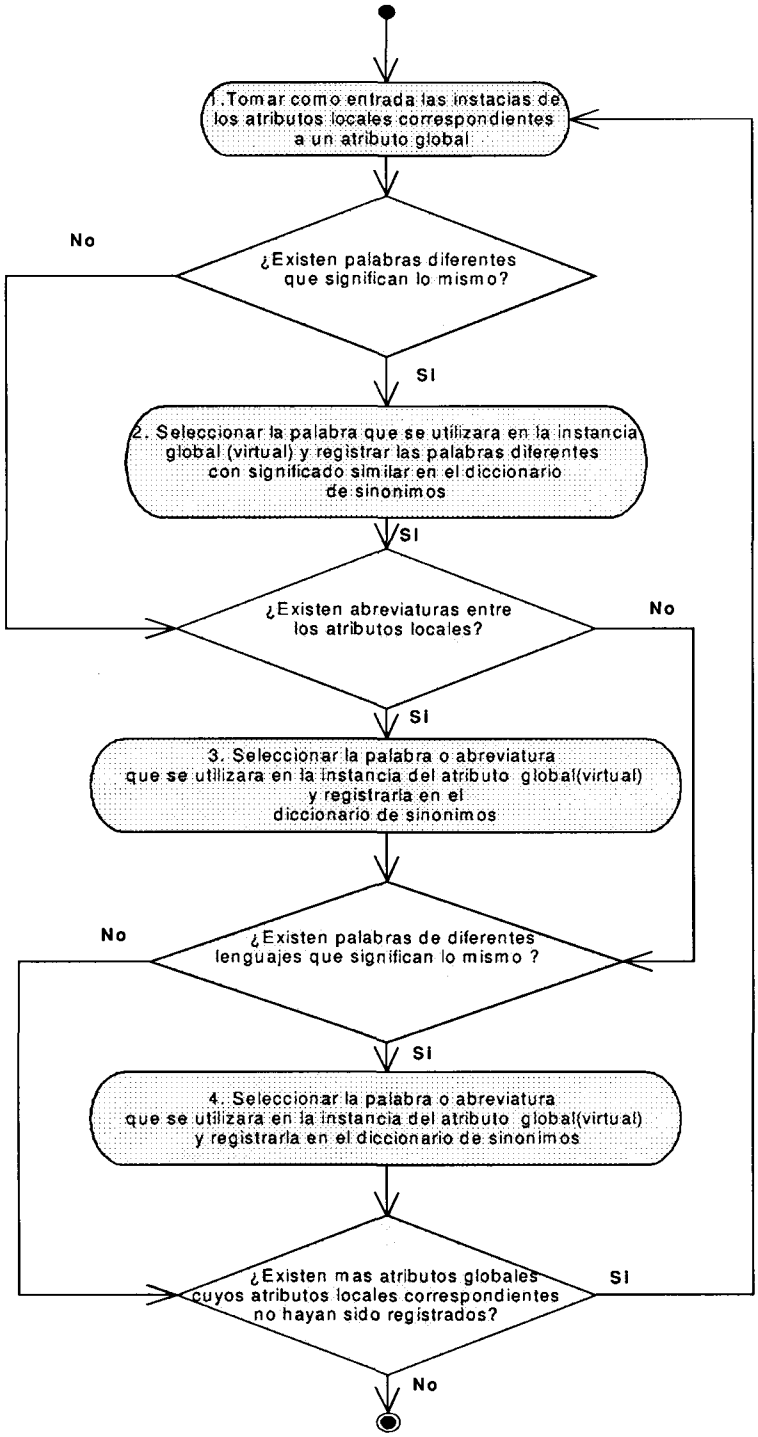

Figura 4.12: Método para la creación de un diccionario de sinónimos.

Al igual que en los pasos de la metodología para crear un esquema global propuesta en este capítulo, este paso también se ha aplicado al caso de estudio. Las tablas 4.10 (tabla

A), 4.11 (tabla B), 4.12 (tabla C), 4.13 (tabla D) presentan los resultados de esta actividad. La tabla 4.10 presenta los atributos locales que se tomaron como entrada, en donde se analiza si estos utilizan sinónimos, abreviaturas o palabras de diferentes lenguajes. Como se puede notar, solamente existen dos atributos que son semánticamente equivalentes utilizando abreviaturas, dichos atributos son LUGAR_TRABAJO en BIBLIOTECA1 y LUGAR en BIBLIOTECA2.

1. Tomar como entrada las instancias de los atributos locales correspondientes a un atributo global						**¿Existen palabras diferentes que significan lo mismo?**	**¿Existen abreviaturas en las instancias?**	**¿Existen palabras de diferentes lenguajes que significan lo mismo?**
Biblioteca1		**Biblioteca2**		**Biblioteca3**				
Tabla y Atributo		Tabla y atributo		Tabla y atributo				
Autor	**Autor_id**	**Autores**	**Clave**	**Autores_Art**	**Id**	No	No	No
	Nombre_autor		Nombre		Nombre_autor	No	No	No
			Apellido					
	Lugar_trabajo				Lugar	No	Si	No
Arts	**Artículo_Id**	**Artículos_Conf**	**Id**	**Artículos_Journal**	**Id**	No	No	No
	Keywords					No	No	No
	Titulo		Titulo_art		Nombre_art	No	No	No
	Año_pub				Año	No	No	No
	Precio_art		Precio			No	No	No
	Autor_art		**Clave_autor**		**Autor_id**	No	No	No
					Paginas	No	No	No
					Vol_journal	No	No	No
Journal	**Vol**			**Vol_journal**	**Id**	1No	No	No
	No				Número	No	No	No
	ISSN				ISSN	No	No	No
	Revista				Nombre_rev	No	No	No
	Artículo					No	No	No
Proceeding_conf	**Artículo**					No	No	No
	ISBN					No	No	No
	Lugar					No	No	No
	Nombre					No	No	No
		Conferenc	Año			No	No	No
			Lugar			No	No	No
			ISBN			No	No	No

Tabla 4.10: Método para la creación de un diccionario de sinónimos. Tabla A.

2. Seleccionar la palabra que se utilizaran en la instancia global (virtual) y sus sinónimos en las instancias de los atributos locales				
Biblioteca11	Biblioteca2	Biblioteca3	Esquema global	
			Tabla y atributo	Palabra
Palabra	Palabra	Palabra		

Tabla 4.11: Método para la creación de un diccionario de sinónimos. Tabla B.

3. Seleccionar la abreviatura o palabra que se utilizara en la instancia global (virtual) y sus equivalentes en las instancias de los atributos locales				
Biblioteca1	Biblioteca2	Biblioteca3	Esquema global	
			Tabla y atributo	Palabra
Palabra	Palabra	Palabra		
AZ		Arizona	Autor.Lugar	Arizona
CA		California	Autor.Lugar	California

Tabla 4.12: Método para la creación de un diccionario de sinónimos. Tabla C.

4. Seleccionar las palabras que se utilizaran en la instancia global (virtual) y sus equivalentes en las instancias de los atributos locales				
Biblioteca1	Biblioteca2	Biblioteca3	Esquema global	
Palabra	Palabra	Palabra	Tabla y atributo	Palabra

Tabla 4.13: Método para la creación de un diccionario de sinónimos. Tabla D.

Las tablas 4.11, 4.12 y 4.13 muestran las palabras registradas en el diccionario de sinónimos. La tablas 4.11 y 4.13 no contienen ningún dato porque en este caso no existen palabras diferentes que signifiquen lo mismo, ni tampoco palabras de diferentes lenguajes. Sin embargo, la tabla 4.12 muestra que se identificaron cuatro palabras que deben ser registradas en el diccionario de sinónimos, las cuales son "AZ" que es semánticamente equivalente a "Arizona", "CA" que es semánticamente equivalente a "California "

Aunque en esta tesis no se habla acerca de otros problemas relacionados con los datos de las bases de datos, existen más conflictos que deben ser abordados, por ejemplo datos obsoletos, datos incorrectos, diferentes formatos, etc. Constantemente surgen nuevas tecnologías de bases de datos como DBMS´s, middleware de bases de datos, los cuales pueden generar otros tipos de conflictos. Es posible crear nuevas reglas de conversión para conflictos más específicos, en el siguiente capítulo se muestra cómo es posible hacer eso.

4.7 Conclusión

En este capítulo se ha presentado una introducción a la integración semántica de datos, así como las opciones de integración existentes. Para ilustrar la problemática se ha presentado un caso de estudio para un sistema bibliográfico. Partiendo de este caso, se ha presentado una metodología para integrar esquemas de bases de datos en un esquema global, esta metodología se basa en la identificación de correspondencias semánticas entre los esquemas de las bases de datos locales. Con esta información semántica se ha construido un esquema global para el sistema bibliográfico, el cual contiene correspondencias semánticas para cada una de las tablas y atributos de los esquemas locales. Con el uso de dicho esquema es posible reducir los conflictos entre n-esquemas a conflictos entre 2-esquemas: el esquema global y los esquemas locales. De la construcción de este esquema global se pude concluir que la identificación de correspondencias es un excelente punto de partida para la creación de un esquema global e integración de bases de datos porque dicho esquema contiene la información sobre las correspondencias semánticas y proporciona una vista única. Esta información semántica se puede utilizar para la transformación de consultas y conciliación de resultados de operaciones SQL globales. En el siguiente capítulo se presenta la manera en la que se puede utilizar la información semántica proporcionada por el esquema global mediante una arquitectura basada en componentes de software.

Capítulo 5. Arquitectura del prototipo

En el capítulo 4 se presentó una metodología para la integración de esquemas locales en un esquema global, la cual se basa en la identificación de correspondencias semánticas a nivel tabla, atributo y datos. Dichas correspondencias ayudan en la solución de la heterogeneidad semántica porque los conflictos entre n-esquemas son reducidos a conflictos entre sólo 2-esquemas: el esquema global y un esquema local. Sin embargo, aun después de haber construido un esquema global faltan algunas actividades para lograr la integración de bases de datos. Dichas actividades consisten en implementar una arquitectura de componentes de software que extraigan la información semántica de este esquema y la usen para la solución de conflictos entre las bases de datos locales. Para solucionar dichos conflictos esta arquitectura permite registrar el esquema global y las correspondencias semánticas identificadas en el capítulo anterior en un diccionario de datos global (GDD). Con la información semántica registrada en el GDD es posible extraerla y utilizarla mediante componentes de software para transformar las operaciones SQL globales en operaciones SQL locales.

En este capítulo se presenta la descripción de la arquitectura para integrar bases de datos propuesta en este trabajo de tesis. Ésta se enfoca en la solución de la heterogeneidad semántica y está compuesta de componentes de software y un diccionario de datos global. El diccionario de datos global contiene el esquema global y las correspondencias semánticas entre el esquema global con cada uno de los esquemas de las bases de datos locales. Los componentes de software constan de un analizador semántico (GSA), el cual se encarga de extraer la información semántica. Reglas de transformación, las cuales son codificadas en el componente GTR. Un transformador de consultas (GQT), el cual es el componente central de la arquitectura y se encarga de realizar las transformaciones de consultas utilizando la información proporcionada por los componentes GSA y GTR, este componente también se encarga de enviar las consultas transformadas a los DBMS locales y conciliar los resultados. Además de estos componentes, la arquitectura aquí propuesta utiliza una API DML para la captura de las consultas globales.

Las secciones que se presentan en este capítulo son las siguientes: en la sección 1 se presentan los antecedentes de la integración semántica de datos, en la sección 2 se presentan los objetivos y límites de la arquitectura, en la sección 3 se presenta la descripción de la arquitectura desde dos enfoques: datos y componentes, en la sección 4 se describe la secuencia de operaciones entre los componentes de la arquitectura. La sección 5 presenta una descripción de los componentes de la arquitectura propuesta. En la sección 6 se presenta la arquitectura de implementación, la sección 7 presenta un método para la implementación de la arquitectura basada en componentes y en la sección 8 se presentan las conclusiones de este capítulo.

5.1 Integración semántica de datos

En el capítulo 4 ya se presentó la problemática de la integración semántica de datos y se presentó un caso de estudio donde se integran los esquemas de tres bases de datos heterogéneas en un esquema global. También se mencionó que el esquema global ayuda a solucionar la heterogeneidad semántica entre las bases de datos mediante el registro de las correspondencias semánticas entre el esquema global y las bases de datos locales. En este trabajo de tesis se propone utilizar el registro de estas correspondencias como un medio para la solución de la heterogeneidad entre las bases de datos locales. La información semántica registrada es accedida por los componentes de software de la arquitectura propuesta. De esta manera es posible que dichos componentes transformen las operaciones SQL globales en operaciones para las bases de datos locales y concilien los resultados heterogéneos en un resultado único.

5.2 Objetivos y límites de la arquitectura

Los principales objetivos de esta arquitectura son los siguientes:

1. *Proporcionar un acceso transparente a las bases de datos:* lo cual significa que los usuarios globales desconocen la ubicación y estructura de las bases de datos locales. El acceso a las bases de datos se logra a través del esquema global.
2. *Proporcionar una vista unificada del conjunto de bases de datos*: como si se tratara de un solo sistema de bases de datos, es decir, los usuarios globales sólo conocen la estructura del esquema global, el cual es una base de datos virtual, ya que no almacena registros de datos.
3. *Proporcionar capacidades para borrar o actualizar información de las bases de datos.*

Todos estos objetivos se deben cumplir respetando la autonomía de las bases de datos lo cual significa que los usuarios de las bases de datos locales no son afectados por el acceso de los usuarios globales.

La limitación que tiene la arquitectura es la falta de una solución para conflictos por diferentes expresiones, datos incorrectos u obsoletos de las bases de datos locales. Por ejemplo: diferentes notaciones para las direcciones, teléfonos o incluso nombres. La falta de solución para estos conflictos se debe principalmente a la falta de información precisa para formar reglas que describan las diferencias de las expresiones. Sin embargo la arquitectura aquí descrita implementa un componente llamado GTR, donde es posible capturar mediante código Java las diferencias entre los datos de las bases de datos locales.

5.3. Descripción de la arquitectura para integrar bases de datos

En esta sección se describe la arquitectura del prototipo de tesis. La descripción se divide en dos partes:

1. Arquitectura con un enfoque en datos: se puede apreciar las bases de datos y los esquemas que son utilizados.
2. Arquitectura enfocada en componentes: se puede apreciar los componentes de software que integran esta arquitectura y la función que realizan para solucionar la heterogeneidad semántica.

A continuación se describen estos dos enfoques de arquitecturas.

5.3.1 Arquitectura basada en datos

La arquitectura propuesta en esta tesis se encuentra basada en la arquitectura reportada en [Özsu y Valduriez, 1999] de Tsichristzis y Klug en 1978 la cual está compuesta de tres vistas: externa, conceptual e interna. La idea de esta arquitectura es presentar transparencia mediante la vista conceptual. La arquitectura propuesta en esta tesis presenta transparencia mediante un esquema global. Dicho esquema es una vista conceptual del conjunto de bases de datos a integrar. Las vistas internas son los esquemas de las bases de datos locales. Y las vistas externas son las partes del esquema global a las que los usuarios globales pueden acceder.

Tal como se muestra en la figura 5.1, los elementos de la arquitectura enfocada en datos aquí propuesta son los siguientes:

- **FDBMS** (*Federated DBMS*): En este servidor reside el esquema global cuya descripción se almacena en el diccionario de datos global.
- **GS** (*Global Schema*): Representa el conjunto de los esquemas de las bases de datos locales.
- **LDBMS-Server**(*Local DBMS*): Es el DBMS para una base de datos local.
- **LES** (*Local External Schema*): Es una vista del LCS, a la cual un usuario global puede acceder.
- **LCS** (*Local Conceptual Schema*): Es el esquema conceptual de una base de datos local.
- **LDB** (*Local Database):* Es una base de datos local.

5.3.2 Arquitectura basada en componentes

Debido a que la arquitectura basada en componentes está enfocada en la resolución de la heterogeneidad semántica de un conjunto de bases de datos locales, sus componentes, los cuales son ilustrados en la figura 5.2, trabajan en conjunto para extraer la información de semántica de las bases de datos, transformar las llamadas SQL, ejecutar las consultas en cada una de las bases de datos locales y conciliar los resultados en un resultado global.

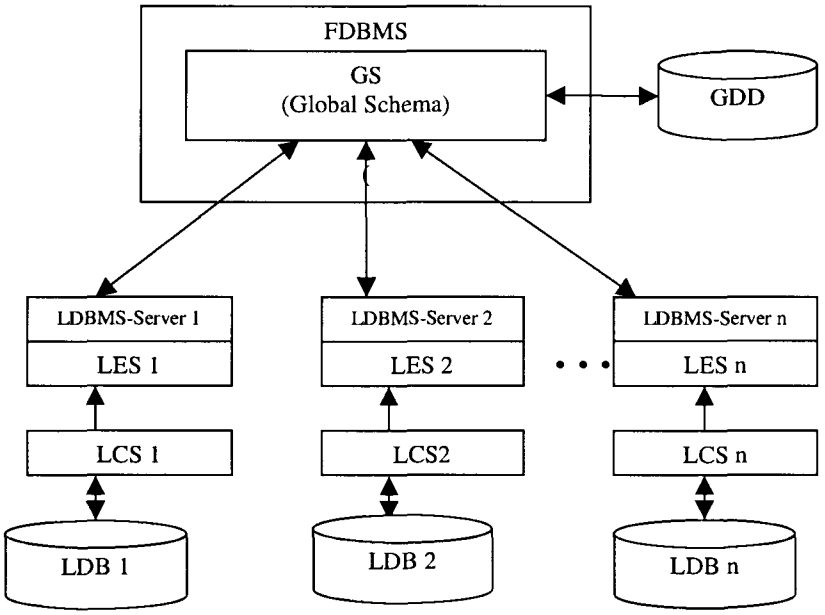

Figura 5.1: Arquitectura propuesta (basada en datos).

La arquitectura aquí propuesta está basada en la arquitectura basada en componentes desarrollada en [Álvarez 2003], de la cual se reutilizan dos componentes: la API de acceso a los datos y el componente LDBMS-Server. El trabajo de Álvarez propone una arquitectura genérica basada en componentes de la distribución de datos de un sistema de bases de datos distribuidas. A diferencia de la arquitectura de Álvarez, donde las bases de datos son creadas y distribuidas mediante sus componentes. Las bases de datos locales de la arquitectura aquí propuesta existen antes de la implementación y son heterogéneas. El trabajo de Álvarez se enfoca en la distribución de los datos mientras que este trabajo se enfoca en la solución de la heterogeneidad semántica. Sin embargo, es posible reutilizar el componente API DML de su arquitectura, ya que dicho componente es una capa de alto nivel, lo cual significa que no toma en cuenta la estructura de los esquemas de las bases de datos locales. Otro componente reutilizado del trabajo de Álvarez es el LDBMS-Server, el cual ejecuta las operaciones SQL en los DBMS locales y proporciona transparencia a nivel DBMS.

Los componentes de la arquitectura propuesta en esta tesis son los siguientes:

- **GDD** (*Global Data Diccionary)*: Este componente tiene doble función, la primera es describir el **GS** (*Global Schema*). El GS es el esquema global resultante de la metodología presentada en el capítulo 4. La segunda función del GDD es la descripción de las correspondencias semánticas entre el GS con cada una de las bases de datos locales.
- **API DML SQL**: Es la interfase SQL utilizada para tomar como entrada las operaciones SQL de los usuarios globales. Las operaciones soportadas son de tipo **DML** (*Data Manipulation Language*), las cuales sólo permiten manipular los datos. Las operaciones SQL **DDL**(*Data Definition Languaje*) no se utilizan porque son para cambiar las estructuras de las bases de datos y una de las restricciones al integrar las bases de datos es respetar la autonomía de las bases de datos locales, lo cual significa que éstas no deben ser cambiadas en cuanto a su estructura para no afectar la operación de los usuarios locales.
- **GSA** (*Global Semantic Analizer*): Este componente tiene la función de extraer la información semántica requerida por el GQT para hacer las transformaciones de las operaciones SQL realizadas globalmente.
- **GTR** (*Global Transformation Rules*): Este componente contiene las reglas globales de transformación, las cuales son utilizadas por el componente GQT para solucionar los conflictos de tipos y unidades de datos.
- **GQT** (*Global Query Transformer*): Este es el componente central de la arquitectura y tiene las funciones de recibir las operaciones ejecutadas de manera global, transformarlas, ejecutarlas en los servidores LDBMS-Server y regresar los resultados a los usuarios globales.
- **LDBMS-Server**: Este componente recibe las operaciones SQL de los usuarios globales, las ejecuta en la base de datos local y regresa el resultado al FDBMS.

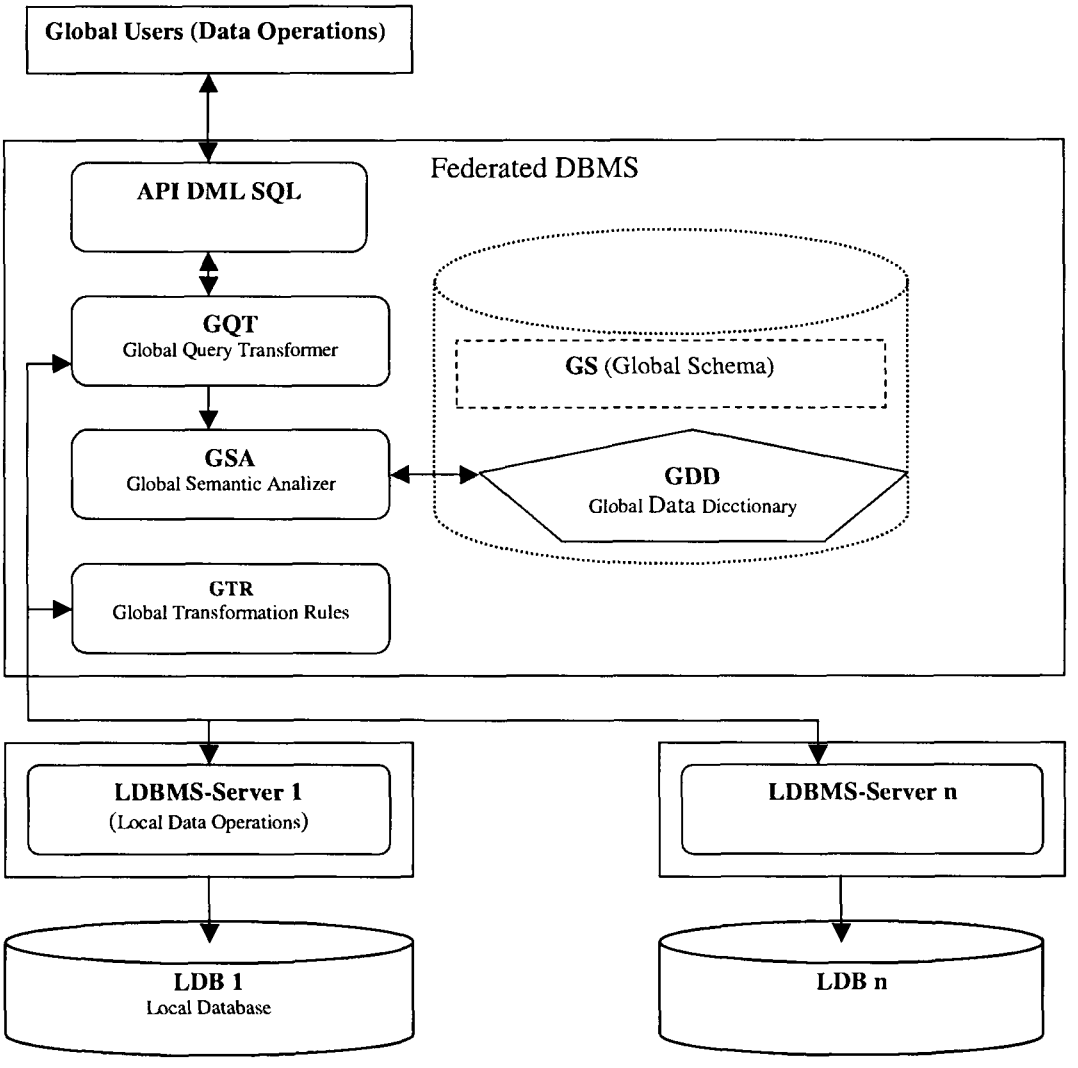

Figura 5.2: Arquitectura para integrar bases de datos propuesta (basada en componentes).

5.4 Secuencia de operaciones de la arquitectura para integrar bases de datos propuesta

Tal como se presenta en la figura 5.3, cuando un usuario global realiza una operación SQL, el FDBMS-Server recibe la llamada mediante el componente GQT y realiza las siguientes operaciones:

1. Obtiene la información semántica enviando mensajes al GSA.
2. En caso de ser necesario, hace transformaciones de tipos de datos y unidades enviando mensajes al componente GTR.
3. Después transforma la operación SQL global en *sub operaciones* para cada base de datos local.
4. Los servidores LDBMS-Server ejecutan los operaciones SQL locales y regresan el resultado al GQT.
5. En caso que se trate de una operación tipo `SELECT`, el GQT hace conciliación de resultados. Las operaciones `INSERT`, `UPDATE` y `DELETE` no regresan resultados en registros de datos al usuario por lo tanto no se requiere conciliación.
6. El GQT regresa el resultado global al usuario.

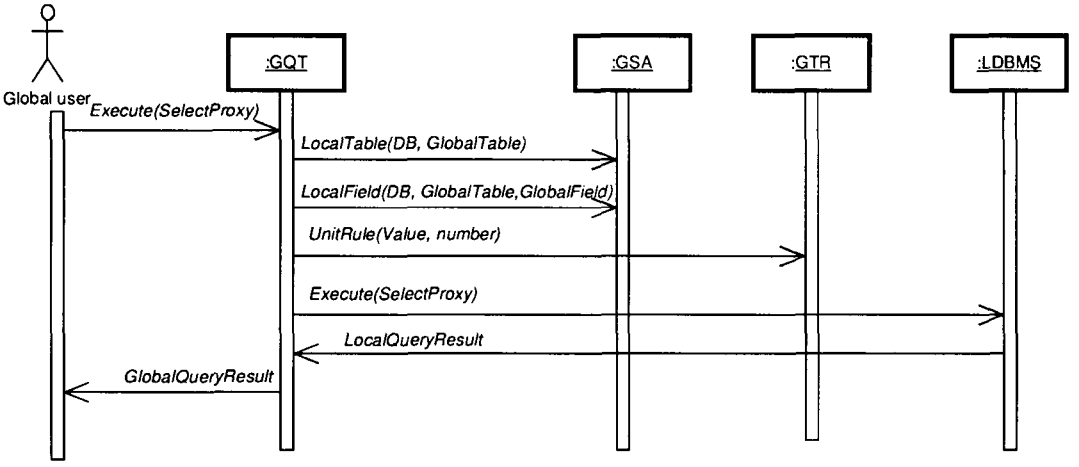

Figura 5.3: Diagrama de secuencia UML de una llamada SQL DML.

En la figura 5.3 se presenta el diagrama de secuencia de una llamada API DML.

En esta figura se presenta la interacción entre los componentes de la arquitectura. Primeramente un usuario global ejecuta una operación SQL. Dicha operación es recibida por el componente GQT el cual se comunica con los componentes GSA, GTR y LDBMS-Server para obtener la información semántica, transformar la operación SQL y ejecutarla en la base de datos local. También se puede apreciar que la operación SQL regresa un resultado al usuario global porque es de tipo SELECT.

5.5 Componentes de la arquitectura

En esta sección se presenta una descripción detallada de los componentes de la arquitectura para integrar bases de datos propuesta en esta tesis.

5.5.1 GDD (Global Data Dictionary)

A diferencia de arquitecturas de bases de datos distribuidas [Álvarez 2003], las cuales están enfocadas en problemas relacionados con la *distribución de los datos*, el propósito fundamental del GDD propuesto en este trabajo de tesis es el registro de las correspondencias semánticas de un GS con cada una de las bases de datos locales. Cuando se hace el registro de las correspondencias semánticas, el GDD debe almacenar, no sólo los atributos y tablas locales, sino también los atributos y tablas globales. Las tablas y atributos globales forman el GS. Por lo tanto el GDD también almacena el GS. Debido a que se debe tener conocimiento de la información semántica, el GS debe ser definido antes de registrar las correspondencias semánticas. En el caso de estudio de este trabajo de tesis, el GS se definió en el capítulo 4 utilizando la metodología para integrar esquemas propuesta, así mismo se identificaron las correspondencias semánticas a nivel tabla, atributo y datos. En esta sección se muestra la forma como se registran estas correspondencias en el GDD.

En la figura 5.4 se presenta el esquema del GDD propuesto en este trabajo de tesis. La estructura del GDD permite el registro de las correspondencias semánticas. El GDD esta compuesto de seis tablas:

1. DATABASES
2. GLOBAL_TABLE
3. LOCAL_TABLE
4. GLOBAL_FIELD
5. LOCAL_FIELD
6. TRANSFORMATION_RULES

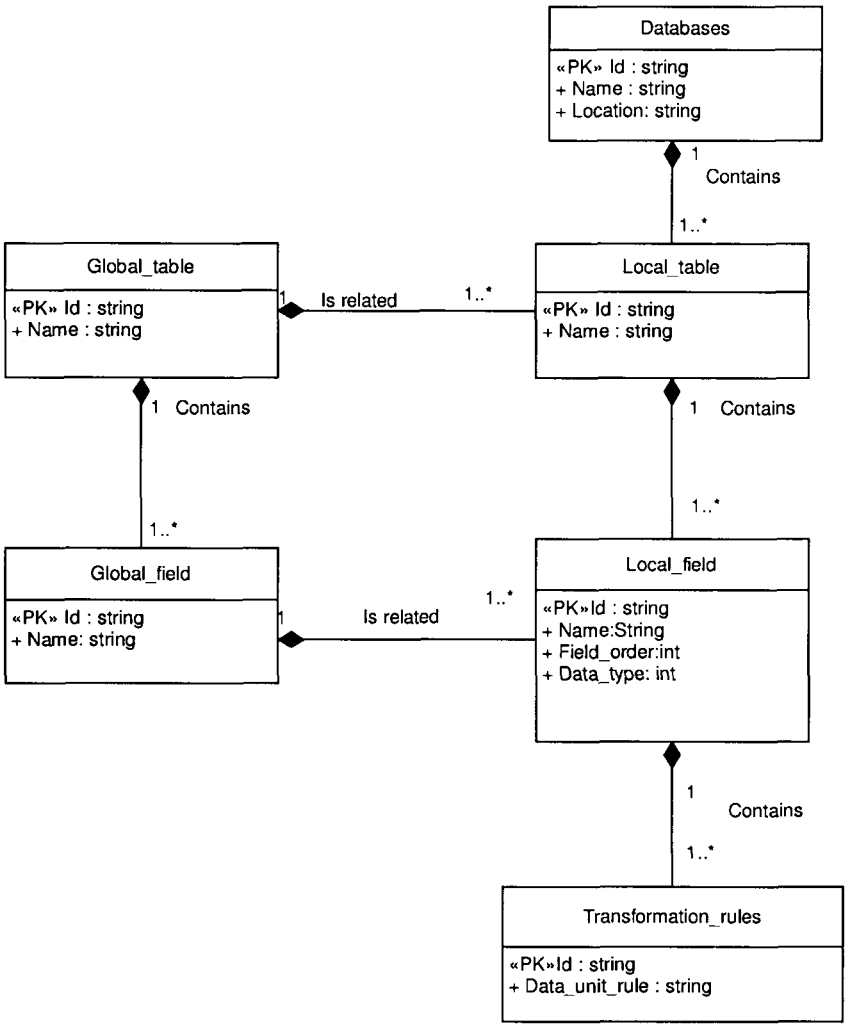

Figura 5.4: Modelo de datos del GDD propuesto.

Las *correspondencias semánticas a nivel tabla* se registran mediante las tablas GLOBAL_TABLE y LOCAL_TABLE. Cada registro en la tabla GLOBAL_TABLE representa una tabla global del GS y contiene una llave primaria que la identifica. Cada registro en la tabla LOCAL_TABLE representa una tabla local y contiene una llave foránea para hacer referencia a su tabla global correspondiente.

Las *correspondencias semánticas a nivel atributo* se registran mediante las tablas GLOBAL_FIELD y LOCAL_FIELD. Cada registro en GLOBAL_FIELD representa un atributo o campo global. Cada registro en LOCAL_FIELD representa un atributo local, esta tabla contiene una llave foránea para hacer referencia a su atributo global correspondiente.

Las *correspondencias semánticas a nivel de dato* deben ser registradas de diferente manera que las correspondencias semánticas a nivel de tabla y atributo porque no es suficiente con una simple referencia al atributo correspondiente. En la mayor parte de los casos es necesario aplicar reglas de transformación, las cuales son difíciles de escribir en un campo de una base de datos. Por tal razón, en este trabajo de tesis se propone la existencia de un componente de software que contenga las reglas para transformaciones a nivel de dato. Sin embargo, dichas reglas deben tener una referencia en el GDD. Estas referencias se registran en la tabla TRANSFORMATION_RULES.

A continuación se presenta la descripción de las tablas del GDD.

5.5.1.1 DATABASES

La tabla DATABASES, presentada en la tabla 5.1 contiene el conjunto de bases de datos que participan en la federación. Un atributo ID contiene la llave que identifica a la base de datos de manera única. El atributo NAME contiene el nombre de la base de datos. Un atributo LOCATION de esta tabla contiene la localización de la base de datos, el cual puede ser una dirección IP.

Table Name	**DATABASES**					
Attributes	Types	PK?	FK?	Unique?	Description	Example
ID	STRING	PK		<Unique>	LLave identificador de cada base de datos local	DB001
Name	STRING				Nombre de la base de datos local	Biblioteca1
Location	STRING				Una dirección de red (IP,) donde se encuentra la base de datos local.	//127.0.0.1

Tabla 5.1: Estructura de la tabla DATABASES.

5.5.1.2 GLOBAL_TABLE

La tabla GLOBAL_TABLE, presentada en la tabla 5.2 contiene el conjunto de tablas del GS. El atributo ID es la llave que identifica a la tabla global de manera única. Un atributo NAME describe el nombre de la tabla.

Table Name	**GLOBAL_TABLE**					
Attributes	Types	PK?	FK?	Unique?	Description	Example
ID	STRING	PK		<Unique>	Llave que identificador de cada tabla local	GT001
Name	STRING			<Unique>	Nombre de cada tabla global	Autores

Tabla 5.2: Estructura de la tabla GLOBAL_TABLE.

5.5.1.3 LOCAL_TABLE

Para cada tabla global del GS debe existir por lo menos una tabla local correspondiente con la cual existe una correspondencia semántica a nivel de tabla. La tabla LOCAL_TABLE del GDD, presentada en la tabla 5.3 contiene las tablas del conjunto de bases de datos locales. El atributo ID contiene la llave que identifica a la tabla local de manera única y el campo DBID contiene la llave que identifica la base de datos local a la que pertenece dicha tabla. El atributo GLOBAL_TABLE contiene la llave foránea de la tabla global correspondiente. El atributo NAME contiene el nombre de la base de datos local.

Table Name	LOCAL_TABLE					
Attributes	Types	PK?	FK?	Unique?	Description	Example
ID	STRING	PK		<Unique>	Contiene la llave que identifica a cada tabla local de manera única.	LT001
DBID	STRING		FK		Contiene la llave foránea de la base de datos local a la que pertenece una tabla.	DB001
GLOBAL_TABLE	STRING		FK		Contiene la llave foránea de la tabla global semánticamente correspondiente a la tabla local	GT001
NAME	STRING				Almacena el nombre de la tabla local	Autor

Tabla 5.3: Estructura de la tabla LOCAL_TABLE.

5.5.1.4 GLOBAL_FIELD

Los campos de las tablas del GS son descritos mediante la tabla GLOBAL_FIELD, presentada en la figura 5.4, ésta contiene un atributo ID, el cual es el identificador del campo global. Un atributo GLOBAL_TABLE contiene las tablas globales a las que pertenecen dichos campos. El atributo NAME contiene el nombre del campo global.

Table Name	GLOBAL_FIELD					
Attributes	Types	PK?	FK?	Unique?	Description	Example
ID	STRING	PK		<Unique>	Contiene la llave que identifica al campo global	GF001
GLOBAL_TABLE	STRING		FK		Contiene la llave foránea que hace referencia a la tabla global a la que pertenece un campo global.	GT001
NAME	STRING				Contiene el nombre del campo global	Id_Autor

Tabla 5.4: Estructura de la tabla GLOBAL_FIELD.

Table Name	LOCAL_FIELD					
Attributes	Types	PK?	FK?	Unique?	Description	Example
ID	STRING	PK		<Unique>	Contiene la llave que identifica de manera única al campo local.	LF001
DBID	STRING		FK		Contiene la llave foránea de la base de datos local a la que pertenece un campo local.	DB001
LOCAL_TABLE	STRING		FK		Contiene la llave foránea de la tabla local a la que pertenece el campo local	LT001
GLOBAL_FIELD	STRING		FK		Contiene la llave foranea del campo global correspondiente al campo local	GF001
FIELD _ORDER	INTEGER				Contiene el orden en el que un campo local debe estar. Aplicable sólo cuando un campo global contiene correspondencias semánticas con varios campos locales.	1
DATATYPE	INTEGER				Contiene el tipo de dato de un campo local. Se utilizan claves de tipo entero: 1, para el tipo entero, 2 para el tipo STRINGo y 3 para el tipo flotante.	2
NAME	STRING				Contiene el nombre del campo local	Autor_Id

Tabla 5.5: Estructura de la tabla LOCAL_FIELD.

5.5.1.5 LOCAL_FIELD

La tabla LOCAL_FIELD, cuya estructura es presentada en la tabla 5.5 es la más importante del GDD, en esta se concentra la mayor parte de la información de las correspondencias semánticas. Para cada campo global del GS debe existir por lo menos un campo local correspondiente. En la tabla LOCAL_FIELD se encuentran los campos locales correspondientes a cada uno de los campos globales del GS. El atributo que identifica al campo local es ID. El atributo DBID es la llave foránea que identifica a la base de datos local donde se encuentra el campo local. El atributo LOCAL_TABLE es la llave foránea para identificar a la tabla local que pertenece el campo local. El atributo que hace referencia al campo global es GLOBAL_FIELD. Tal como se presenta en el capítulo 4 en la sección 4.3.4, para algunos campos globales puede existir más de un campo local correspondiente, descrito como un conflicto de atributos faltantes pero implícitos. Con el fin de solucionar dicho problema, esta tabla contiene un campo FIELD_ORDER, el cual informa si el campo local es el único correspondiente o si existen más, en caso que existan más da la posición de dicho campo. El atributo DATATYPE contiene el tipo de dato del campo local.

5.5.1.6 TRANSFORMATION_RULES

Algunas correspondencias semánticas no son posibles registrarlas directamente mediante correspondencias entre tablas. Tal es el caso de las correspondencias semánticas de nivel de datos. Estas pueden ser de tres tipos: tipos de datos, unidades o expresiones. En la presente tesis se propone que estas correspondencias se implementen mediante reglas, las cuales se encuentran en los métodos del componente GTR. Pero aun cuando se implementan de esta manera, necesitan un identificador que haga referencia a la regla en dicho componte. Este identificador se encuentra almacenado en la tabla

TRANSFORMATION_RULES, la cual es presentada en la tabla 5.6. El campo DATA_UNIT_RULE de esta tabla contiene el número identificador. De esta manera se puede dar apertura a soluciones para problemas complejos por el uso de diferentes expresiones o unidades. En esta investigación se implementó una regla para transformación de tipos de unidades donde en el GS se usa el tipo de unidad "DOLAR" y en las bases de datos locales se utilizan "PESO" y "DOLAR", en este caso la función en el GTR contiene el código para hacer la conversión. Un atributo o campo ID contiene la clave que identifica a la regla de transformación. El atributo LOCAL_FIELD contiene la llave foránea para identificar al campo local que contiene dicha regla.

Table Name	TRANSFORMATION_RULES					
Attributes	Types	PK?	<Unique>	FK	Description	Example
ID	STRING	PK	<Unique>		Contiene la clave de la regla de transformación del campo	TR001
LOCAL_FIELD	STRING			FK	Contiene la llave foránea que identifica el campo local al que se le aplica la regla de transformación.	GF009
DATA_UNIT_RULE	INTEGER				Contiene el identificador de la regla de transformación. Este identificador se envía como parámetro al componente GTR, para que aplique la regla de transformación adecuada	1

Tabla 5.6: Estructura de la tabla TRANSFORMATION_RULES.

Con la definición de las tablas del GDD es posible implementar un componente GDD. La figura 5.5 presenta el componente GDD, en la cual se puede ver que dicho componente es reutilizable, lo cual significa que este mismo GDD se puede implementar para otras arquitecturas de integración de bases de datos.

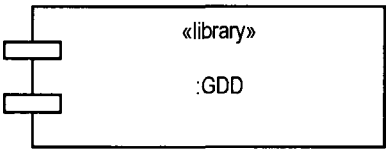

Figura 5.5: Implementación del componente GDD.

5.5.2 API de acceso a los datos

El trabajo [Álvarez 2003] utiliza la API para llamadas básicas DML (Data Manipulation Language) correspondientes a los estatutos SQL `INSERT`, `DELETE` y `UPDATE`, y las API para llamadas DDL (Data Definition Language) correspondiente a los estatutos SQL `CREATE TABLE`, `ALTER TABLE` y `DROP TABLE`.

La principal restricción de un proyecto de integración de bases de datos es respetar la autonomía de las bases de datos locales. Por lo tanto no es posible utilizar la API DDL, ya

que sus operaciones alteran las estructura de las bases de datos locales. Al afectar dichas estructuras, la operación de los usuarios locales también es afectada y de esta manera se afecta la autonomía de las bases de datos locales.

Las operaciones que se manejan son las siguientes:

- `Select`: Selecciona registros del conjunto de bases de datos heterogéneas, los resultados se concilian y forman un resultado global. Algunos de los problemas que resultan al ejecutar una llamada `Select` son: conflictos por nombres de tablas y atributos, estructuras de las tablas y tipos de datos.
- `Insert`: Inserta renglones al conjunto de bases de datos. Algunos de los problemas que se pueden presentar al agregar información nueva son las diferencias en los datos, ya sea por el tipo de estos, por las unidades utilizadas o también por las expresiones.
- `Delete`: Borra renglones de un conjunto de bases de dato heterogéneas. Los problemas que se pueden presentar son en la cláusula `WHERE`, donde en ocasiones, las columnas o atributos no se encuentran en las bases de datos locales, de esta manera se altera la información que la operación debe regresar. Por ejemplo si la cláusula `WHERE` global dice que el año del artículo sea igual a 2003 y en la tabla local no se encuentra el campo correspondiente al año, el resultado se verá afectado. En este trabajo de tesis, se propone que las llamadas con cláusulas `WHERE` que no puedan ser completadas correctamente en las bases de datos locales se anulen para que el resultado global no se vea afectado.
- `Update`: Actualiza renglones en el conjunto de bases de datos heterogéneas, los problemas que se presentan en este tipo de llamadas son una combinación de las dos anteriores.

En la tabla 5.7 se presentan las cláusulas de SQL que son soportadas por los componentes de la arquitectura. Las cláusulas `GROUP BY` y `ORDER BY` no son soportadas, sin embargo es posible extender el componente GQT para darles soporte.

CLAUSULA	¿Soportada?
SELECT	SI
INSERT	SI
UPDATE	SI
DELETE	SI
WHERE	SI
FROM	SI
SET	SI
VALUES	SI
GROUP BY	NO
ORDER BY	NO

Tabla 5.7: Cláusulas SQL soportadas por la arquitectura propuesta.

Con la definición de las operaciones SQL que son posibles procesar mediante esta arquitectura, es posible implementar el componente DBAPI, dicho componente se presenta

en la figura 5.6, en notación UML. La figura muestra que el componente DBAPI tiene interfaces bien definidas, las cuales son las operaciones SQL DML. En esta figura también se puede notar que el componente DBAPI es accedido mediante un mensaje DBConnection.

Algunos ejemplos de operaciones DML se muestran en la tabla 5.8.

OPERACIÓN	EJEMPLO
SELECT	SELECT Articulos.Nombre_artículo, Autores.Nombre_autor FROM Artículos, Autores WHERE Autores.Nombre_autor = "Amit P. Sheth" And Articulos.Autor = Autores.Nombre_autor
INSERT	INSERT INTO Articulos (Articulos.Nombre_artículo, Articulos.Año) VALUES("Metodología para integrar bases de datos", 2003)
DELETE	DELETE FROM Articulos WHERE Articulos.Año = 2002
UPDATE	UPDATE Articulos SET Articulos.Año = 2003 WHERE Articulos.Año = 2002

Tabla 5.8: Algunos ejemplos de llamadas SQL DML.

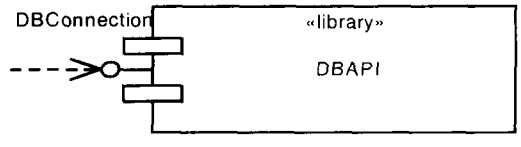

Figura 5.6: Implementación del componente DBAPI de Álvarez [Álvarez 2003].

5.5.3 GSA (Global Semantic Analizer)

El objetivo principal del componente GSA es obtener la información sobre las correspondencias semánticas del GDD. En la figura 5.7 se muestra la clase GSA en notación UML, al realizar una instancia de dicha clase se crea un objeto de este tipo, dicho objeto forma al componente GSA de la arquitectura propuesta.

La información semántica se obtiene por medio de los métodos `get` de esta clase. Por ejemplo: para obtener la tabla local correspondiente a una tabla global se llama al método `getLocalTable`, donde los parámetros que se le envían como entrada son la tabla global y la base de datos de la tabla local.

Cuando se ejecuta el método `getLocalField`, se obtiene el campo local correspondiente a un campo global, y en caso de tener reglas de transformación, este

método obtiene los identificadores de dichas reglas y los guarda en una estructura de datos llamada `LocalColums`. Estos identificadores son posibles obtenerlos mediante los métodos `getLocalFieldOrder, getUnitRule y getDataTypeRule.`

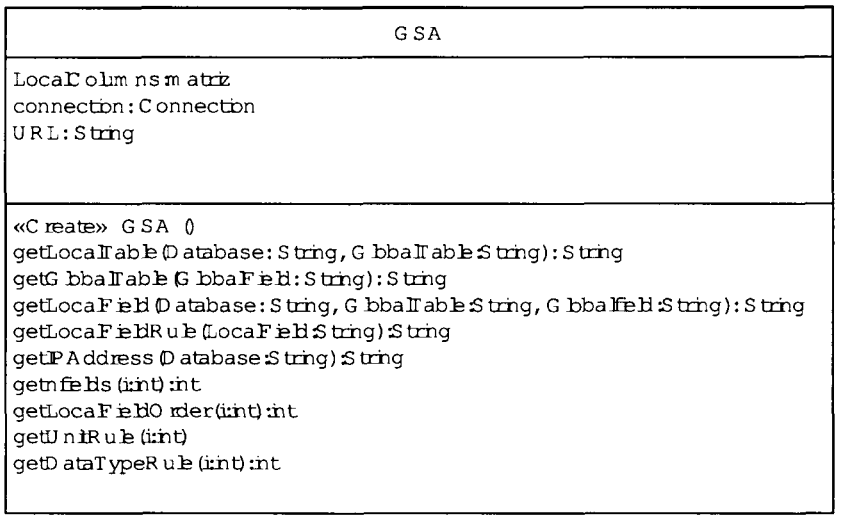

Figura 5.7: Clase del componente GSA.

En la tabla 5.9 se presenta la descripción de los métodos de la clase `GSA`. Cada renglón de esta tabla es la descripción de un método.

Con los métodos de la clase `GSA` es posible obtener la información de las correspondencias semánticas entre tablas y atributos del GS y los esquemas locales. Esta información sirve para hacer las transformaciones que se requieren al ejecutar una operación SQL realizada por un usuario global.

Ya ha sido definida la clase del componente GSA, la cual contiene sus interfaces bien definidas, con estos elementos es posible implementar un componente de tipo GSA, dicho componente se presenta en la figura 5.8.

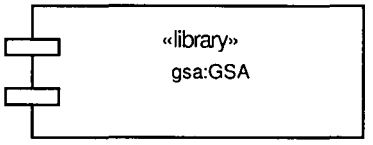

Figura 5.8: Implementación del componente GSA

5.5.4 GTR (Global Transformation Rules)

La clase correspondiente al componente GTR presentada en la figura 5.9, contiene los métodos para hacer conversiones de datos, ya sea por diferentes tipos de datos, unidades o expresiones. Para cambiar el tipo de unidad utilizado por un dato, primeramente se obtiene el número identificador a la regla mediante el método `getUnitRule` del componente GSA, el número identificador se encuentra almacenado en la tabla TRANSFORMATION_RULES del GDD. Después se envía un mensaje al método *`UnitRule`* del componente GTR con los parámetros número identificador a la regla y el dato a ser transformado. El resultado que regresan los métodos de la clase GTR son los datos transformados. Por lo tanto, el uso que se le da a los métodos del componente GTR es en cláusulas SQL enfocadas a los datos tales como `WHERE` y `SET`. En la tabla 5.10 se presentan la descripción de los métodos, parámetros de entrada y de salida de la clase GTR.

METODO	DESCRIPCION	CLAUSULAS	PARAMETROS	
			ENTRADA	SALIDA

`GetLocalTable`	Obtiene la tabla local correspondiente de una tabla global.	`-INSERT INTO`
		`-UPDATE`
		`-DELETE`
		`-FROM`
			-Tabla global del GS.
			-ID de la base de datos local.
				- Tabla local correspondiente.

`GetGlobalTable`	Obtiene la tabla global a la que pertenece un campo global	-	-Un campo global.	- Una tabla global

`GetLocalField`	Obtiene el campo local correspondiente de un campo global.	`-SELECT`
		`-WHERE`
		`-INSERT INTO`
		`-SET`
			-Tabla global del GS.
			- Campo global
			-ID de la base de datos local
				- Campo local correspondiente

`GetIPAddres`	Obtiene la dirección IP de la base de datos local.	-	-ID de la base de datos local.	- Una dirección IP.

`GetLocalFieldOrder`	Obtiene el orden en el que debe de ir un atributo en una concatenación de atributos para formar un atributo global o local, según sea el caso.	`-WHERE`
		`-SET`
		`-VALUES`
			-ID del atributo local.	-Un número identificador del orden.

`GetLocalFieldUnitRule`	Regresa un número identificador a una regla de transformación para tipo de unidades diferentes.	`-WHERE`
		`-SET`
		`-VALUES`
			-ID del atributo local.	- Un número de referencia a la regla de tipos de datos.

`GetLocalFieldDataType Rule`	Regresa un número identificador a una regla de transformación para tipo de dato diferentes.	`-WHERE`
		`-SET`
		`-VALUES`
			-ID del atributo local.	- Un número de referencia a la regla de tipos de datos.

Tabla 5.9: Métodos de la clase GSA.

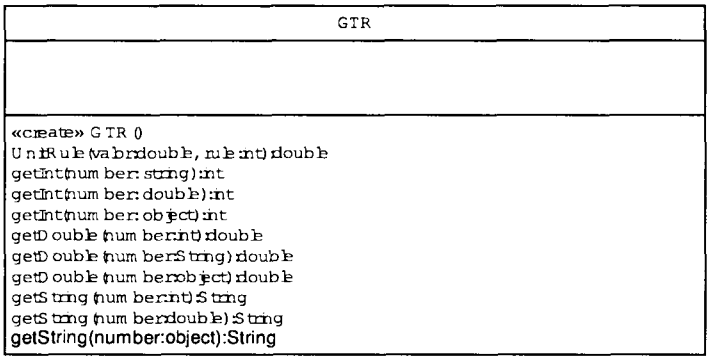

Figura 5.9: Clase del componente GTR.

METODO	DESCRIPCION	CLAUSULAS	PARAMETROS	
			ENTRADA	SALIDA
`UnitRule`	Aplican una regla de transformación de tipo de unidades a un valor dado	-INSERT INTO -UPDATE -DELETE -FROM	-Valor a transformar -ID de la regla de transformación	- Valor transformado
Métodos `getInt`	Aplican una regla para obtener tipo de dato entero dado un valor de cualquier tipo. Esta regla está compuesta de varios métodos que se ejecutan mediante polimorfismo.	-INSERT INTO -UPDATE -DELETE -FROM	-Valor a transformar -ID de la regla de transformación	- Valor transformado
Métodos `getDouble`	Aplican una regla para obtener tipo de dato doble dado un valor de cualquier tipo. Al igual que en la regla anterior esta regla está compuesta de varios métodos que se ejecutan mediante polimorfismo.	-INSERT INTO -UPDATE -DELETE -FROM	-Valor a transformar -ID de la regla de transformación	- Valor transformado
Métodos `getString`	Aplican una regla para obtener tipo de dato String dado un valor de cualquier tipo. Al igual que en las reglas anteriores, esta regla utiliza el polimorfismo.	-INSERT INTO -UPDATE -DELETE -FROM	-Valor a transformar -ID de la regla de transformación	- Valor transformado

Tabla 5.10: Métodos de la clase GTR.

Los métodos de la clase GTR permiten resolver la heterogeneidad semántica por diferencias entre unidades y tipos de datos. En este trabajo de tesis no se implementaron reglas para diferentes expresiones. Sin embargo, se pueden implementar métodos para resolver este tipo de heterogeneidad mediante un nuevo método donde se pueden incorporar las reglas correspondientes. También es posible incorporar nuevas reglas para diferentes unidades en el método `UnitRule`, donde cada nueva regla debe tener un número identificador.

Ya ha sido definida la clase del componente GTR, dicha clase contiene sus interfaces bien definidas, con estos elementos es posible implementar un componente de tipo GTR, este componente se presenta en la figura 5.10.

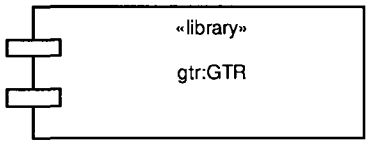

Figura 5.10: Implementación del componente GTR.

5.5.5 GQT (Global Query Transformer)

Este es el principal componente de la arquitectura para integrar bases de datos propuesta. Las principales funciones de este componente son: la transformación de las llamadas SQL DML, la ejecución de estas llamadas en los LDBMS, y la conciliación de los resultados heterogéneos.

Cuando una operación SQL se ejecuta de manera global, el componente GQT recibe la operación encapsulada en un objeto llamado PROXY. Dicho objeto es parte de la API SQL reutilizada del trabajo de Álvarez [Álvarez 2003]. Existen un objeto de tipo PROXY para cada operación SQL: `SelectProxy`, `InsertProxy`, `UpdateProxy` y `DeleteProxy`. El objeto PROXY contiene los métodos y propiedades necesarias para dividir la consulta global en tablas y atributos. Se habla más del objeto `Proxy` en [Álvarez 2003]. Si se tiene conocimiento del tipo de operación `SQL`, métodos y propiedades, entonces es posible transformarla obteniendo las tablas y atributos semánticamente equivalentes mediante el componente GSA.

La figura 5.11 presenta la clase GQT y sus métodos.

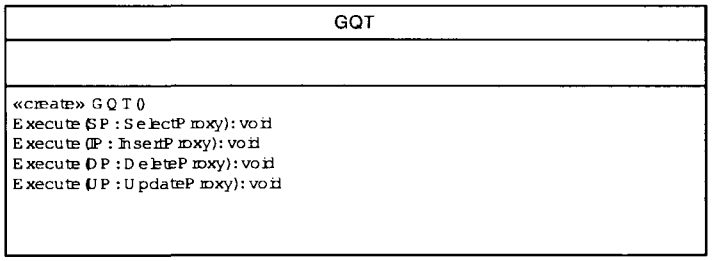

Figura 5.11: Clase del componente GQT.

En la tabla 5.11 se presentan los métodos `Execute` del componente GQT, dichos métodos son la parte central del prototipo, ya que estos envían mensajes a los componentes GSA para obtener las correspondencias semánticas, y al componente GTR para ejecutar las reglas de transformación en los valores de las llamadas. Existen cuatro métodos `Execute` que se ejecutan mediante polimorfismo de acuerdo al tipo objeto PROXY a transformar. El

método `Execute (SP:SelectProxy)` tiene la principal diferencia con los otros tres métodos de tener la capacidad de conciliar los resultados en un resultado único.

Una vez que se han definido los métodos de la clase GQT, es posible implementar un componente de tipo GQT, la figura 5.12 presenta dicho componente.

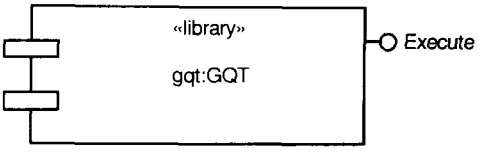

Figura 5.12: Implementación del componente GQT.

METODO	DESCRIPCION	PARAMETROS	
		ENTRADA	SALIDA
`EXECUTE(SP: SelectProxy)`	Toma como entrada una llamada SQL de tipo `SELECT`, la transforma utilizando la información semántica, la ejecuta en los LDBMS-Server y por ultimo concilia los resultados.	`- SelectProxy:` Una operación SQL de tipo `SELECT`	- Resultado de la consulta global
`EXECUTE(IP: InsertProxy)`	Este método toma como entrada una llamada SQL tipo `INSERT`, la transforma y ejecuta en los LDBMS-Server y regresa el resultado al usuario.	`InsertProxy:` Una operación SQL de tipo `INSERT`	- Número de registros afectados por la operación.
`EXECUTE(UP: UpdateProxy)`	Toma como entrada una llamada SQL `UPDATE`, la ejecuta en los LDBMS-Server y regresa el resultado al usuario.	`-UpdateProxy:` Una consulta SQL de tipo `UPDATE`	- Número de registros afectados por la operación
`EXECUTE(UP: DeleteProxy)`	Este método toma como entrada una llamada SQL tipo `DELETE`, la transforma, la ejecuta en cada uno de los LDBMS-Server y regresa el resultado al usuario.	`-DeleteProxy:` Una operación SQL de tipo `UPDATE`	- Número de registros afectados por la operación

Tabla 5.11: Métodos de la clase GQT.

Transformación de las llamadas SQL DML

Como se mencionó en la sección anterior, cuando un cliente global ejecuta una operación SQL, se ejecuta el método `Execute` del componente GQT, este método recibe la operación SQL encapsulada en un objeto de tipo *Proxy*, dicho objeto forma parte de la API SQL DML. En este trabajo de tesis se utilizan cuatro tipos de objetos Proxy: `SelectProxy`, `InsertProxy`, `DeleteProxy` y `UpdateProxy`. Los métodos `get` de un objeto de tipo *Proxy* permiten obtener el detalle de una operación SQL global, lo cual significa que se pueden obtener las tablas, columnas, atributos y datos implicados en la operación. Con este detalle y utilizando las correspondencias semánticas entre el GS y las bases de datos locales, es posible transformar una operación SQL global en sub operaciones para ejecutarse en cada base de datos local.

A continuación se presenta el algoritmo para transformar y distribuir una consulta SQL de tipo SELECT, los pasos que se siguen son los siguientes:

1. El FDBMS-Client crea una operación SQL global mediante el objeto `SelectProxy`. Crear una operación SQL local.
2. El FDBMS recibe la operación a través del componente GQT, y crea una operación SQL para una base de datos local.
3. Se obtienen los atributos locales semánticamente equivalentes que se encuentran en cláusula `SELECT`.
4. Se agregan los atributos locales a la operación SQL de la base de datos local
5. Se obtienen las tablas locales semánticamente equivalentes de la cláusula `FROM` mediante el componente GSA.
6. Se agregan las tablas semánticamente equivalentes a la operación SQL local.
7. Se obtienen los atributos locales semánticamente equivalentes de la cláusula `WHERE` utilizando el componente GSA.
8. Se verifica la existencia de reglas de transformación.
9. En caso de haber reglas de transformación para los datos de la operación, se ejecutan mediante los métodos `DataTypeRule` y `UnitRule` del componente GTR.
10. Se agregan los atributos y datos transformados a la operación SQL local.
11. Se ejecuta la operación SQL local en el LDBMS correspondiente.
12. Se verifica la existencia de más bases de datos. En caso de haber más, se regresa al paso 2.
13. En caso de no haber más bases de datos y de ser una operación `Select` conciliar los resultados resultado único.
14. Regresar el resultado al FDBMS-Client.

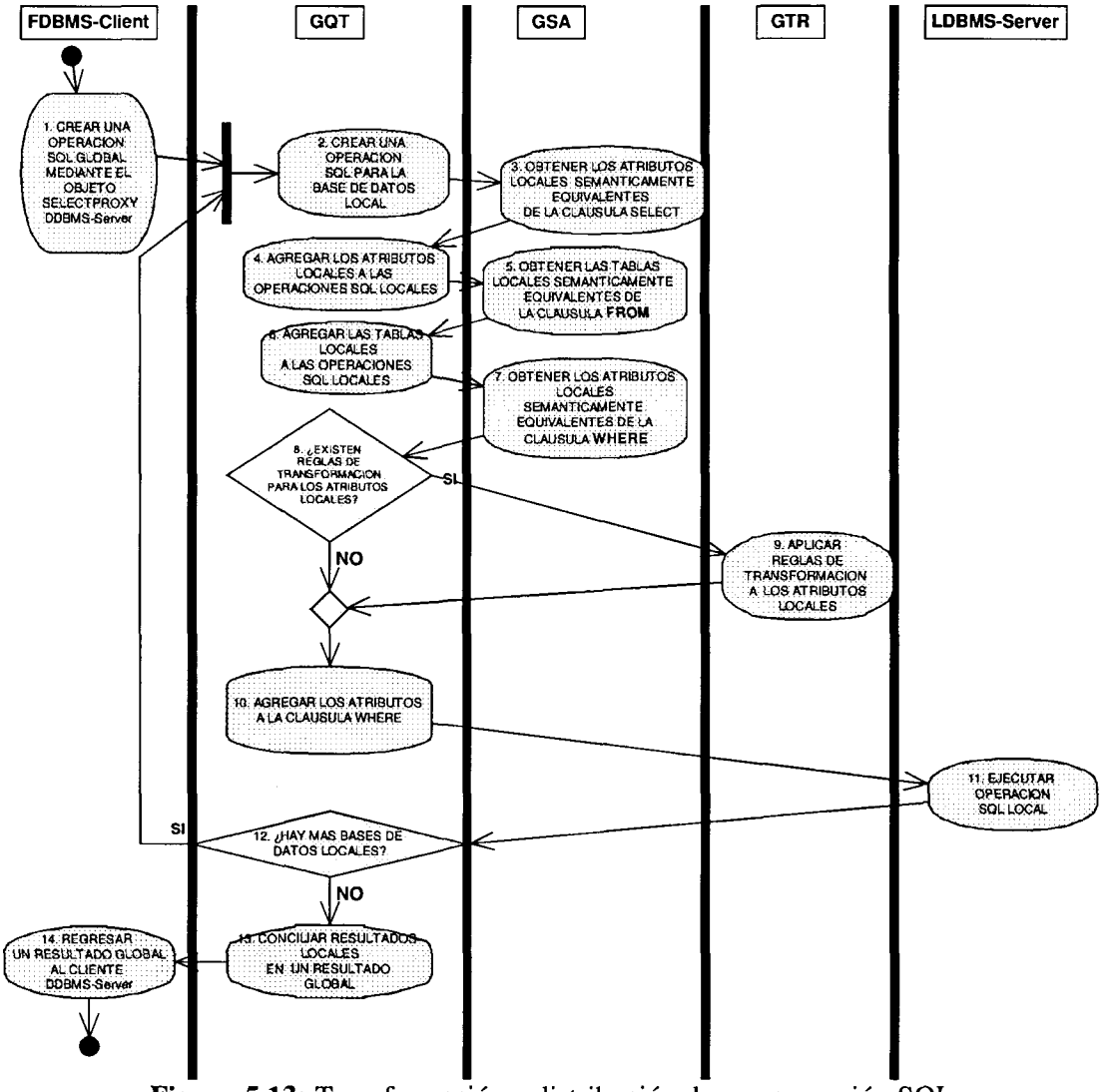

Figura 5.13: Transformación y distribución de una operación SQL.

Para ilustrar a detalle el mecanismo de transformación de una operación `Select`, en la figura 5.13 se presenta el algoritmo que se empleó en esta investigación. Para ilustrar dicho algoritmo se utiliza un diagrama de actividades UML. El algoritmo esta compuesto de tres partes: transformación de las cláusulas `FROM` (tablas), `SELECT` (columnas), y `WHERE`.

5.5.6 Servidor LDBMS-Server

El servidor LDBMS-Server contiene al componente LDBMS, dicho componente ayuda a ejecutar las llamadas SQL DML. En la figura 5.14 se muestra la figura de implementación del servidor LDBMS. Las llamadas SQL DML, hechas por el componente GQT, son ejecutadas en la base de datos local y el resultado se regresa al componente GQT. En el trabajo de Álvarez [Álvarez 2003] se presenta más información sobre el detalle de diseño de este componente.

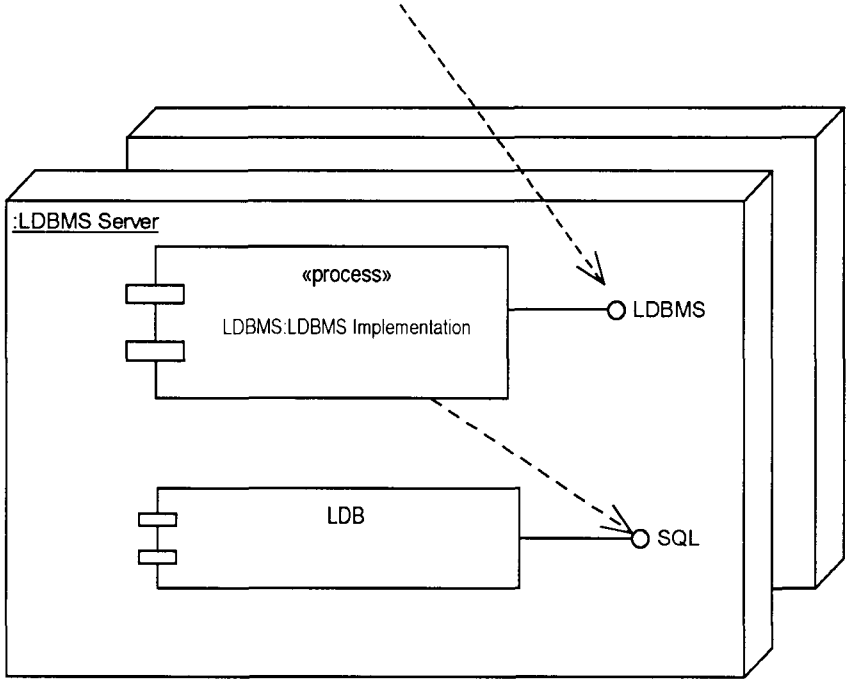

Figura 5.14: Implementación del servidor LDBMS de Álvarez. [Álvarez 2003]

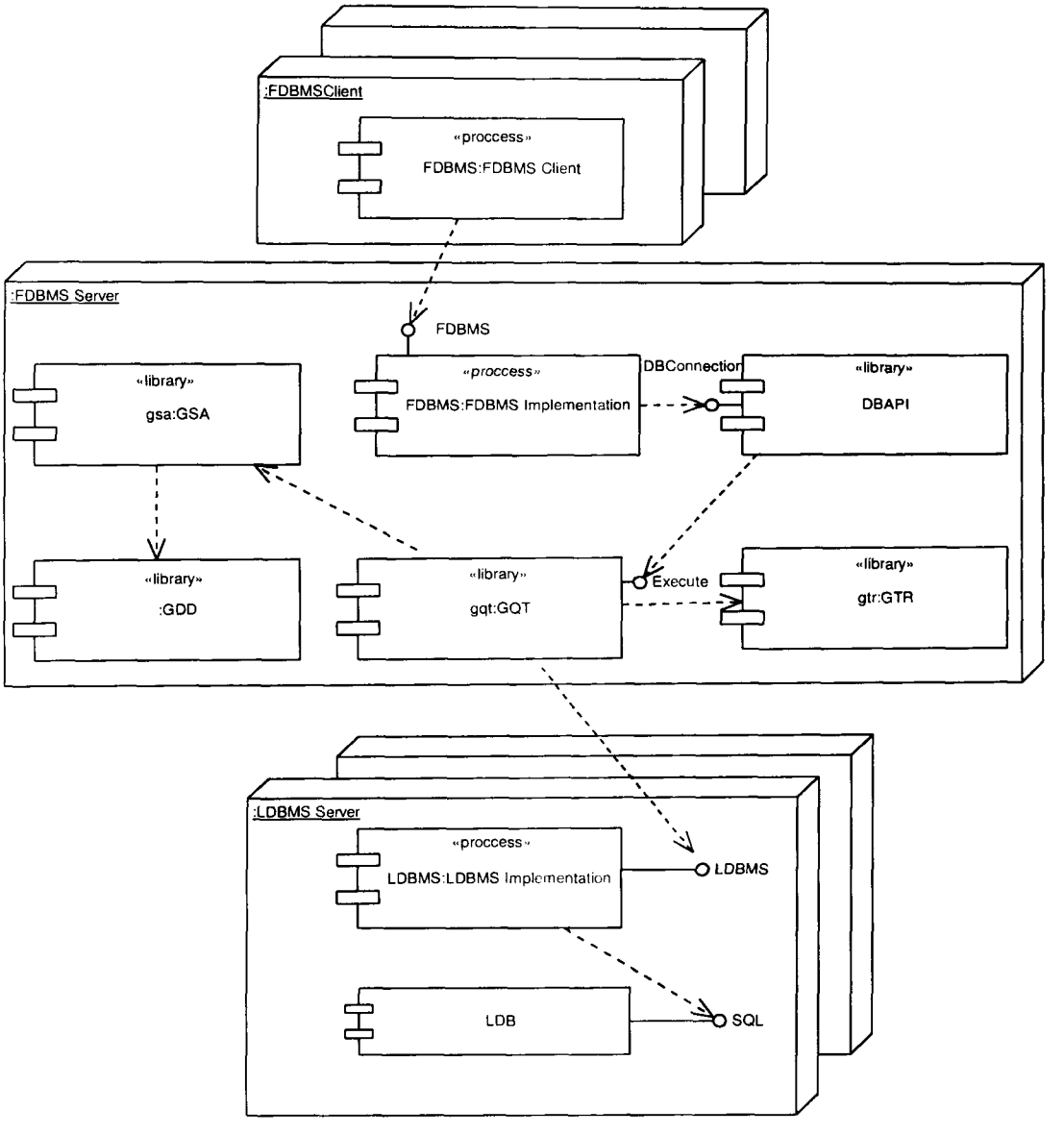

Figura 5.15: Arquitectura de implementación para integrar bases de datos propuesta.

5.6 Arquitectura de implementación de la arquitectura basada en componentes

Una vez definidos todos los componentes de la arquitectura para integrar bases de datos, es posible definir una arquitectura de implementación, en la figura 5.15 se presenta esta arquitectura y sus componentes. En dicha figura se puede notar que existen tres servidores: un cliente FDBMS, un FDBMS-Server y un LDBMS-Server. Las operaciones SQL a nivel global, son ejecutadas en el FDBMS-Server por los usuarios a través de un cliente FDBMS. Por cada operación SQL, el FDBMS crea objetos PROXY a través del componente DBAPI y los envía al componente GQT, el cual interactúa con los componentes GSA, GDD y GTR para hacer las transformaciones necesarias. Por último, el componente GQT ejecuta las operaciones SQL locales en los LDBMS-Servers y en caso de ser necesario, concilia los resultados heterogéneos en un resultado global.

5.7 Método para la implementación de la arquitectura para integrar bases de datos propuesta

Ya se ha definido un esquema global para el caso de estudio, también se ha definido una arquitectura para integrar bases de datos la cual se basa en el registro y extracción de las correspondencias semánticas entre las bases de datos locales. Con estos elementos, es posible definir una estrategia para la implementación física de dichos componentes en el caso de estudio. Para lograr dicha implementación se proponen los siguientes pasos:

1. Crear un GS que represente la unión del conjunto de bases de datos locales.
2. Registrar las correspondencias semánticas del GS con cada una de las bases de datos locales.
3. Implementar un componente para obtener la información semántica, en este caso el componente a implementar es un GSA.
4. Implementar un componente que transforme los datos en caso de ser necesario, en este caso de estudio, el componente es un GTR.
5. Implementar un componente que transforme las llamadas API DML y las ejecute en los servidores LDBMS. En este caso de estudio se implementa el componente GQT que accede a tres LDBMS´s, uno para cada base de datos.
6. Implementar una aplicación para presentar a los usuarios el resultado de las llamadas API DML. Este componente es el cliente del prototipo propuesto en este trabajo de tesis.

5.8 Conclusión

En este capítulo se ha presentado una arquitectura para integrar bases de datos enfocada en el registro y extracción de la heterogeneidad semántica entre un conjunto de bases de datos locales. Se han presentado los componentes de software necesarios para formar esta arquitectura, así como los métodos necesarios que los componen. Asimismo, se ha propuesto una estrategia para la implementación de esta arquitectura, la cual ha aplicado en el caso de estudio.

En este capítulo se concluye que es posible solucionar la heterogeneidad semántica mediante el registro y extracción de las correspondencias semánticas entre las bases de datos locales con el uso de un GDD y componentes de software. Con el empleo de dicha solución es posible integrar un conjunto variable de bases de datos autónomas y heterogéneas, lo cual es el objetivo principal de este trabajo de tesis. En el siguiente capítulo se presentan los detalles físicos y técnicos de la implementación así como las herramientas utilizadas.

Capítulo 6. Implementación del prototipo

Ya se ha definido una metodología para integrar *n* esquemas locales en un esquema global. También se ha presentado la descripción de una arquitectura de componentes de software que permite extraer las correspondencias semánticas entre el esquema global y los esquemas locales para transformar operaciones SQL globales en operaciones locales. Este capítulo presenta los detalles de la implementación del prototipo mediante el caso de estudio: "Integración de tres bases de datos heterogéneas y autónomas".

La organización del capítulo es la siguiente: en la sección 1 se presenta una introducción al prototipo de la tesis, en la sección 2 se presenta la conexión física de las bases de datos locales con los componentes de la arquitectura propuesta, donde se presenta la configuración de las bases de datos locales, así como las herramientas utilizadas para la implementación del prototipo. La sección 3 presenta el GDD con las correspondencias semánticas de las bases de datos del caso de estudio. La sección 4 presenta un análisis de los conflictos presentados entre las bases de datos locales con el GS y se presenta en el análisis del comportamiento del prototipo al ejecutar una consulta global, en la cual se presenta el proceso de su transformación y los conflictos que se resolvieron. En las secciones 5 y 6 se presentan los conflictos resueltos y no resueltos. La sección 7 presenta las conclusiones de este capítulo.

6.1 Introducción al prototipo

En la sección 4.3 se presentó una situación que requiere integración de tres bases de datos, Biblioteca1, Biblioteca2 y Biblioteca3. Estas bases de datos contienen artículos científicos relacionados con el área de computación los cuales pueden ser proceedings y journals. La integración de estas bases de datos es necesaria ya que se requiere que los usuarios puedan acceder a la información que estas contienen como si se trataran de una misma base de datos. Pero la heterogeneidad que existe entre estas bases de datos es un obstáculo para su integración. La heterogeneidad puede ser en cuanto a DBMS´s, esquemas y datos. Para lograr la integración de estas bases de datos, en el capítulo 4 se presentó la forma de solucionar la heterogeneidad semántica mediante un esquema global y en el capítulo 5 se presentó la arquitectura para extraer las correspondencias semánticas para la transformación de operaciones SQL globales. Sin embargo, aún no se ha presentado la forma en la que es posible implementar físicamente estos componentes utilizando las herramientas disponibles, dicha forma se presenta en este capítulo.

6.2 Conexión física de las bases de datos locales con los componentes de la arquitectura

En la sección 4.3 se presentaron los esquemas de las bases de datos locales, de los cuales se construyó un esquema global. En el capítulo 6 se presentó la arquitectura para integrar bases de datos basada en la solución de la heterogeneidad semántica de las bases de datos. Aún cuando se tiene la solución para este tipo de heterogeneidad, existen problemas relacionados con la implementación física de las bases de datos, los cuales son un obstáculo para integrar las bases de datos. Dichos problemas son los siguientes:

- DBMS heterogéneos;
- Localización de las bases de datos locales;
- Comunicación entre los componentes de la arquitectura.

En esta sección se describe la solución a estos problemas. Primeramente se presenta la configuración de las bases de datos, donde se presentan los detalles técnicos del estado de las bases de datos, así como de los DBMS´s que las manejan. Después se presenta la arquitectura de la conexión física de las bases de datos locales, donde se presentan los componentes adicionales que se implementaron, así como las herramientas que se utilizaron y la manera en la que estos componentes solucionan los problemas anteriormente presentados.

6.2.1 Configuración de las bases de datos

En la tabla 6.1 se presenta la configuración de las bases de datos locales.

NOMBRE	DBMS	ENFOQUE	LOCALIZACION(IP)	SOFTWARE LIBRE?
Biblioteca1	SQL Server 8	Relacional	127.0.0.1	NO
Biblioteca2	IBM DB2 8.1	Relacional	127.0.0.1	NO
Biblioteca3	PosgreSQL 7.3.1	Objeto-Relacional	127.0.0.1	SI

Tabla 6.1: Configuración de las bases de datos locales.

Cada base de datos local utiliza diferente DBMS, dichos DBMS tienen diferentes enfoques, SQL Server e IBM DB2 son relacionales mientras que PosgreSQL es objeto-relacional. SQL Server e IBM DB2, son versiones de prueba mientras que PosgreSQL es software libre. La ubicación de las bases de datos es local (*localhost*) porque residen en una máquina local, sin embargo es posible que el prototipo propuesto funcione con diferentes ubicaciones de las bases de datos.

6.2.2 Arquitectura

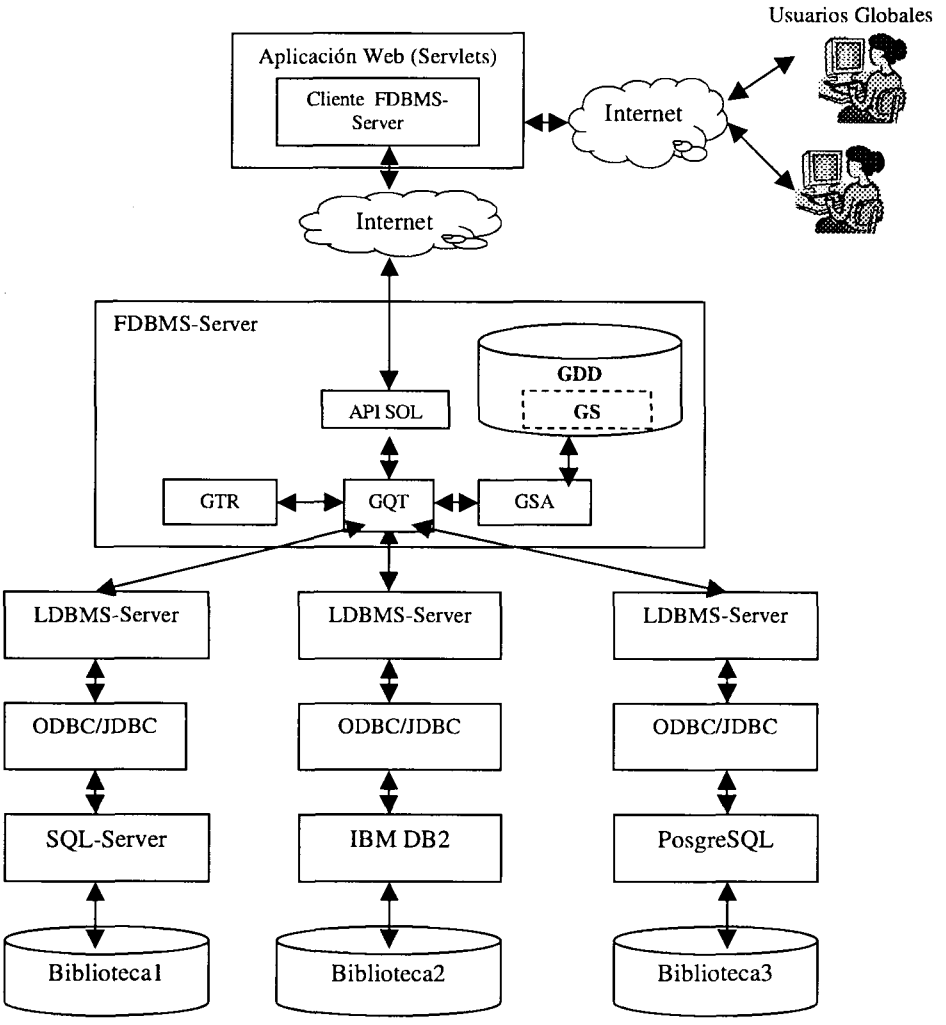

Figura 6.1: Arquitectura de la conexión física de las bases de datos locales.

En la figura 6.1 se presenta la arquitectura de la conexión física en la cual se implementaron los siguientes componentes:

- *Bases de datos locales:* Son las bases de datos que se integran mediante la arquitectura, en este prototipo existen tres: Biblioteca1, Biblioteca2 y Biblioteca3.
- *DBMS:* Cada base de datos local está manejada por un DBMS diferente, en este prototipo se implementó SQL-Server en Biblioteca1, IBM DB2 para Biblioteca2 y PosgreSQL para Biblioteca3.
- *ODBC/JDBC:* Es la capa de middleware orientado a bases de datos que se implementó para solucionar la heterogeneidad por diferentes DBMS.

- *LDBMS-Server:* Es el componente que ejecuta las llamadas API DML en los DBMS locales. Existe un nodo por cada DBMS. Para lograr la integración entre los LDBMS-Server con los DBMS´s en cada nodo LDBMS-Server se hace referencia al controlador ODBC/JDBC de cada DBMS.
- *FDBMS-Server:* Este componente recibe las llamadas API DML del cliente FDBMS, las transforma y las distribuye en los LDBMS-Server. En la arquitectura existe un FDBMS-Server el cual contiene los componentes de la arquitectura propuesta, incluyendo el GDD. La comunicación entre el Cliente-FDBMS con el FDBMS-Server, y del FDBMS-Server con los LDBMS-Server es lograda mediante Java RMI. Dicha herramienta permite crear y registrar estos componentes mediante objetos distribuidos. En el prototipo se crearon un objeto FDBMS-Server y tres objetos LDBMS-Server. El Cliente-FDBMS accede a los métodos de FDBMS-Server, de tal manera que le permite ejecutar las llamadas SQL-DML realizadas por los usuarios globales. El FDBMS-Server después de recibir el mensaje de ejecución de una llamada SQL global, utiliza sus componentes para transformarla y distribuirla en los LDBMS-Server´s.
- *Aplicación Web (Servlets).* Se implementó una aplicación Web utilizando Servlets, la cual se implementó mediante Apache Tomcat, esta herramienta es software libre(*Open Source*) para crear servidores Web. Cuando un usuario global realiza una operación SQL, la ejecuta en la aplicación Web mediante Servlets. Los Servlets crean un cliente FDBMS-Server. Dicho cliente se comunica con el FDBMS-Server mediante la herramienta Java-RMI. De esta manera se permiten ver a los métodos del objeto FDBMS-Server como locales al cliente FDBMS-Server.
- *Usuarios Globales (Global Users).* Los usuarios globales son los que realizan las operaciones SQL en la aplicación Web. En la arquitectura presentada se puede notar que los usuarios pueden estar en cualquier sitio de Internet. Sin embargo, la misma arquitectura se puede aplicar a una LAN.

En el prototipo de esta tesis sólo se integraron tres bases de datos, sin embargo, es posible integrar un número variable de estas. De la misma manera es posible implementar un número variable de clientes FDBMS-Server y cada cliente puede tener un número variable de usuarios. Por lo tanto, la arquitectura es abierta, es decir es posible trabajar con *n* bases de datos y *n* usuarios.

Aunque en este prototipo se utilizó Java RMI para la implementación de objetos distribuidos, también es posible implementar esta misma arquitectura utilizando CORBA. Implementando CORBA, es posible que la arquitectura funcione con componentes desarrollados en diferentes plataformas de programación. Por ejemplo: el FDBMS-Server puede ser desarrollado en Java, mientras que algunos LDBMS-Server en lenguaje C y otros en Java. También es posible implementar la arquitectura con *Web Services*, donde cada nodo LDBMS-Server podría ser un objeto Web.

6.3 Datos del GDD

Los datos del GDD propuesto en esta tesis registran la estructura del GS y sus correspondencias semánticas con los esquemas de las bases de datos locales. En el capítulo cinco ya se ha definido un GS para las tres bases de datos del caso de estudio, en el capítulo anterior se describieron las tablas del componente GDD. Con estos elementos es posible capturar en el GDD la descripción de la estructura del GS, la estructura de los esquemas de las bases de datos locales y las correspondencias semánticas entre éstos.

A continuación se presentan las tablas de datos del GDD.

6.3.1 Databases

En la tabla 6.2 se presentan los datos de la tabla DATABASES, cada registro de esta tabla representa una base de datos local. En el prototipo de esta tesis se implementaron tres bases de datos locales.

DATABASES		
Id	Name	Address
BD001	Biblioteca1	//127.0.0.1/LDBMS1
BD002	Biblioteca2	//127.0.0.1/LDBMS2
BD003	Biblioteca3	//127.0.0.1/LDBMS3

Tabla 6.2: Datos de la tabla DATABASES.

6.3.2 Global_table

Los datos de la tabla GLOBAL_TABLE son presentados en la tabla 6.3. Cada registro representa una tabla del GS. En el GS que se construyó en el capítulo 4. Se identificaron cinco tablas. Cada una de estas tablas tiene por lo menos una correspondencia semántica con las tablas de las bases de datos locales.

GLOBAL_TABLE	
Id	**Name**
GT001	Autor
GT002	Arts
GT003	Journal
GT004	Proceeding_conf
GT005	Conferencia

Tabla 6.3: Datos de la tabla GLOBAL_TABLE.

6.3.3 Local_table

Los datos de la tabla LOCAL_TABLE son presentados en la tabla 6.4. Cada registro en esta tabla representa una tabla de una base de datos local, la referencia a dicha base de datos se realiza mediante la llave foránea DBID. Cada tabla local contiene una correspondencia semántica a una tabla global. La correspondencia es hecha mediante la llave foránea GLOBAL_TABLE.

LOCAL_TABLE			
Id	**DBID**	**Global_table**	**Name**
LT001	DB001	GT001	Autor
LT002	DB001	GT002	Arts
LT003	DB001	GT003	Journal
LT004	DB001	GT004	Proceeding_conf
LT005	DB002	GT001	Autores
LT006	DB002	GT002	Artículos_Conf
LT007	DB002	GT005	Conferencia
LT008	DB003	GT001	Autores_Art
LT009	DB003	GT002	Artículos_Journal
LT010	DB003	GT003	Vol_journal

Tabla 6.4: Datos de la tabla LOCAL_TABLE.

6.3.4 Global_field

La tabla 6.5 presenta los datos de la tabla GLOBAL_FIELD. Cada registro en esta tabla representa un campo global que pertenece a una tabla del GS. La llave foránea GLOBAL_TABLE hace referencia a la tabla que pertenece un campo global. Cada campo global contiene por lo menos una correspondencia semántica con un atributo local.

GLOBAL_FIELD		
Id	Global_table	Name
GF001	GT001	Autor_Id
GF002	GT001	Nombre_autor
GF003	GT001	Lugar_trabajo
GF004	GT002	Autor_art
GF005	GT002	Artículo_id
GF006	GT002	Keywords
GF007	GT002	Titulo
GF008	GT002	Año_pub
GF009	GT002	Paginas
GF010	GT002	Issue
GF011	GT002	Precio_art
GF012	GT003	Vol
GF013	GT003	No
GF014	GT003	ISSN
GF015	GT003	Revista
GF016	GT004	Lugar
GF017	GT004	Nombre
GF018	GT004	ISBN
GF019	GT005	Nombre
GF020	GT005	Año
GF021	GT005	Lugar
GF022	GT005	ISBN

Tabla 6.5: Datos de la tabla GLOBAL_FIELD.

6.3.5 Local_field

Los datos de la tabla LOCAL_FIELD son presentados en la tabla 6.6, cada registro de esta tabla representa un campo local y cada campo local pertenece a una tabla de una base de datos local, mediante la llave foránea LOCAL_TABLE se hace la referencia a la tabla local a la que pertenece el atributo local.

LOCAL_FIELD						
Id	DBID	Local_table	Global_table	Global_field	Field_order	Name
LF001	DB001	LT001	GT001	GF001		Id
LF002	DB001	LT001	GT001	GF002		Nombre_autor
LF003	DB001	LT001	GT001	GF003		Lugar_trabajo
LF004	DB001	LT001	GT001	GF004		Autor_id
LF005	DB001	LT002	GT002	GF005		Id
LF006	DB001	LT002	GT002	GF006		Keywords
LF007	DB001	LT002	GT002	GF007		Titulo
LF008	DB001	LT002	GT002	GF008		Año_pub
LF009	DB001	LT002	GT002	GF011		Precio_art
LF010	DB001	LT003	GT002	GF012		Autor_art
LF011	DB001	LT003	GT003	GF013		Vol
LF012	DB001	LT003	GT003	GF014		No
LF013	DB001	LT003	GT003	GF015		ISSN
LF014	DB001	LT003	GT003	GF016		Revista
LF015	DB001	LT004	GT004	GF017		Lugar
LF016	DB001	LT004	GT004	GF018		Nombre
LF017	DB001	LT004	GT004	GF019		ISBN
LF018	DB002	LT005	GT001	GF001		Clave
LF019	DB002	LT005	GT001	GF002	1	Nombre
LF020	DB002	LT005	GT001	GF002	2	Apellido
LF021	DB002	LT006	GT002	GF004		Clave_autor
LF022	DB002	LT006	GT002	GF005		Id
LF023	DB002	LT006	GT002	GF007		Titulo_art
LF024	DB002	LT006	GT002	GF011		Precio
LF025	DB002	LT007	GT005	GF020		Nombre
LF026	DB002	LT007	GT005	GF021		Año
LF027	DB002	LT007	GT005	GF022		Lugar
LF028	DB002	LT007	GT005	GF023		ISBN
LF029	DB003	LT008	GT001	GF001		Id
LF030	DB003	LT008	GT001	GF002		Nombre_autor
LF031	DB003	LT008	GT001	GF003		Lugar
LF032	DB003	LT009	GT002	GF007		Nombre_art
LF033	DB003	LT009	GT002	GF008		Año
LF034	DB003	LT009	GT002	GF009		Paginas
LF035	DB003	LT009	GT002	GF010		Issue
LF036	DB003	LT010	GT003	GF014		Número
LF037	DB003	LT010	GT003	GF015		ISSN
LF038	DB003	LT010	GT003	GF016		Nombre_rev

Tabla 6.6: Datos de la tabla LOCAL_FIELD

Cada atributo local tiene una correspondencia semántica con un atributo global, para hacer esta correspondencia la llave foránea GLOBAL_FIELD hace referencia al atributo global correspondiente. Es posible que varios campos locales tengan una correspondencia semántica con el mismo atributo global, esto se puede notar en la tabla cuando el atributo GLOBAL_FIELD contiene el mismo valor, es decir, hace referencia al mismo campo global.

6.3.6 Transformation_Rules

La tabla 6.7 presenta los datos de la tabla TRANSFORMATION_RULES. Cada registro de esta tabla representa un identificador que hace referencia a una regla que se encuentra en el componente GTR. La llave foránea LOCAL_FIELD hace referencia al atributo local que se le aplicarán las reglas indicadas en el atributo DATA_TYPE_RULE. Dicha regla se aplica a un atributo local cuando los tipos de datos, unidades o expresiones que se utilizan en el GS son diferentes a las de las bases de datos locales.

En este caso se implementó una regla para transformar tipos de unidades de dólares a pesos, ya que la unidad utilizada para el atributo global PRECIO de la tabla global ARTICULOS es dólares mientras que en la base de datos a la que pertenece el campo con llave LF009 utiliza pesos.

TRANSFORMATION_RULES		
Id	Local_field	Data_Type_rule
TR001	LF009	2

Tabla 6.7: Datos de la tabla TRANSFORMATION_RULES.

En la sección 6.5 en el paso 6 se muestra la forma en la que se hacen las transformaciones a los datos mediante el componente GTR.

6.4 Conflictos encontrados entre el GS y las bases de datos locales

Una ventaja de crear el GS es que los conflictos entre *n-esquemas* de bases de datos se reducen a conflictos entre *2-esquemas*: el GS y cada base de datos local. Sin embargo, después de crear el GS, aun existen conflictos, estos conflictos son entre el GS y los esquemas de las bases de datos locales. Estos conflictos existen porque no es posible crear un esquema global que sea homogéneo con todos los esquemas locales, sin embargo la heterogeneidad semántica se puede resolver más fácil entre 2-esquemas que entre n-esquemas.

De acuerdo a la clasificación de conflictos de Kim y Seo [Kim y Seo 1991], se encontraron los siguientes conflictos:

Esquema:

1. *Conflictos por diferentes nombres para atributos semánticamente equivalentes:* por ejemplo: la tabla AUTOR en el GS contiene el atributo AUTOR_ID, cuyos atributos correspondientes son AUTOR_ID en Biblioteca1, CLAVE en Biblioteca2 e ID en Biblioteca3.
2. *Conflictos por diferentes nombres para tablas semánticamente equivalentes*: por ejemplo, la tabla AUTOR del GS, es AUTOR en Biblioteca1, AUTORES en Biblioteca2, y AUTORES_ART en Biblioteca3.
3. *Conflictos por atributos faltantes pero implícitos:* por ejemplo en la tabla AUTOR del GS se encuentra el atributo NOMBRE, cuyos atributos correspondientes en Biblioteca2 son NOMBRE y APELLIDO.
4. *Atributos faltantes:* las tablas de las tres bases de datos contienen diferente número de atributos, lo cual indica que algunas tablas no contienen todos los atributos de sus tablas semánticamente equivalentes.
5. *Tablas faltantes:* En el GS se encuentran todas las tablas semánticamente equivalentes de las bases de datos locales, pero estas últimas no contienen todas las tablas semánticamente equivalentes del GS.

Datos:

1. *Diferentes tipos de datos*: El atributo AÑO_PUB de la tabla ARTS del GS es de tipo `String`, mientras que en la tabla ARTS de Biblioteca1 su atributo semánticamente equivalente es de tipo `Integer`.
2. *Diferentes unidades*: En la tabla ARTS del GS se encuentra el campo PRECIO_ART que maneja el tipo de unidad "DÓLAR" pero en Biblioteca1 el campo semánticamente equivalente PRECIO_ART utiliza el tipo de unidad "PESOS".

6.5 Análisis de la transformación de las operaciones globales

En el capítulo anterior se presentó el algoritmo que sigue el método `Execute` del componente GQT para transformar las operaciones SQL globales en operaciones SQL locales. En este capítulo se presenta el análisis de la transformación de una operación SQL global en operaciones SQL locales mediante un ejemplo. Dicho ejemplo es el siguiente: supóngase que un usuario global desea conocer "los artículos y autores, cuyo precio sea de 9 dólares".

Las etapas de este proceso son las siguientes:

1. Un usuario global ejecuta un operación SQL global en el FDBMS a través de la aplicación Web.

Operación SQL global:

```
SELECT Autor.Nombre_autor, Arts.Titulo
FROM Autores, Arts
WHERE Arts.Precio_art = 9
AND Autor.Autor_id = Arts.Autor_art;
```

2. La operación SQL global es recibida por el método *EXECUTE* del componente GQT.
3. Se obtienen los datos de la base de datos local: Nombre, Id, localización.
   Nombre: Biblioteca1
   Id: DB001
   Localización: 127.0.0.1

4. Se transforma las cláusulas FROM de la operación SQL global, y se agregan la operación SQL local:
   a) Se buscan las tablas locales correspondientes utilizando el componente GSA.

METODO EJECUTADO	TABLA LOCAL RESULTANTE
GSA.LocalTable("DB001", "Autor");	"Autor"
GSA.LocalTable("DB001", "Arts");	"Arts"

   b) Se agregan las tablas locales a la operación SQL local mediante el método FROM.

METODO EJECUTADO	SIGNIFICADO SQL
LocalSelect.Table("Autor")	FROM Autor, Arts
LocalSelect.Table("Arts")	

5. Se transforma las cláusulas SELECT
   a) Se buscan las columnas locales correspondientes utilizando el componente GSA.

METODO EJECUTADO	ATRIBUTO LOCAL RESULTANTE
GSA.LocalField("DB001", "Arts","Titulo");	"Titulo"
GSA.LocalfField"DB001","Autor",Nombre_autor");	"Nombre_autor"

   b) Se agregan las columnas locales a la operación SQL local mediante el método Column.

METODO EJECUTADO	SIGNIFICADO SQL
LocalSelect.Column("Autor.Nombre_autor")	SELECT Autor.Nombre, Arts.Titulo
LocalSelect.Column("Arts.Titulo")	

6. Se transforma la cláusula WHERE
    a) Se buscan las columnas locales correspondientes utilizando el componente GSA.

METODO EJECUTADO	ATRIBUTO LOCAL RESULTANTE
GSA.LocalField("DB001", "Arts","Precio_art");	"Precio_art"

    b) Se verifica mediante el componente GSA, si las columnas locales tienen reglas de transformación para aplicar a los datos de la cláusula WHERE. En caso de haber reglas, se aplican y se obtiene el nuevo dato transformado.

METODO EJECUTADO	Resultado
UnitRule = GSA.getUnitRule("Precio_art")	2
If (UnitRule > 0)	Yes

METODO EJECUTADO	Resultado
NuevoValor = GQT.UnitRule(9, 2)	90

    c) Se agrega la cláusula WHERE a la operación SQL local mediante el método WHERE.

METODO EJECUTADO	SIGNIFICADO SQL
LocalSelect.WHERE("Arts.Precio_art","=", 90)	WHERE Arts.Precio_art = 90

7. Se transforma la cláusula WHERE de tipo JOIN
    a) Se buscan las columnas locales correspondientes utilizando el componente GSA.

METODO EJECUTADO	ATRIBUTO LOCAL RESULTANTE
GSA.LocalField("DB001", "Autor","Autor_id");	"Autor_id"
GSA.LocalfField"DB001","Arts",Autor_art);	"Autor_art"

    b) Se agregan la cláusula WHERE con las columnas locales a la operación SQL local mediante el método *JOIN*.

METODO EJECUTADO	SIGNIFICADO SQL
LocalSelect.JOIN(Autor.Autor_id", "Arts.Autor_art")	AND Autor.Autor_id= Arts.Autor_art

8. Se ejecuta la operación SQL local y se guarda el resultado

BIBLIOTECA1, DB001, 127.0.0.1		

LocalSelect	`SELECT Autor.Nombre_autor, Arts.Titulo`
	`FROM Autor, Arts`
	`WHERE Arts.Precio_art = 90`
	`AND Arts.Autor_art = Autor.Autor_id`	

Resultado	**Autor.Nombre_Autor**	**Arts.Titulo**
	Laks V. S. Lashmanan	Interoperability of multiple autonomous database

9. Se repiten los pasos del 3 al 8, hasta el número de bases de datos locales. En Biblioteca2 se puede notar que el atributo global NOMBRE_AUTOR fue dividido en NOMBRE y APELLIDO.

BIBLIOTECA2, DB002, 127.0.0.1			

LocalSelect	`SELECT Autores.Nombre, Autores.Apellido,`
	`Articulos_Conf.Titulo_art,`
	`FROM Autores, Articulos_Conf`
	`WHERE Articulos_Conf.Precio = 9.0`
	`AND Articulos_Conf.Clave_autor = Autores.Clave;`		

Resultado	**Autores.Nombre**	**Autores.Apellido**	**Articulos_Conf.Titulo_art,**
	Farshad	Hakimpour	Resolving semantic heterogeneity in schema integration

En Biblioteca3 no se ejecutó la operación SQL debido a que no existe el atributo "Precio"y este atributo se encuentra en la cláusula `WHERE`. Ejecutar la operación SQL local, sin esta cláusula `WHERE` es alterar el resultado global, ya que se presentarían artículos cuyo precio no sería de 9 dólares.

BIBLIOTECA3, DB003, 127.0.0.1		

LocalSelect	`SELECT Autores_Art.Nombre_autor,`
	`Articulos_Journal.Nombre_art`
	`FROM Autores_Art, Articulos_Journal`
	`WHERE Articulos_Journal.Autor_id = Autores_Art.Id;`	

Resultado	**Autores_art.Nombre_autor**	**Articulos_Journal.Nombre_art**
	No se ejecutó el operación SQL local.	

10. Se concilian el conjunto de resultados heterogéneos
    a) Se crea una tabla temporal con las columnas de la operación SQL global

TEMPORAL	
Autores.Nombre_autor	**Articulos.Nombre_artículo**

    b) Se inserta cada registro resultante de cada operación SQL local en la tabla temporal

TEMPORAL		
Autores.Nombre_autor		**Articulos.Nombre_artículo**
Laks V. S. Lashmanan		Interoperability of multiple autonomous database
Farshad	Hakimpour	Resolving semantic heterogeneity in schema integration

    c) Se seleccionan los registros de la base de datos temporal ignorando los repetidos. Esta selección se guarda en un resultado global. El método `Execute` del componente GQT regresa un resultado global a la aplicación Web.
    d) La aplicación Web presenta el resultado al usuario

Laks V. S. Lashmanan, Interoperability of multiple autonomous database
Farshad Hakimpour, Resolving semantic heterogeneity in schema integration

6.6 Conflictos solucionados

De acuerdo al ejemplo presentado en la sección 6.5, se puede notar que el usuario global sólo conoce la existencia de un esquema global, así como los tipos de datos y unidades que se utilizan en las instancias "virtuales" de dicho esquema", es decir, al usuario global se le proporciona transparencia de semánticas así como de DBMS. Por lo tanto, según la clasificación de Kim y Seo [Kim y Seo 1991], los conflictos solucionados son los siguientes:

Esquema:

- Nombres diferentes para tablas semánticamente equivalentes
- Nombres diferentes para atributos semánticamente equivalentes
- Nombres iguales para atributos semánticamente no equivalentes
- Nombres iguales para tablas semánticamente no equivalentes
- Atributos faltantes
- Atributos faltantes pero implícitos
- Tablas faltantes
- Tipos de datos diferentes

Datos:

- Diferentes unidades

6.7 Conflictos no solucionados

Conflictos de esquema:

- Restricciones de Integridad
- Conflictos de tabla versus atributo

Conflictos de datos:

- Diferentes expresiones
- Datos obsoletos o incorrectos
- Diferentes precisiones

Los conflictos de restricciones de integridad no fueron solucionados, los conflictos de tabla versus atributo no fueron implementados en este prototipo pero es posible capturar correspondencias semánticas que les den solución.

Los conflictos de diferentes expresiones y diferentes precisiones no fueron implementados en la arquitectura del prototipo. Sin embargo, éstos pueden ser solucionados agregando reglas de trasformación en el componente GTR de la arquitectura del prototipo propuesto en esta tesis.

Los datos obsoletos o incorrectos no fueron solucionados ya que falta un mecanismo para detectarlos, en general, para la solución de estos conflictos no existen reglas precisas a seguir. Un ejemplo de este tipo de conflictos es cuando se tienen dos tablas semánticamente equivalentes AUTORES, las cuales tienen un atributo semánticamente equivalente LUGAR_DE_TRABAJO. Podría ser que ambas tablas tengan registros acerca del mismo autor, pero en algunas no se haya actualizado el lugar donde trabaja el autor. Para estos casos es difícil detectar cúando el dato es incorrecto. Sin embargo se pueden hacer análisis en las tablas de ambas bases de datos para establecer de cúal tomar el dato actualizado y de esta manera implementar alguna regla que indique la acción correcta a realizar.

6.8 Conclusión

En el presente capítulo se ha presentado la implementación física del prototipo de tesis mediante un caso de estudio.

Se han analizado las bases de datos locales con un enfoque en el estado físico, es decir, los DBMS que las manejan, ubicación y conexión física a los componentes de la arquitectura. Asimismo, se han descrito los clientes Web para el FDBMS que se crearon, así como las herramientas que se utilizaron.

También se analizaron los conflictos encontrados entre el GS con cada una de las bases de datos locales.

Se han analizado las etapas de ejecución, transformación, conciliación de datos de una operación SQL global, mediante un ejemplo.

Por ultimo se presentaron los conflictos solucionados así como los no solucionados, para los conflictos que no se solucionaron se presentan alternativas.

Capítulo 7. Conclusiones y trabajos futuros

La conclusión y aportación principal de este trabajo de tesis es una estrategia para integrar bases de datos basada en el registro y extracción de correspondencias semánticas.

Esta estrategia propone que el proceso de integración de bases de datos se realice en las siguientes tres etapas:

1. Crear un GS tomando como entrada el conjunto de esquemas de las bases de datos locales.
2. Registrar correspondencias semánticas en un GDD.
3. Proporcionar acceso a las correspondencias semánticas mediante componentes de software para la solución de la heterogeneidad semántica.

Cada una estas etapas ha sido aplicada a un caso de estudio y un prototipo para integrar bases de datos, donde se abordan los diferentes casos y conflictos que se pueden presentar en proceso de integración de bases de datos. De esta manera, los objetivos de esta tesis han sido alcanzados y han resultado en las siguientes aportaciones:

- Una estrategia para integrar bases de datos;
- Un método intuitivo para la creación de un GS;
- Un modelo de datos de un GDD para describir el GS y las correspondencias semánticas entre el GS y el conjunto de bases de datos locales;
- Una arquitectura de componentes de software para la solución de la heterogeneidad semántica;
- La solución de conflictos de esquema mediante el uso de correspondencias semánticas;
- La solución de conflictos de datos mediante reglas de transformación codificadas en un componente de software.

El desarrollo de esta tesis muestra que la heterogeneidad semántica entre bases de datos puede ser a nivel de tabla, atributo y datos. A nivel de tabla y atributo, es posible capturar las relaciones mediante un GDD. A nivel de datos, es posible solucionar la heterogeneidad semántica utilizando reglas de transformación que se pueden capturar mediante código.

En este trabajo de tesis también se muestran algunas de las diferentes herramientas comerciales que se pueden utilizar para integrar bases de datos.

Algunas de las lecciones aprendidas en este trabajo de tesis son las siguientes:

- Conocer las necesidades y problemas de integración de bases de datos de las empresas.
- Conocer las soluciones comerciales y no comerciales para integrar bases de datos.
- Es posible solucionar la heterogeneidad semántica mediante componentes de software.
- UML facilita el diseño de componentes de software.
- Java y RMI son herramientas que facilitan la creación de arquitecturas de objetos distribuidos.
- El GS representa la unión del conjunto de bases de datos a integrar.
- El empleo de un GS reduce los conflictos entre n-bases de datos a conflictos binarios, es decir entre dos bases de datos: el GS y la base de datos local.
- Es posible solucionar los conflictos de datos mediante reglas globales de transformación.
- Las correspondencias semánticas ayudan a transformar una operación SQL global en sub operaciones para ejecutarse en las bases de datos locales.

Algunas líneas de trabajo futuro que pueden realizarse son las siguientes.

A corto plazo:

- Desarrollar una herramienta que permitan a usuarios globales capturar la heterogeneidad semántica. En esta tesis se capturaron las correspondencias semánticas directamente en el GDD, sin embargo, es posible facilitar el proceso de registro de correspondencias semánticas con una aplicación que permita capturar estas correspondencias.
- Desarrollo de nuevas reglas de transformación que permitan solucionar conflictos de datos. En esta tesis sólo se capturó una regla de transformación que convierte de dólares a pesos, sin embargo es posible agregar nuevas reglas al componente GTR.

A largo plazo:

- Desarrollar una herramienta que permita extraer de manera automática las reglas de heterogeneidad semántica de un conjunto de bases de datos. En esta tesis se creó el GS y se registraron las correspondencias semánticas de manera manual, sin embargo, obtener las reglas de heterogeneidad semántica de manera automática es un área de oportunidad muy importante que puede facilitar el proceso de integración de bases de datos enormemente.
- Agregar la capacidad al esquema global de auto actualizarse dependiendo de los cambios en las bases de datos locales. Las bases de datos locales constantemente cambian y estos cambios deben ser reflejados en el GS, la actualización automática de este esquema es uno de los principales retos que existen al integrar bases de datos mediante un esquema global.

Desarrollar una herramienta para capturar la heterogeneidad semántica y agregar nuevas reglas de transformación manualmente es un trabajo que no requiere investigación

adicional. Sin embargo, capturar la heterogeneidad semántica de manera automática es un trabajo que requiere una gran investigación, ya que no existen reglas precisas para conocer cúando dos entidades, ya sean tablas, atributos o datos son semánticamente equivalentes.

Apéndices

A. Acrónimos

- **FDBMS**: Sistema manejador de bases de datos federado (*Federated Data Base Management System*). Es una colección de DBMS´s autónomos y posiblemente heterogéneos cooperando entre sí. (Sección 2.3).
- **GDD:** Diccionario de datos global (*Global Data Diccionary*): Este componente tiene doble función, la primera es describir el GIS (Global Integrated Schema). El GIS es el esquema global resultante de la metodología presentada en el capítulo 4. La segunda función del GDD es la descripción de las correspondencias semánticas entre el GIS con cada una de las bases de datos locales. (Sección 5.3).
- **GS**: Esquema Global (*Global Schema*). Es una vista virtual de la unión de un conjunto de bases de datos (Sección 5.3).
- **GQT:** Transformador de Consultas Global (*Global Query Transformer*): Este es el componente central de la arquitectura y tiene las funciones de recibir las operaciones ejecutadas de manera global, transformarlas, ejecutarlas en los servidores LDBMS-Server y regresar los resultados a los usuarios globales. (Sección 5.3).
- **GSA:** Analizador Semántico Global (*Global Semantic Analizer*): Este componente tiene la función de extraer la información semántica requerida por el GQT para hacer las transformaciones de las operaciones SQL realizadas globalmente. (Sección 5.3).
- **GTR:** Reglas Globales de Transformación (*Global Transformation Rules*): Este componente contiene las reglas globales de transformación, las cuales son utilizadas por el componente GQT para solucionar los conflictos de tipos y unidades de datos.
- **LCS**: Esquema local conceptual (*Local Conceptual Schema):* Es el esquema conceptual de una base de datos local. (Sección 5.3).
- **LDB:** Base de datos local (*Local Database*): Es una base de datos local. (Sección 5.3).
- **LDBMS-Server:** Sistema Manejador de bases de datos local (*Local DBMS*). Es el DBMS para una base de datos local. (Sección 5.3).
- **LES**: Esquema local externo (*Local External Schema*).Es una vista del LCS, a la cual un usuario global puede acceder. (Sección 5.3).
- **MDBMS**: Sistema manejador de bases de datos múltiples (*Multi Data Base Management System*). Soporta la operación de varios sistemas de bases de datos donde cada sistema de base de datos es manipulado por un sistema de base de datos distribuido (Sección 2.3).

Referencias

[Anahory and Murray, 1997] S. Anahory, D. Murray. *Data warehousing in the real world*: A practical guide for building decision support systems, Addison-Wesley, England, 1997.

[Álvarez, 2003] F. Álvarez. *Distribución de datos para bases de datos distribuidas, una arquitectura basada en componentes de software*. Tesis de maestría, ITESM, Campus Monterrey, 2003.

[Batini et. al., 1986] C. Batini, M. Lenerini, S. B. Navathe: *A comparative analysis of methodologies for database schema integration.* ACM Computing Surveys, Vol.18, No. 4 pp 323 - 364, December 1986.

[Bright, 1994] M. Bright, A. Hurson, S. Pakzad: *Automated resolution of semantic heterogeneity in multidatabases*. ACM Transactions on Database Systems, Vol. 19, No. 2, June 1994.

[Bergamaschi, 1999] S. Bergamaschi, S. Castano, M. Vincini: *Semantic integration of semistructured and structured data sources*. ACM SIGMOD Record, Vol 28, No. 1, pp 54-59, March 1999.

[Bouguettaya et. al., 1999] A. Bouguettaya, B. Benatallah, M. Ouzzani, and L. Hendra: *Using Java and CORBA for implementing Internet databases.* Proceedings of the 15th International Conference on Data Engineering, pp 218 –227, 1999.

[Chung and Mah, 1995] S. Chung and P. Mah: *Schema integration for multidatabases using the unified relational and object-oriented model.* Proceedings of the 1995 ACM 23rd annual conference on computer science , Nashville, Tennessee, United States, 1995.

[Grant et. al., 1993] J. Grant, W. Litwin, N. Roussopoulos, and T. Sellis: *Query languages for Relational Multidatabases.* The Int'l Journal on Very Large Data Bases, pp 53-171, April 1993.

[Gray, 2004] J. Gray: *The next database revolution.* Proceedings of the 2004 ACM SIGMOD international conference on Management of data. Paris, France, 2004.

[Haas and Lin, 2002] L. Haas and E. Lin: *IBM Federated Database Technology,* IBM Corporation URL: http://www-106.ibm.com/developerworks/db2/library/techarticle/0203haas/0203haas.html Access date: October 2004.

[Hayes, 2003] Holly Hayes: *DB2 Information Integration: The Big Picture,* IBM Corporation URL: http://www7b.software.ibm.com/dmdd/zones/db2ii/bigpicture.html, July, 2003. Access date: August 2003.

[Hakimpour and Geppert, 2001] F. Hakimpour, A. Geppert: *Resolving semantic heterogeneity in schema integration*, ACM Press New York, NY, USA, Proceedings of the international conference on Formal Ontology in Information Systems, Ogunquit, Maine, United States, pp 297-308, October 17, 2001.

[Hull and Zhou, 1996] R. Hull. and G. Zhou: *A framework for supporting data integration using the materialized and virtual approaches.* ACM Press New York, NY, USA, International Conference on Management of Data, Montreal, Quebec, Canada, pp 481-492, 1996.

[Inmon, 1996] W.H. Inmon: *Building the Data Warehouse*, John Willey, New York, United States, 1996.

[Johannesson, 1994] P.Johannesson: *Linguistic instruments and qualitative reasoning for schema integration,* Proceedings of the third international conference on Information and knowledge management, Gaithersburg, Maryland, United States, pp 252 - 262, 1994.

[Johnson, 1999] A. Johnson: *Data Warehousing*, Computer world, United States, December 6, 1999.

[Kim and Seo, 1991] W. Kim y J. Seo: *Classifying schematic and data heterogeneity in multidatabase systems* .Computer Magazine, Vol. 24, No.12, pp 12 –18, December, 1991.

[Lakshmanan et. al., 2001] L. Lakshmanan, F. Sadri and Subbu N. Subramanian: *SchemaSQL: An extension to SQL for multidatabase interoperability*, ACM Transactions on Database Systems, Vol. 26, No. 4, pp 476 – 519, December, 2001.

[Lee and Baik, 1999] J. Lee and D. Baik: *SemQL: Semantic Query Language for Multidatabase Systems,* ACM Press New York, NY, USA, Proceedings of the eighth international conference on Information and knowledge management , Kansas City, Missouri, United States , pp 259 – 266, 1999.

[Linthicum, 2000] D. Linthincum: *Enterprise Application Integration*, Addison-Wesley Information Technology Series, United States, 3rd edition, May, 2000.

[Litwin, 1984] W. Litwin: *A relational multidatabase manipulation language.* 1st IEEE Conference on Data Engineering, Los Angeles, California, United States, February 1984.

[Litwin, 1998] W. Litwin: *From Database Systems to Multidatabase Systems*: Why and How. Proceeding of the 6th British National Conference on Databases (BNCOD6), pp 161-188, 1998.

[Litwin et. al., 1989] W. Litwin, A. Abdellatif, A. Zeroual, B. Nicolas: *MSQL: A multidatabase language. Information Sciences*, pp 59-101, December, 1989.

[Litwin and Abdellatif, 1986] W. y A. Abdellatif: *Multidatabase Interoperability,* IEEE Computer, Vol. 19, No. 12, December, 1986.

[Litwin et. al., 1982] W. Litwin, J.Boudenant, C. Esculier, A. Ferrier, A. Glorieux, J. La Chimia, B., K. Kab, C. Moulinoux, P. Rolin and C. Stangret*: SIRIUS Systems for Distributed Data Management, Distributed Databases*, pp. 311-36, Amsterdam, North-Holland Publishing Company, 1982.

[Litwin et. al., 1990] W. Litwin, L. Mark and N. Roussopoulos*: Interoperability of Multiple Autonomous Databases.* ACM Computing Surveys, pp 267 - 293 , Vol. 22, No. 3, September, 1990.

[Microsoft, 1996] *DCOM Technical Overview. Microsoft* URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomtec.asp, Access date: November 2004.

[Microsoft, 2003] *Defining the Basic Elements of .NET.* Microsoft URL: http://www.microsoft.com/net/basics/whatis.asp, Access date: July 2003.

[Microsoft, 2004] *Business Intelligence and Data Warehousing in SQL Server 2005,* Microsoft URL: http://www.microsoft.com/technet/prodtechnol/sql/2005/evaluate/dwsqlsy.mspx, Access date: November 2004.

[Özsu and Valduriez, 1999] T. Özsu and P. Valduriez: *Principles Of Distributed Database Systems*, Prentice Hall, 1999.

[Oracle, 2003] *Oracle9i Warehouse Builder, Architectural White Paper,* Oracle URL: http://www.oracle.com/ip/develop/ids/index.html?owb.html, Access date: July 2003.

[Parent and Spaccapietra, 1998] C. Parent and S. Spaccapietra: *Issues and Approaches of Database Integration.* Communications of the ACM, Vol 41, No 5, pp 166-178, May, 1998.

[Rishe et. al., 2000] N. Rishe, R. Athauda, J. Yuan and S. Chen: *Semantic Relations: The Key to Integrating and Query Processing in Heterogeneous Databases*, Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, pp.717-722, 2000.

[Seligman and Rosenthal, 2001] L. Seligman and Arnold Rosenthal: *XML's Impact on Databases and Data Sharing*, IEEE Computer Society, Vol. 34, No 6, pp 59- 67, June, 2001.

[Sheth and Larson, 1990] A. Sheth. and J. Larson: *A Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases.* ACM Computing Surveys, Vol 22, No. 3, pp 183- 236 , September, 1990.

[Shoffner, 2000] Shoffner, Michael: *Write your own MOM!*, JavaWorld URL:http://www.javaworld.com/javaworld/jw-05-1998/jw-05-step_p.html. Access date: November 2003

[Sun, 2002] *Java remothe method invocation distributed computing for Java*, Sun Microsystems URL: http://java.sun.com/marketing/collateral/javarmi.html, 2002. Access date: November, 2003

[Sybase, 2001], *Sybase Data Integration: Analyzing your options*, Sybase URL: http://www.sybase.com/content/1016941/4339DataIntegrationv8.pdf, 2001. Access date: July , 2003.

[W3C, 2004] *Web Services Arquitecture,* W3C URL: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/. 2004. Access date: October, 2004.

[Xuequn, 1999] W. Xuequn, *A CORBA-based architecture for integrating distributed and heterogeneous databases*, Fifth IEEE International Conference on Engineering of Complex Computer Systems, ICECCS '99. Las Vegas, NV, United States, pp 143- 152, October, 1999.