Visualiz eas

María ni

e enfrentar, está ganando
stos espacios de informa-
años p gran variedad de autores

tuitivas y técnicas de navegación es esencial para permitir
espacios de información complejos nos referimos a espacios
y grandes sino además, heterogéneos en su estructura. Los
s se caracterizan, por ejemplo, por información que relaciona
lo, tablas, gráficos y video) y que difiere en su dimensión.
e información debe manejar una interacción con los datos que
til del conjunto total de datos disponibles. La exploración de
aspectos:

acción para moverse (browse) en el espacio de información y buscar

nformación que vuelca gráficamente un espacio de información o re-
da

alisis para facilitar la interpretación de los datos retornados como resul-
en el proceso de búsqueda.

e la información se refiere a cómo obtener información con ciertas carac-
consulta que el usuario puede formular es altamente dependiente de modelo
nformación subyacente, en este caso los datos una base de datos. Los usua-
puestas necesidades específicas deben interactuar con los datos para precisar
y qué no desde su punto de vista. Para que sea útil, esta interacción debe
n todos los niveles del proceso de visualización.
lenguajes de consulta para bases de datos relacionales deben moverse hacia una
d, permitiendo más potencia en la especificación de la consulta. Actualmente,
nvestigación importante lo constituyen los lenguajes de consulta visuales. Las

Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)
e financiado por la Secretaría de Ciencia y Tecnología (Universidad Nacional del Sur).

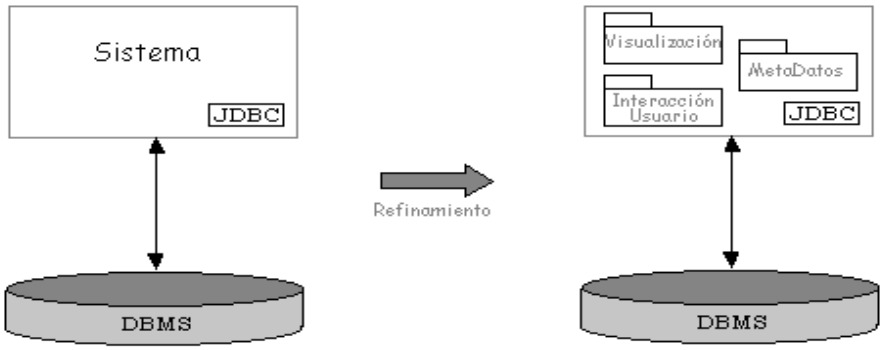

Figura 1: Arquitectura *two-tier*

metáforas visuales pueden ayudar a los usuarios no expertos a formular consultas booleanas complejas.

Cuando una persona desea obtener determinada información de una base de datos, tiene una o más metas en mente y usa un sistema de búsqueda como la herramienta que le brinde la posibilidad de alcanzar esas metas. Estas metas pueden requerir información de muy amplio espectro: el resultado de una consulta puede mostrarse en forma *precisa* o *imprecisa*, de modo tal que al visualizarlos posteriormente de dicha búsqueda se pueda tener la posibilidad de observar diferencias. Así, en este proceso interactivo de especificación de la consulta, visualización de los resultados obtenidos de la búsqueda y examen de los mismos, la información es presentada al usuario ya sea al finalizar el proceso o en la reformulación de la consulta para alcanzar la meta deseada.

Las búsquedas por palabras clave, los lenguajes de consulta y las consultas guiadas por ejemplos son parte de la exploración en las bases de datos. En particular, últimamente han adquirido relevancia las consultas dinámicas debido a su efectividad; estas consultas son especificadas mediante barras de desplazamiento que definen intervalos de interés en los atributos. Este tipo de consulta permite una integración natural de la búsqueda y la visualización de los resultados. El proceso de visualización tendiente a realizar una exploración visual de los datos necesita integrarse con los sistemas utilizados para manipular grandes volúmenes de información relacional y estructurada, incluyendo la conexión con sistemas de bases de datos relacionales. El primer paso de este proceso lo constituye la extracción de los datos a visualizar desde de la base de datos.

En este trabajo se presenta un proyecto cuyo objetivo es diseñar un sistema que realice la transformación de los datos tabulares de una base de datos relacional en un tabla de datos general, de modo que permita su posterior visualización. El sistema debe posibilitar al usuario que seleccione, según su criterio, algunos de los atributos presentes en una base de datos para generar una nueva tabla a partir de la cual se pueda visualizar sólo la información que le resulta relevante. De esta manera el sistema en cuestión deberá establecer una conexión con cualquier base de datos relacional, no importando su estructura ni sus contenidos, obtener información sobre su conformación, y a partir de ella solicitar al usuario el grupo información que le resulta interesante, a fin de que el sistema elabore una tabla interna, o *tabla de metadatos*, con la información concreta de la base de datos para su posterior visualización.

2 Arquitectura del sistema

El esquema general del sistema se puede observar en la figura 2. Las componentes intervinientes son la base de datos con los datos a ser visualizados, y la aplicación de visualización en sí. Esta aplicación esta formada por un módulo de interacción con el usuario, que se encarga de mostrarle al usuario la estructura de la base de datos y le permite a este elegir cuales son las tablas de su

interés e inclusive las columnas que considera relevantes para el proceso de visualización; otro módulo con el proceso de visualización propiamente dicho, que toma la tabla con los metadatos que fueron escogidos por el usario y se encarga de mostrarselos graficamente; y un tercer módulo encargado de recuperar la estructura original de la base datos y genera la tabla de metados según los requerimientos del usuario generado por el módulo de interface.

Para la implementación de este sistema se escogió el lenguaje de programación Java, con el objetivo de aprovechar su portabilidad y la disposición de librerías de acceso a diversas bases de datos. Por esta razón el último de los módulos descriptos debe usar la API JDBC [WFC+99]. La implementación que se está llevando a cabo actualmente sólo utiliza la conexión ODBC de Windows, pero la arquitectura está diseñada para poder extenderse a otros protocolos de acceso a bases de datos.

2.1 JDBC

[WFC+99]

JDBC es una API Java que permite el acceso virtual a cualquier tipo de dato tabular. Esta API consiste en un grupo de clases e interfaces escritas en el lenguaje de programación Java y que a su vez proveen una API estándar para desarrollar herramientas para bases de datos y hace posible aplicaciones sobre estas bases de datos. La manera en la que se logra este objetivo se reduce básicamente al envío de consultas SQL a la base de datos relacional concreta; para que esto sea posible debe soportar todos los posibles dialectos de SQL. Algunas versiones de JDBC también brindan la posibilidad de interactuar con diferentes fuentes de datos. De esta manera una aplicación puede acceder virtualmente a cualquier fuente de datos y correr bajo cualquier plataforma que contenga una máquina virtual Java; es decir que utilizando esta API, el programador no se ve en la necesidad de escribir un programa diferente para el acceso a datos en diferentes plataformas, sino que simplemente escribe un único programa utilizando API JDBC; el programa tiene entonces la capacidad de enviar las consultas de manera apropiada a la fuente de datos. La combinación de la plataforma Java y la API permite al programados abstraerse de la arquitectura subyacente, dándole el lema de programación "escriba una vez, ejecute donde quiera". La funcionalidad de la API, puede resumirse en los siguientes tres pasos:

1. Establecer una conexión con una fuente de datos.
2. Enviarle consultas y sentencias de actualización a esa fuente de datos.
3. Procesar el resultado.

Esta API Java que se está describiendo se adecuá perfectamente a las condiciones establecidas por el sistema que se pretende desarrollar. Esto implica que la API logra establecer el proceso de comunicación y obtención de datos vía ODBC de manera mucho más adecuada y segura en el sistema concreto.

Se podría estar tentado en utilizar ODBC; el problema de esta alternativa es que la interface de ODBC está implementada en el lenguaje C y las llamadas desde Java a código C provocan un decremento de algunas propiedades importantes como la seguridad y la portabilidad de las aplicaciones. La traducción literal del código fuente del ODBC a un API Java tampoco es una buena solución ya que la traducción de algunas instrucciones C a Java no son triviales. Por ejemplo, aquellas que utilicen punteros, estructura de datos que no existe como tal en Java. El otro punto a favor del uso de un API puramente Java radica en el hecho de que la aplicación completa es totalmente portable y segura ya que puede pasar a ejecutarse en cualquier máquina que tenga una máquina virtual Java.

Un aspecto que es de suma importancia, es la capacidad de manipular metadatos, *i.e.* datos sobre la estructura de la base de datos como los tipos de las columnas de las tablas; esta información es imprescindible para el desarrollo del sistema, más específicamente en la construcción

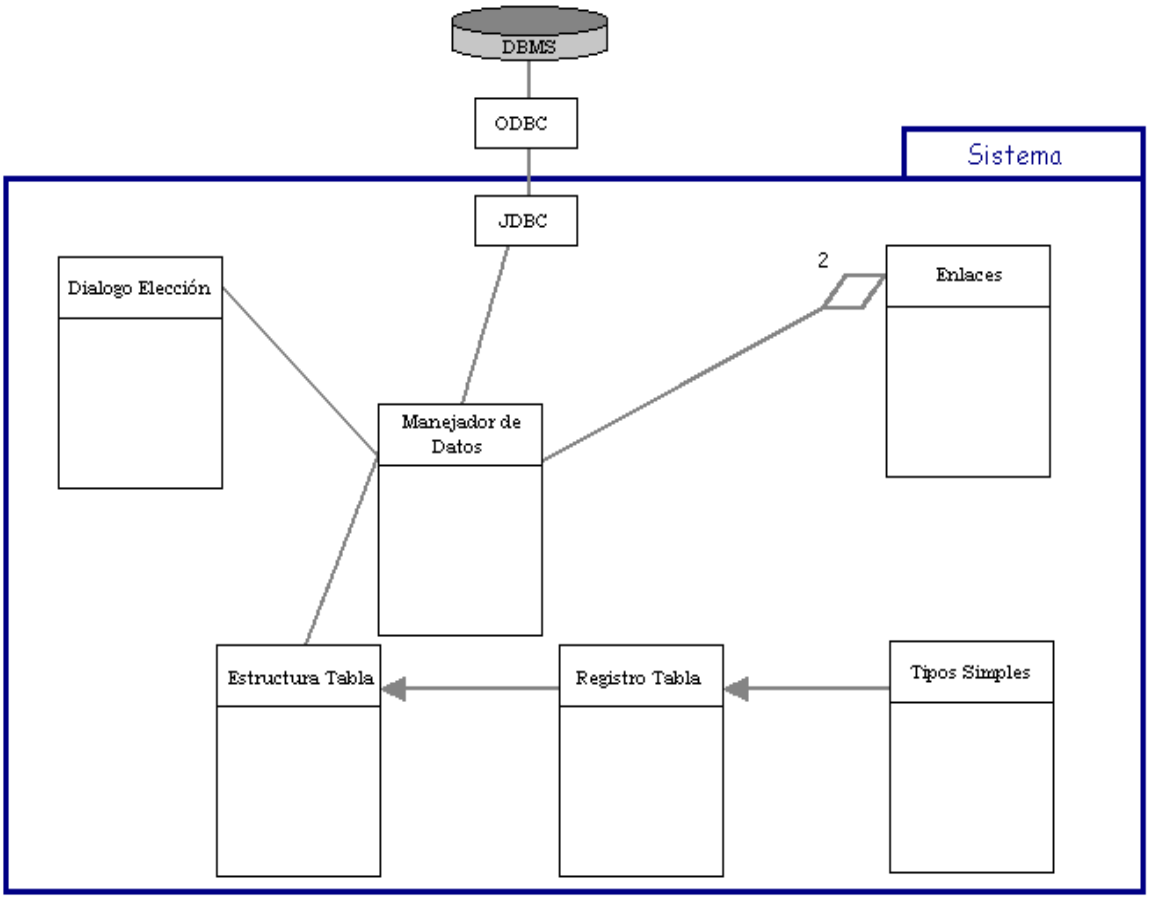

Figura 2: Diagrama de clases del sistema

de la tabla interna previamente mencionada. Esto no sólo brinda una manera de manipular la información de los datos almacenados en la base de datos sino también una manera de conocer datos sobre la estructura de la base. La mayoría de las bases de datos están organizadas en tablas; para este tipo de aplicaciones es muy importante poder recuperar información sobre la base de datos, como la cantidad de tablas de la base de datos, las columnas de cada tabla, el tipo de datos y nombre de cada columna, las claves primarias, etc. Cada tipo de manejador de base de datos, DBMS, tiene su propia funcionalidad para otorgar este tipo de información. El API JDBC provee una interface que es implementada mediante un driver de manera que los métodos devuelvan información sobre el driver y/o el manejador de datos para el cual el driver fue escrito. Esta interface brinda a los usuarios y a las herramientas una forma estándar de obtener metadatos.

2.2 Estructura de la aplicación servidora

En este caso, la idea es empezar realizando una aplicación que se adapte a la forma *two-tier* propuesta. A pesar de que puede resultar claro la utilidad de transformar más adelante el sistema en *three-tier*, a fin de que resulte más general y modular.

En principio el sistema tendría la estructura que se observa en la figura 2.2.

De acuerdo al diagrama de clases, el sistema cuenta con las siguientes clases:

API JDBC: Es el API Java que fue mencionado en la sección 2.1 y que llevará a cabo la conexión con la base de datos a través de ODBC.

Manejador de Datos: Es quien mantendrá la interacción con JDBC, indicándole que datos pedirle a la base de datos, estos datos pueden ser datos concretos o bien metadatos. A

su vez en su funcionalidad incluye el pasar la información primero a la Clase Diálogo Elección y mas adelante a la clase Estructura Tabla para formar la estructura que haya sido indicada por el usuario.

Diálogo Elección: Esta interacción con el usuario, indicará al manejador de datos. cuales datos de cada tabla disponible guardar en la estructura interna dada por las clases siguientes.

Estructura Tabla: Esta clase guarda la forma de una tabla particular, como la tabla esta formada por registros, conoce a la clase Registro Tabla a pesar de que esta última no conoce la existencia de esta clase.

Registro Tabla: Esta clase representa a cada uno de los registros de una tabla particular. Requiere del conocimiento de la clase Tipos Simples ya que necesita los métodos necesarios para su manipulación.

Tipos Simples: Son los tipos básicos que puede contener la tabla. Esta clase no tiene conciencia de la existencia de las otras clases a pesar de que es conocida desde la clase Registro Tabla.

Enlaces: Esta relacionada a travesee de una agregación con la clase Estructura Tabla. Su función es guardar la relación que pueda existir entre dos tablas diferentes de la base de datos, guardando a su vez el campo que las mantiene enlazadas.

La base de datos o DBMS, pude variar de una base Access hasta una Oracle, pasando por cualquier otro tipo existente. Por la naturaleza de la implementación esta funcionará sólo sobre el sistema operativo Windows, ya que la interacción está desarrollada para motores que se comprendan con ODBC.

La elección de Java como lenguaje de implementación está fundamentada en las características del lenguaje, es decir, en el hecho de que Java es un lenguaje de programación robusto, seguro, fácil de utilizar y entender y downlodable de los entornos de red; estas características lo hacen un buen lenguaje base para aplicaciones sobre bases de datos.

Una posible futura extensión de este sistema sería pasar el modelo de una arquitectura *two-tier* a una *three-tier*, obteniendo así una estructura en la que se cuenta con una aplicación servidora que establece el contacto con la base de datos y construye la tabla de metadatos, solicitada por los usarios a través de navegadores o aplicaciones remotas. Estas aplicaciones remotas cumplirian el papel de clientes, y conforman la tercera parte de la arquitectura. Las implicacias de esta decisión serían que la aplicación en la nueva arquitectura no contendría ninguna interacción con el usuario de manera directa; como así tampoco estaría involucrada en el proceso de visualización de manera concreta, solo provee los datos de la base solicitados en forma de una estructura Java.

Referencias

[BYRN99] Ricardo Baeza-Yate and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[CMS99] Stuart Card, John Mackinlay, and Ben Shneiderman. *Readings in Information Visualization*. Morgan Kaufmann, 1999.

[Str98] Thomas Strothotte. *Computational Visualization*. Springer Verlag, 1998.

[WFC+99] Seth White, Maydene Fisher, Rick Cattell, Graham Hamilton, and Mark Hapner. *JDBC API Tutorial and Reference, Second Edition*. Addison Wesley, 1999.