Consultas Heterogéneas en Bibliotecas Digitales Distribuidas

M.M. Martínez*, C. E. Cuesta*, P. de la Fuente* y J. C. Lamirel+

* Universidad de Valladolid
Departamento de Informática (ATC, CCIA, LSI)
Campus Miguel Delibes, 47011 Valladolid, España
email: {mercedes,cecuesta,pfuente}@infor.uva.es

+ LORIA, BP 239
54506 Vandoeuvre Cedex, France
email: lamirel@loria.fr

Resumen

La heterogeneidad en los lenguajes de consulta dificulta la integración de bibliotecas ya existentes con el fin de constituir una nueva biblioteca digital en entornos distribuidos. A la multiplicidad en los lenguajes de consulta se suma la diversidad en los formatos de los datos y en los resultados de las consultas de cada servidor. La biblioteca que acoge un nuevo servidor debe garantizar la interoperabilidad entre el recién llegado y el resto del sistema. En este trabajo hacemos una revisión del problema y las posibles soluciones, y presentamos una propuesta basada en la integración en el sistema de mediadores que asumen las tareas de traducción necesarias para garantizar la interoperabilidad en el sistema. La implementación de la solución propuesta opera sobre una biblioteca documental (documentos XML) donde tanto los datos como los sistemas de acceso a dichos datos están distribuidos; cada servidor de datos del sistema contiene su propio software de indexación y búsqueda en sus bases de documentos. La propuesta es suficientemente flexible para utilizarla en casos de mayor heterogeneidad como, por ejemplo, en la integración de bases de datos relacionales y repositorios que contienen documentos XML.

PALABRAS CLAVE: Biblioteca digital, federación, mediadores, distribución, heterogeneidad, interoperabilidad, lenguajes de consulta

1 Introducción

En este trabajo proponemos una solución al problema de la heterogeneidad en los lenguajes de consulta en bibliotecas digitales distribuidas. Las bibliotecas digitales distribuidas integran varias colecciones, repartidas en distintos servidores. La visión del usuario de la biblioteca es, no obstante, la de una única colección (un único punto de entrada) sobre la cual realiza sus consultas.

Cuando hablamos de **heterogeneidad** en los lenguajes de consulta nos referimos a la situación de varias consultas con distinta sintaxis, pero semántica común. Por ejemplo, la consulta "*buscar la aparición simultánea de los términos X e Y en los documentos de la base*", puede traducirse en "`X and Y`", "`X Y`", "`+X +Y`", etc. Sin embargo, todas ellas son representaciones de la misma consulta. En estas situaciones, es necesaria una etapa en la cual se traduzca la consulta del usuario a las consultas que acepta cada herramienta de búsqueda.

El entorno para el cual implementamos nuestra solución es una colección distribuida que contiene documentos semiestructurados (XML [14]) sobre los cuales se crean dos tipos de índices. Por un lado, se utiliza una herramienta de indexación y búsqueda en texto plano[8]; por otro, se utiliza una herramienta de búsqueda en documentos XML [14].

La organización de este trabajo es la siguiente. La primera parte es una introducción a las bibliotecas digitales distribuidas y federadas, los tipos de heterogeneidad que pueden presentar, y las técnicas utilizables en este tipo de sistemas, con especial énfasis en la solución basada en mediadores. En la sección 2 describimos nuestra solución: su arquitectura, interfaces, datos y metadatos, así como el flujo de datos en el sistema. La

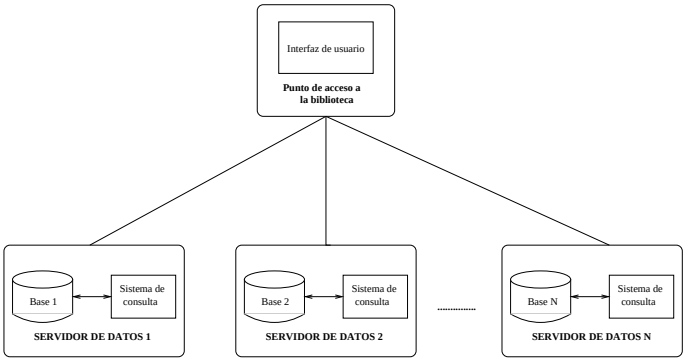

Figura 1: Biblioteca distribuida

sección 3 revisa algunas bibliotecas existentes en las que también se ha abordado este problema. Finalmente, presentamos las conclusiones y la propuesta de trabajo futuro.

1.1 Bibliotecas digitales distribuidas

Las **bibliotecas digitales** ofrecen a los usuarios información, agrupada en colecciones *organizadas*, accesibles a través de un único "punto de entrada" a la biblioteca. Permiten acceder a la información que almacenan, consultarla, y proporcionan otros servicios tendentes a facilitar la interacción del usuario con el sistema. La *distribución* y la *heterogeneidad*, en caso de que existan, son transparentes al usuario, que percibe la biblioteca como una colección única y homogénea.

Las bibliotecas digitales **distribuidas** están formadas por varias colecciones, dispersas en servidores distintos. La causa más común de distribución es que existan varios proveedores de datos, cada uno de los cuales aporta una o más colecciones a la biblioteca. A la hora de integrar un nuevo repositorio al sistema pueden ocurrir dos cosas: Primera, el nuevo servidor se integra al sistema aportando sus propios métodos de acceso a los datos. En este caso debe realizarse en algún momento la "traducción" del lenguaje de consulta local a un lenguaje de consulta común al sistema. Sería el caso de una biblioteca **federada**, donde componentes diseñados para trabajar de forma autónoma cooperan entre sí [9, 4]. Segunda, en el momento de la integración en el sistema, cada servidor de datos adopta un módulo de indexación y búsqueda común con la totalidad del sistema. A este tipo de sistema, donde los componentes se diseñan cuidadosamente para cooperar entre ellos, lo denominamos, sencillamente, **distribuido**.

Por ejemplo, la figura 1 presenta el caso más sencillo de distribución. La biblioteca consta de un **punto de acceso**, que proporciona la interfaz de usuario y redirige las consultas de usuario a los **servidores de datos**. Una vez recibe las respuesta de todos los servidores de datos, integra los resultados individuales y los presenta al usuario como una sola respuesta. En cada servidor de datos hay una réplica de una cierta herramienta de consulta; el lenguaje de consulta que aceptan los servidores es el mismo para todos, y los resultados que devuelven tienen un único formato, que será el del software de consulta instalado en todos ellos.

1.1.1 Heterogeneidad e interoperabilidad

Las causas de heterogeneidad en una biblioteca son múltiples, aunque se pueden establecer dos grandes categorías: *heterogeneidad en los* **datos** (incluidos los metadatos) y *heterogeneidad en las* **aplicaciones** que los crean y los manipulan.

En lo que concierne a la **heterogeneidad de los datos**, puede deberse a la heterogeneidad de formatos (`.doc, .tex, .xml, .gif, .tiff`, etc), a la coexistencia de datos de distinta naturaleza (estructurados, semiestructurados, no estructurados), o incluso a problemas relacionados con la terminología. Estos últimos son especialmente relevantes.

Si se trata de las **aplicaciones**, la causa de heterogeneidad puede estar en el tipo de aplicaciones, el modo de tratar los datos (de los sistemas de gestión de bases de datos a las herramientas de recuperación de información textual), y en las plataformas o lenguajes de implementación.

La distribución de los servicios de una biblioteca incrementa la probabilidad de encontrar cualquiera de los tipos de heterogeneidad mencionados, máxime cuando los proveedores de información son independientes.

El problema no se limita a presentar de modo uniforme los datos al usuario. Se debe conseguir que los servidores que componen la biblioteca interaccionen correctamente, a pesar de la heterogeneidad en el sistema. Esto es, debe existir **interoperabilidad** entre los elementos del sistema. Esta propiedad resulta importante porque aporta niveles de autonomía altos para los componentes, menor coste en la infraestructura, facilidad de uso y colaboración entre las aplicaciones, y *escalabilidad*.

1.2 Soluciones propuestas en sistemas heterogéneos

Las soluciones para integrar repositorios heterogéneos se pueden clasificar según dos criterios. El primer criterio es la política de resolución de las consultas utilizada en el sistema. El segundo es la solución adoptada para implementarlos.

Atendiendo a la política de resolución de consultas, podemos distinguir dos tipos de sistemas [15]:

1. La integración se produce **bajo demanda** (*on-demand*). Supone dos pasos:
   (a) Una vez aceptada una consulta, se deciden los servidores de datos a los cuales se puede reenviar y se generan las consultas apropiadas para cada servidor.
   (b) Los resultados recibidos de los servidores del sistema sufren una traducción, y se integran antes de presentarlos al usuario.
2. La integración es previa a la consulta del usuario. Se conoce como **data warehousing** (almacén de datos). En este caso los pasos son:
   (a) De cada repositorio de datos se extrae la información considerada relevante en el sistema para efectuar las búsquedas. Los datos extraídos de los distintos servidores de datos del sistema se mezclan y almacenan en un repositorio centralizado.
   (b) Cuando el usuario realiza una consulta, ésta es procesada en el repositorio central, sin acceder a los servidores de datos.

La solución basada en *data warehousing* implica la existencia de un servidor suficientemente potente en el sistema para albergar y manipular el repositorio central, así como de métodos eficientes de actualización incremental de dicho repositorio.

La solución *bajo demanda* es apropiada cuando es importante disponer de datos actualizados en todo momento . Es la única solución posible cuando es difícil, o incluso imposible, disponer de un servidor donde crear un repositorio central con los datos extraídos de todos los repositorios.

En cuanto a la solución elegida para implementar el sistema, se pueden distinguir tres tipos de soluciones[9]: la utilización de *agentes intermedios* o *mediadores*, la *especificación* de la interacción entre los componentes, y la utilización de *agentes móviles*.

Agentes intermedios o mediadores

Son componentes de software que asumen las tareas de traducción necesarias para incorporar un nuevo servidor al sistema. Facilitan la autonomía de los sistemas locales. La contrapartida es que, cada vez que se incorpora un nuevo componente, hay que incorporar un agente. Por ello, esta tecnología se beneficia enormemente del uso de estándares abiertos, y de la utilización de metadatos en la descripción y traducción entre componentes. Algunos ejemplos de agentes intermedios son los gateways SQL - XML o HTTP - Z39.50[1].

Soluciones basadas en especificación de la interacción

En este caso la solución consiste en la completa descripción de la semántica y estructura de todos los datos y operaciones, de modo que los componentes del sistema puedan interaccionar entre ellos después de la inspección de sus respectivas especificaciones. Aunque proporciona un alto grado de autonomía, llevarla a cabo es muy complejo. En esta aproximación, los metadatos son aún más importantes que en las implementaciones basadas en mediadores.

Agentes móviles

Los agentes móviles son componentes software que viajan a través de la red hacia puntos donde acceden a los servicios o recursos que necesitan. Una vez finalizan la ejecución, retornan a su localización original con los resultados. Por ejemplo, podrían emplearse para realizar una consulta remota.

[1] Z39.50 es un protocolo de Recuperación de Información muy utilizado en el acceso distribuido a un OPAC (Online Public Access Catalog) y ampliamente conocido en entornos relacionados con la biblioteconomía.

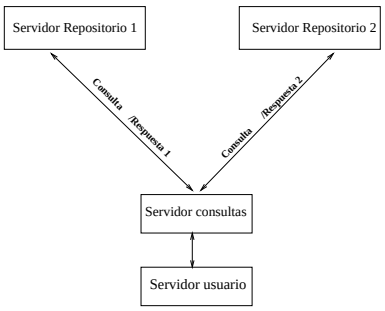

Figura 2: Distribución sin heterogeneidad

2 Propuesta de solución utilizando mediadores

En esta sección presentamos nuestra propuesta para la integración de servidores con lenguajes de consulta heterogéneos en una biblioteca digital distribuida. El tipo de biblioteca que estamos tratando es federada, con tratamiento de las consultas bajo demanda. La solución adoptada es la utilización de *mediadores*, que se implementan como *objetos distribuidos*[6].

La propuesta consta de la arquitectura (sección 2.1), la estructura de los datos que viajan por el sistema y metadatos (secciones 2.2 y 2.3), la organización temporal en el flujo de los datos (sección 2.4), y las interfaces (sección 2.5).

2.1 Arquitectura de la propuesta

Partimos de la arquitectura inicial para redistribuir la consulta en un sistema donde el lenguaje de consulta es único (ver figura 2), que se corresponde con la situación explicada en el último párrafo del apartado 1.1. El servidor de usuario envía una consulta al servidor de consultas, que, una vez ha decidido a qué repositorios debe redirigir la consulta, la redirecciona intacta a todos ellos. Tras recibir los resultados, lo siguiente que hará es fusionarlos.

Los elementos que forman parte del sistema son:

1. El **servidor de usuario**. Ofrece la interfaz de usuario. Permite al usuario recuperar documentos de los servidores de repositorios y consultar la biblioteca.
2. El **servidor de consultas**. Encargado de recoger las consultas que le proporciona el servidor de usuario y efectuar aquellas operaciones necesarias para devolver una lista de registros respuesta.

   Concretamente, se ocupa de:
   (a) Decidir a qué repositorio o repositorios debe redirigir cada una de las consultas.
   (b) Redireccionar la consulta a los servidores de datos.
   (c) Integrar los resultados que recibe de los servidores de datos y devolver el conjunto resultante al servidor de usuario.
3. Los **servidores de datos** (o **servidores de los repositorios**). Uno por cada base de datos del sistema. Ofrecen dos servicios: primero, ejecutan consultas sobre la base de datos, cuyos resultados devuelven en un formato propio; segundo, permiten recuperar registros de la colección.

En este sistema introducimos los **traductores** (mediadores), con lo que obtenemos la arquitectura que aparece en la figura 3:

- Los **traductores de consultas** (*trad-consultas* en la figura). Tantos como lenguajes de consulta diferentes haya en el sistema. Son traductores 1:1, del lenguaje de consulta "genérico" en el cual llega la consulta del usuario al servidor de consultas, al lenguaje de consulta local de cada herramienta de consulta.
- Los **traductores de formato** (*trad-formatos* en la figura). Traducen los resultados del formato local de respuesta de cada servidor de datos al formato común del servidor de consultas.

La composición de estos dos módulos da lugar a un componente, que denominamos **traductor**.

Ahora, el servidor de consultas interacciona con los traductores, en vez de hacerlo directamente con los servidores de datos.

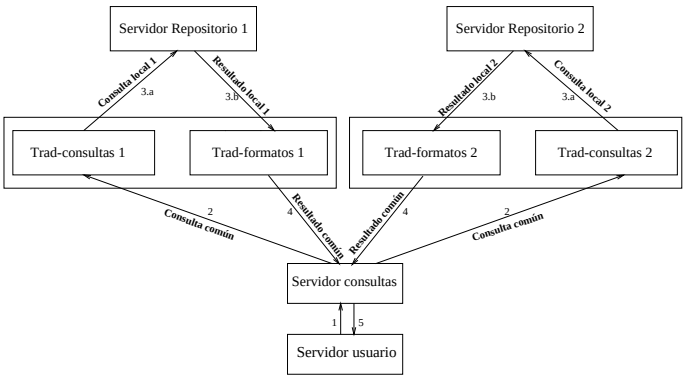

Figura 3: Distribución con heterogeneidad en los lenguajes de consulta

2.2 Metadatos del sistema

Los metadatos que necesita el **servidor de consultas** para redireccionar la consulta que recibe como entrada son:

- Lista de repositorios del sistema. Necesita saber cuáles son los repositorios del sistema, y su ubicación.
- Lista de lenguajes de consulta que hay en el sistema y traductor asociado a cada lenguaje. Además, se debe mantener la correspondencia entre los repositorios y sus lenguajes de consulta.
- Criterios de búsqueda en cada servidor de datos.

2.3 Datos resultado

Los resultados de una consulta son un conjunto de identificadores, correspondientes a los registros que responden a esa consulta: Cada *conjunto-resultado* consta de los siguientes datos:

- Identificador del conjunto-resultado. Este identificador lo utilizará el servidor de consultas para distinguir entre los distintos conjunto-resultado que mantiene en cada momento.
- Lista de *elementos*. Cada uno de ellos contiene a su vez:
  - Identificador local del elemento dentro de la base de datos en la que se encuentra.
  - Identificador del elemento dentro del conjunto-resultado.
  - Identificador de la base de datos o repositorio dentro del sistema. Se compone del identificador del servidor más el identificador de la base dentro del servidor.
  - Opcionalmente, puede incluirse el registro.

La figura 4 muestra la especificación para un objeto de tipo conjunto-resultado (*ResultSet*) en el lenguaje de definición de interfaces (IDL) de CORBA [12] en el sistema, acorde con una implementación donde los agentes son objetos distribuidos.

```
//Result Set (IDL)

struct ElResul {
        unsigned long   OrdenEl; // id. del elemento dentro del result set
        string          Servidor; // servidor donde se encuentra el elemento
        string          Repositorio; // repositorio en el cual se encuentra
        string          IdDoc;  // id. local del documento (URL)
        string          Doc;     // el documento resultado
};

struct ResultSet {
        string                  IdRS; // id. del result set
        sequence <ElResul>      LisResul; // lista de resultados
};
```

Figura 4: Especifición IDL para el conjunto-resultado

2.4 Secuencia temporal del flujo de datos

La secuencia temporal del flujo de datos en el sistema tras una consulta del usuario también se puede apreciar en la figura 3:

1. El servidor de consultas recibe la consulta del servidor de usuario.
2. El servidor de consultas redirecciona la consulta. Inspecciona los metadatos del sistema para seleccionar los repositorios y traductores adecuados para cada repositorio, e invoca a los módulos traductores.
3. Cada módulo traductor ejecuta la consulta que ha recibido. Es decir:
   (a) Traduce la consulta genérica al lenguaje de consulta local (*Trad-consultas*$_i$).
   (b) Invoca la herramienta de búsqueda del servidor de datos, pasándole la consulta que acaba de crear (*Consulta local*$_i$).
   (c) Traduce los resultados que recibe la herramienta de búsqueda del formato local al formato general (*Trad-formatos*$_i$).
4. Cada módulo traductor envía los resultados traducidos al servidor de consultas.
5. El servidor de consultas integra los resultados y los devuelve al servidor de usuario.
6. El servidor de usuario muestra los resultados al usuario.

2.5 Interfaces

En esta sección presentamos la especificación de las interfaces que posibilitan la interacción entre los componentes del sistema del modo que hemos descrito.

2.5.1 Servidor de base

1. *ConsultarBase(in string consulta-local, out ResultSet resultados)*
   Su entrada es un string que contiene una consulta que puede ser directamente ejecutada por el software de indexación. La salida es la lista de resultados para esa consulta. Estos resultados son del tipo explicado en la sección 2.3.
2. *RecuperarRegistro(in string identificador): string*
   Recibe como parámetro el identificador del registro solicitado. Devuelve dicho registro.

2.5.2 Traductor

Engloba el traductor de consulta y el traductor de formatos por simple composición de interfaces.

2.5.3 Traductor de consultas

1. *Traducir (in string consulta-usuario, out string consulta-local)*
   A partir de la consulta expresada en el lenguaje *genérico* del usuario, devuelve la consulta expresada en el lenguaje local al cual está asociado el traductor.

2.5.4 Traductor de formatos

1. *Transformar(in string registro, out ElResul ele-result-set)*
   Recibe un registro en un string. Devuelve un elemento resultado, acorde a la especificación vista en la sección 2.3.
2. *Transformar(in string lista-registro, out ResultSet result-set)*
   Recibe una lista de registros en la entrada. Retorna el *conjunto-resultado* equivalente.

2.5.5 Servidor de consultas

1. *BuscarBiblioteca (in string consulta-usuario, out ResultSet result-set)*
   Toma la consulta del usuario como entrada y devuelve la respuesta, que se pasará al servidor de usuario. Esta es la función que invocará el servidor de usuario.

2.6 Distribución física de los componentes

Cada servidor de datos del sistema ofrece una o más colecciones de datos, cada una de ellas accesible a través de su correspondiente interfaz, tal como se ha explicado en la sección anterior. En cada servidor de datos habrá tantos sistemas de indexación y búsqueda como colecciones publique el servidor.

En cuanto a la ubicación física de los componentes traductores nos hemos encontrado dos posibilidades:

1. Cada traductor se ubica en el mismo servidor que la base de datos con la que interacciona (ver figura 5). Esta solución es la más adecuada cuando se pretende minimizar el tráfico de datos en la red. Se requiere la colaboración del servidor de datos, de modo que incorpore dicho traductor al nodo que alberga la colección.

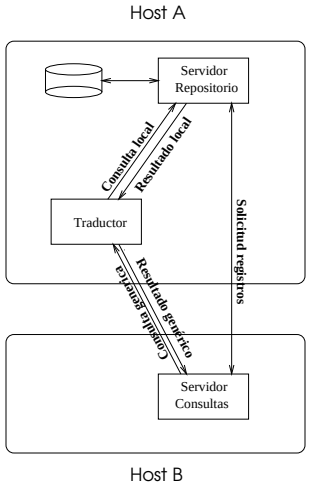

Figura 5: Cada base de datos y su traductor se ubican en el mismo servidor.

2. La ubicación física de los traductores en el sistema es independiente de la ubicación de las bases de datos. Es decir, los traductores no tienen por qué estar al lado de las bases de datos que sirven. Esta solución es óptima cuando se pretende incorporar a una biblioteca nodos "poco colaborativos": servidores en los cuales no se admite la posibilidad de integrar nuevo software.

Ya que en nuestro caso tenemos plena libertad de acción sobre los servidores de datos hemos optado por la alternativa de ubicar los traductores al lado de las bases de datos. Sin embargo, como se ha dicho, la integración de servidores menos colaborativos resulta completamente factible.

3 Trabajos relacionados

Algunos ejemplos de bibliotecas digitales distribuidas en las cuales se utilizan mediadores son la biblioteca digital de ERCIM[2], NDLTD[11], el proyecto de Bibliotecas Digitales de Stanford[10], *Aleph*[7] y AQUARELLE[1]. Stanford propone la utilización de mediadores (*Library Service Proxies*) que se ocupan de las labores de traducción necesarias para integrar servidores implementados inicialmente sobre protocolos diferentes. El modelo *Aleph* también propone una solución para sistemas federados, donde cada comunidad indexa localmente la información que produce; en este caso, la política empleada para resolver las consultas difiere de la nuestra: se intenta resolver localmente las consultas en el servidor que las recibe, y sólo en caso de no poder hacerlo se redirigen a otros servidores. En NDLTD también ponen especial atención en los metadatos que describen a los servidores; sin embargo, uno de sus fines primordiales es permitir consultas multilingües. El proyecto AQUARELLE propone una solución para bibliotecas distribuidas, utilizando el protocolo Z39.50. Cada servidor de datos que se incorpora al sistema debe implementar un servidor Z39.50; en cambio, nosotros sólo requerimos a cada nuevo servidor que exporte sus servicios a través de objetos que implementen las interfaces del sistema, lo cual requiere un esfuerzo mínimo por parte de los proveedores.

Por último, el proyecto CICYT "Biblioteca Virtual de Textos Españoles de Siglo de Oro accesibles via Internet"[3] -en el cual participan tres de los autores de este trabajo- consiste en la implementación de una

biblioteca digital distribuida federada, uno de cuyos objetivos es conseguir la fácil integración de servidores heterogéneos, como son bases de datos relacionales y servidores de documentos XML.

4 Conclusiones y trabajo futuro

4.1 Conclusiones

De las distintas soluciones que actualmente permiten abordar el problema de la heterogeneidad de lenguajes de consulta en las bibliotecas digitales, hemos optado por aquella basada en *mediadores* que asumen la tarea de traducción entre lenguajes de consulta locales y lenguaje de consulta común al sistema. Las razones que nos han inclinado hacia dicha elección son varias. En primer lugar, las soluciones basadas en mediadores han sido ampliamente utilizadas en este tipo de sistemas distribuidos ([13, 1, 2]), demostrando la factibilidad de dicha solución. Segundo, la **escalabilidad** de los sistemas basados en mediadores es alta; integrar un nuevo servidor (un nuevo software de consulta en nuestro caso) requiere a lo sumo la incorporación de un nuevo mediador (en el caso de que no esté ya en el sistema), sin que por ello se vean afectados los módulos con los cuales el sistema ya está funcionando.

Hemos puesto además especial atención en la **flexibilidad** del sistema. Flexibilidad en el tipo de servidores de datos del sistema: aunque la solución implementada es para documentos XML a los cuales se aplican distintas herramientas de indexación y búsqueda, el modelo propuesto es aplicable a sistemas con más heterogeneidad en los datos[2]. Y flexibilidad en la incorporación al sistema de nuevas funcionalidades (gestión de las sesiones de usuario, clasificación de resultados, etc.) sin modificar los módulos ya disponibles.

La utilización de un enfoque basado en *objetos distribuidos* nos libera de tener que ajustarnos a algún protocolo específico en el diálogo entre servidores, y permite incorporar al sistema servidores de modo sencillo.

Como se ve en la sección 2.1, hemos agrupado los traductores de consulta y de formato en un único componente al cual hemos llamado traductor. Esta opción parece más lógica, dado que cada interacción con una colección implica ambas etapas de traducción. Por otra parte, minimizamos el flujo de datos en el sistema. Haber mantenido separadas las dos tareas, hubiera implicado una de estas dos suposiciones:

1. Todos los resultados se adaptan a un formato común. Esto es altamente improbable en un sistema donde las herramientas de consulta son heterogéneas.
2. Los servidores de datos se ocupan de la traducción de formatos en los resultados. Es decir, en el servidor de datos se integra un envolvente para el software de búsqueda, que traduce los resultados antes de devolverlos. Esta suposición implica que los servidores de datos tienen un nivel de colaboración alto, ya que están dispuestos a integrar este módulo. Consideramos que la solución que hemos escogido es más flexible cara a la integración de servidores poco colaborativos.

4.2 Trabajo futuro

La siguiente etapa lógica es introducir la gestión de *sesiones de usuario* en el sistema. Actualmente no se ha planteado esta posibilidad, con lo cual no es posible retroalimentar una consulta o iterar sobre un conjunto de resultados.

La solución presentada es buena para incorporar a un sistema servidores que ya disponen de sus propias herramientas de indexación y consulta de las bases de datos que ofrecen. Nos planteamos no obstante, la experimentación con plataformas de agentes móviles[5] que permitan la incorporación a una biblioteca de nodos no colaborativos. Es decir, dado un repositorio para el cual no se proporcionan herramientas de manipulación y consulta, poder integrarlo en la biblioteca. En este caso recae sobre el sistema la labor de incorporar dicho repositorio. Ya no estamos frente a un nuevo lenguaje de consulta, sino que el sistema debe implantar las herramientas que permitan la consulta de dicho repositorio: crear los índices que permitan la consulta posterior, y proporcionar el software que consulta dichos índices. La creación y mantenimiento de dichos índices sería gestionada por un agente indexador móvil, que en el momento del alta en el sistema del nuevo repositorio se desplace y lo indexe. A su vez, dicho módulo se encargaría de depositar en el servidor el módulo que permite

[2]Por ejemplo, es válido para un sistema donde se integren servidores de datos cuyas bases son documentos XML con servidores que ofertan bases de datos relacionales.

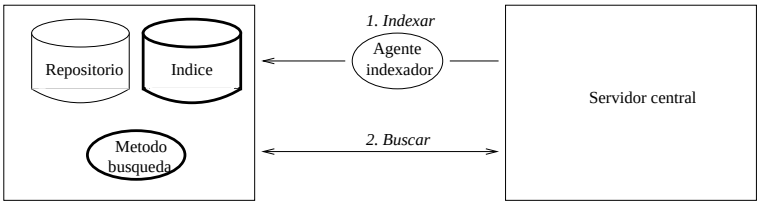

Figura 6: Repositorios sin software de indexación. En negrita, los componentes depositados por el agente indexador móvil.

la consulta. Una vez creados los índices, el proceso de redirección de la consulta se simplifica, ya que no es necesario el módulo traductor, al desaparecer la heterogeneidad (figura 6).

Referencias

[1] AQUARELLE. The Information Network on the Cultural Heritage. http://aqua.inria.fr/Aquarelle/.

[2] S. Biagioni, J.L.Borbinha, R. Ferber, P. Hansen, S. Kapidakis, L. Kovacs, F. Roos, y A. M. Vercoustre. The ERCIM Technical Reference Digital Library. En *Second European Conference on Research and Advanced Technology for Digital Libraries*, septiembre 1998.

[3] Nieves R. Brisaboa, M. José Durán, Charlo Lalín, Juan Ramón López, José R. Paramá, Miguel R. Penabad, y Ángeles S. Places. Propuesta de Arquitectura para un Sistema de Bases de Datos Documentales. En *IV Jornadas de Ingeniería del Software y Bases de Datos*, noviembre 1999.

[4] Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser, y Herbert Weber. Federated Information Systems: Concepts, Terminology and Architectures. Technical report, Technische Universität Berlin, 1999.

[5] Gianpaolo Cugola, Carlo Ghezzi, Gian Pietro Picco, y Giovanni Vigna. Analyzing Mobile Code Languages. En Jan Vitek y Christian Tschudin, editors, *Mobile Object Systems: Towards the Programmable Internet*, volume 1222 of *Lecture Notes in Computer Science*, págs. 93–110. Springer-Verlag, 1997.

[6] Jim Farley. *JAVA. Distributed Computing.* The JAVA series. O'REILLY, 1998.

[7] Gerard Rodríguez i Mulà. *Federation Mechanisms for Large Scale Cooperative Systems*. PhD thesis, Department of Computer Architecture. Universitat Politècnica de Catalunya.

[8] Udi Manber y Sun Wu. Glimpse: A tool to search through entire file systems. En *Proceedings of the USENIX Winter Conference*, págs. 23–32, january 1994.

[9] Adreas Paepcke, Che Chuan K. Chang, Héctor García Molina, y Terry Winograd. Interoperability for Digital Libraries Worldwide. *Communications of the ACM*, april 1998.

[10] Adreas Paepcke, Steve B. Cousins, Scott W. Hassan, Ketchpel Steven P, Martin Roscheisen, Héctor García Molina, y Terry Winograd. Using distributed objects for digital library interoperability. *Computer*, may 1996.

[11] James Powell y Edward A. Fox. Multilingual Federated Searching Across Heterogeneous Collections. *D-Lib Magazine*, september 1998.

[12] Jon Siegel. *CORBA. Fundamentals and Programming*. Wiley Computer Publishing. John Wiley & Sons, Inc., 1996.

[13] Stanford Digital Libraries Project. http://www-diglib.stanford.edu.

[14] W3C. *Extensible Markup Language (XML) 1.0*, Febrero 1998.

[15] Jennifer Widom. Research Problems in Data Warehousing. En *Proc. of 4th Int'l Conference on Information and Knowledge Management (CIKM)*, november 1995.