Gestor de Consultas Distribuidas Heterogéneas

Gonzalo Mena-Mendoza

Universidad Autónoma de Querétaro,
Ciudad Universitaria, Querétaro, Qro., México
mena@computer.org
http://mena.com.mx/gonzalo/maestria/bd/

Resumen. Se presenta un sistema que proporciona los fundamentos para la vista unificada de múltiples bases de datos relacionales, distribuidas y heterogéneas. La presencia de tablas remotas es completamente transparente para el usuario, quien las percibe simplemente como si se tratase de tablas de un catálogo local. Se describe la estrategia principal utilizada para la realización de reuniones sobre tablas distribuidas. Se menciona también la arquitectura y la organización de los componentes desarrollados para crear el sistema.

1 Introducción

Este documento describe un sistema que proporciona los fundamentos para la vista unificada de múltiples bases de datos relacionales, distribuidas y heterogéneas. Los programas clientes acceden a las tablas remotas diferenciándolas de las locales mediante prefijos definidos por el usuario. La presencia de tablas remotas es completamente transparente para el usuario, quien las percibe simplemente como si se tratase de tablas de otro esquema o catálogo local.

1.1 Arquitectura

El diagrama 1 muestra la arquitectura del sistema. El program está escrita en Java y utiliza el estándar JDBC para lograr la conexión a las bases de datos, locales y remotas. Este último sirve tanto para realizar la conexión, ejecutar instrucciones y regresar los resultados, como para obtener los metadatos necesarios.

El sistema cuenta con dos componentes principales: el primero es un programa interactivo en modo terminal que recibe instrucciones del usuario y muestra los resultados de manera tabular. El segundo, procesa las instrucciones, realiza las consultas en las distintas bases de datos involucradas, reúne los resultados y los regresa al cliente de manera unificada.

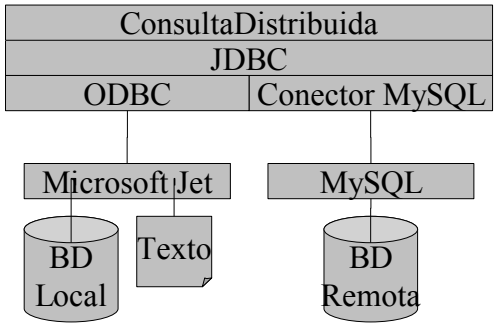

Diagrama 1: bloques del sistema

2 Esquema de Ejemplo

Supóngase que un ingeniero de una compañía tiene acceso de manera remota a la base de datos que contiene el directorio de todos los empleados. Por razones de seguridad solamente cuenta con permisos de lectura, así que no puede modificar el directorio de empleados ni el catálogo de proyectos de la empresa, ni añadir nuevas tablas a dicha base de datos.

Por otra parte el ingeniero desea mantener información de índole personal sobre los compañeros con los que labora habitualmente, por ejemplo, quizás le sea más cómodo referirse a dichas personas por apodo. También puede necesitar mantener una lista de tareas por realizar por proyecto, cada una en conjunto con otra compañero de trabajo. Por privacidad el ingeniero desea mantener esta información en su computadora personal.

El diagrama 2 muestra las relaciones entre las tablas locales y remotas que corresponden al ejemplo.

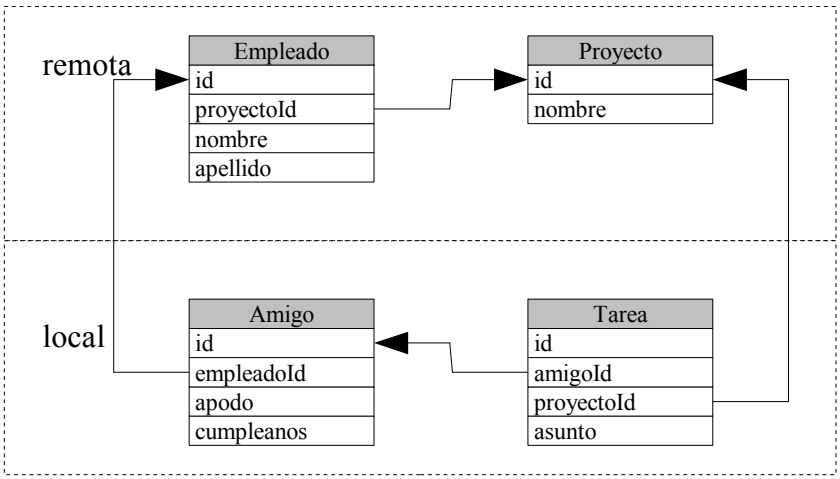

Diagrama 2: modelo entidad-relación

Para este hipotético usuario sería muy práctico poder consultar la información de ambas bases de datos, la corporativa remota y la personal local, de manera unificada y transparente. Por ejemplo, utilizando el prefijo `ext.` para diferenciar las tablas remotas de las locales, se desearía pode la consulta representada por la instrucción 1:

```
SELECT Amigo.apodo, ext.Empleado.apellido
  FROM Amigo, ext.Empleado
 WHERE Amigo.empleadoId = ext.Empleado.id
   AND ext.Empleado.apellido = 'Pérez'
```
(1)

3 Procesamiento de Consultas

Dado que las operaciones de inserción y de actualización se realizan casi en su totalidad sobre tablas individuales, el caso más importante de instrucciones que involucran tablas distribuidas lo presentan las consultas. En muchas aplicaciones las semirreuniones de dos o más tablas son probablemente las consultas más comunes. Se presenta un reto cuando las tablas involucradas se encuentran en múltiples bases de datos, posiblemente heterogéneas.

El procesamiento de dichas consultas se basó en la estrategia de semirreunión descrita por Silberschatz [1]. En términos de SQL, la transformación de las consultas se realiza en los tres pasos principales descritos a continuación, siguiendo con el ejemplo de la instrucción 1.

3.1 Extracción de Consulta Remota

El primer paso consiste en descomponer la consulta original en dos partes, una local y una remota en base a las tablas empleadas. La consulta remota correspondiente a la instrucción 1 se muestra a continuación:

```
SELECT Empleado.apellido, Empleado.id
  FROM Empleado
 WHERE Empleado.apellido = 'Pérez'
```
(2)

Puede verse que además de la columna `Empleado.apellido`, presente en la consulta original, es necesario incluir también `Empleado.id` ya que será utilizada para realizar la reunión con la tabla `Amigo`.

3.2 Creación de Tabla Temporal Local

Una vez ejecutada la instrucción 2, se utilizan los metadatos del juego de resultados obtenido para crear una tabla temporal que contenga representaciones de todas las columnas recibidas, con el tipo y tamaño correcto. Nótese que es necesario incluir el

nombre de la tabla de procedencia ya que puede haber más de una columna con el mismo nombre, esto es muy factible en el caso de múltiples reuniones. La siguiente instrucción muestra la definición de tabla necesaria:

```
CREATE TEMPORARY TABLE TEMP (
    Empleado_apellido VARCHAR(255),
    Empleado_id INT)
```
(3)

A continuación se utiliza una instrucción parametrizada para insertar en la tabla temporal cada una de las filas obtenidas en la instrucción 2.

```
INSERT INTO TEMP (Empleado_apellido, Empleado_id)
    VALUES (?, ?)
```
(4)

3.3 Reunión Local de Resultados

Por último se efectúa localmente le reunión de la parte local de la consulta original con la tabla temporal que contiene los resultados remotos:

```
SELECT Amigo.apodo
     , TEMP.Empleado_apellido AS apellido
  FROM Amigo, TEMP
 WHERE Amigo.empleadoId = TEMP.Empleado_id
```
(5)

Aquí es importante regresar a las columnas resultantes el nombre original empleado por el usuario, para ello se utilizan alias.

4 Procesamiento de Otras Instrucciones

El procesamiento de otros tipos de instrucciones como la definición, inserción y actualización de tablas (`CREATE TABLE`, `DROP TABLE`, `UPDATE`, `INSERT`) es más sencillo ya que comunmente afectan a una sola tabla. Por lo tanto es suficiente determinar, por medio del prefijo utilizado, si se trata de una tabla local o remota y por lo tanto se puede conectar a la base de datos en cuestión y comprometer directamente la instrucción.

5 Implementación

Se escribió en lenguaje Java un programa que implementa la estrategia anteriormente descrita. El código esté organizado en dos clases: `ConsultaDistribuida` constituye el programa interactivo en modo terminal que recibe las instrucciones del usuario y muestra los resultados de manera tabular por medio de un método basado en el implementado por Flanagan [2].

`ResultadoDistribuido` es llamado para procesar las instrucciones, realizar las consultas en las distintas bases de datos involucradas, reúnir los resultados y

regresarlos al cliente de manera unificada. Está modelado como una combinación de las interfaces `ResultSet` y `Statement` del paquete `java.sql`. El análisis de las instrucciones se efectua mediante expresiones regulares.

A continuación se muestra un ejemplo de la ejecución del programa y del procesamiento de una reunión más compleja que la instrucción 1:

```
C:> java ConsultaDistribuida odbc-odbc.conf

Consulta Distribuida
--------------------
Local:  url=jdbc:odbc:amigos
Remota: url=jdbc:odbc:empleados    prefijo=ext

sql> SELECT Amigo.apodo, ext.Empleado.apellido, ext.Proyecto.nombre,
Tarea.asunto FROM Amigo, ext.Empleado, ext.Proyecto, Tarea WHERE
Amigo.empleadoId = ext.Empleado.id AND ext.Empleado.proyectoId =
ext.Proyecto.id AND ext.Proyecto.id = Tarea.proyectoId

---- remota ----

SELECT Empleado.apellido, Proyecto.nombre, Empleado.id, Proyecto.id
  FROM Empleado, Proyecto
 WHERE Empleado.proyectoId = Proyecto.id

---- local ----

DROP TABLE TEMP

CREATE TABLE TEMP (Empleado_apellido VARCHAR(255), Proyecto_nombre
VARCHAR(255), Empleado_id int, Proyecto_id int)

INSERT INTO TEMP (Empleado_apellido, Proyecto_nombre, Empleado_id,
Proyecto_id) VALUES (?, ?, ?, ?)

---- local ----

SELECT Amigo.apodo, TEMP.Empleado_apellido AS apellido, TEMP.Proyecto_nombre
AS nombre, Tarea.asunto
  FROM Amigo, TEMP, Tarea
 WHERE Amigo.empleadoId = TEMP.Empleado_id
   AND TEMP.Proyecto_id = Tarea.proyectoId

+---------+----------+--------+------------+
|  apodo  | apellido | nombre |   asunto   |
+---------+----------+--------+------------+
| Chucho  | Pérez    | Alfa   | requisitos |
| Pancho  | López    | Beta   | diseño     |
| Pancho  | López    | Beta   | requisitos |
+---------+----------+--------+------------+
realizado en 0.062 s.

sql>
```

5.1 Limitaciones

La implementación actual presenta algunas limitaciones en el tipo de consultas que es posible realizar, en comparación con la totalidad disponible en cada gestor de base de datos utilizado. Algunas de estas limitaciones son inherentes a la necesidad de emplear funcionalidad común a todos los gestores de bases de datos.

Otras limitaciones se derivan de la decisión de no obscurecer el diseño básico y la estrategia principal de procesamiento de consultas. La limitación más notoria es la restricción de las condiciones de una consulta a una conjunción múltiple (únicamente pueden separarse por el operador `AND`). Además están la carencia de alias para las tablas (y por lo tanto la autounión), el manejo de funciones, reuniones externas, agrupación y ordenación.

6 Trabajo Futuro

La investigación y desarrollo llevados a cabo hasta el momento dan lugar a numerosas oportunidades. Las más obvias se encuentran en la eliminación de las limitaciones en la sintaxis de las instrucciones anteriormente descritas. La extensión del diseño actual para utilizar un número arbitrario de bases de datos es muy fácil de realizar. También puede trabajarse en la optimización de las estrategias de consulta a partir de las estadísticas de las tablas así como de las velocidades de conexión hacia las bases de datos remotas.

Un proyecto más ambicioso sería el de implementar estas ideas como un manejador de JDBC que hiciera transparente para el cliente la utilización unificada de múltiples bases de datos. Para ello sería necesario implementar todas las interfaces definidas por el API de JDBC.

7 Conclusiones

Se desarrolló un sistema que proporciona una vista unificada de múltiples bases de datos relacionales, distribuidas y heterogéneas. En particular, se describió la estrategia para la realización de reuniones entre tablas ubicadas en distintas bases de datos. Para ello se desarrollaron herramientas que facilitan la implementación de este tipo de estrategias. Este trabajo puede servir de base para futuros desarrollos relacionados con la realización de consultas distribuida.

Referencias

1. Silberschatz, A., Korth H.F., Sudarshan, S.: Fundamentos de Bases de Datos. 4ta. edn. McGraw-Hill, Madrid (2002) 480-483
2. Flanagan, D.: Java Examples in a Nutshell. O'Reilly, Sebastopol (1997) 329-330