Rev. Fac. Ing. Univ. Antioquia N.° 44. pp. 144-157. Junio, 2008

Modelo de servicios *web* para replicación de instantáneas sobre motores múltiples de bases de datos

Web services model to snapshot replication over multiple database engines

Roberto Erazo, Carlos Cobos[*]*, Martha Mendoza*

Grupo de I+D en Tecnologías de la Información, Departamento de Sistemas, Universidad del Cauca. Sector Tulcán, Edificio FIET, Oficina 422, Popayán, Colombia

(Recibido el 12 de septiembre de 2007. Aceptado el 29 de Enero de 2008)

Resumen

En este artículo se presenta un modelo de servicios *Web* para soportar replicación de instantáneas en múltiples motores de bases de datos. Este modelo es abierto y esta compuesto por un conjunto de servicios *web*, agentes y una consola de administración que permite realizar tareas administrativas en forma independiente de la aplicación y del motor de bases de datos. El modelo usa un catálogo maestro de replicación (CMR) para almacenar los datos de replicación. En el documento, se presenta la arquitectura empleada y el comportamiento del modelo para que el lector adquiera un mejor entendimiento del concepto de replicación de datos. Además, se discuten algunos trabajos de investigación relacionados con los servicios de replicación independientes de los motores de bases de datos.

---------- *Palabras clave*: Servicios *web*, modelo abierto de replicación, instantáneas, RDBMS, inserción, extracción, suscripción, publicación, distribución, SOA.

Abstract

In this paper, we present a web service model to support snapshots replication in multiple database engines. This model is open and is composed by a

* Autor de correspondencia: teléfono: + 57 + 2 + 820 98 00 ext. 2119, fax: + 57 + 2 + 820 98 00 ext. 2102, correo electrónico: ccobos@unicauca.edu.co. (C. Cobos).

set of web services, agents and a manage console that allows to perform administrative tasks independently from the application and the database engines. This model uses a replication master catalog (CMR) to store the replication data. We show the architecture used and the behavior of our replication model to provide a better understanding about the data replication, as well as, discussion of several papers related to the replication services independent from the database engines.

---------- *Keywords:* Web services, open replication model, snapshot, RDBMS, push, pull, subscription, publication, distribution, SOA.

Introducción

En la actualidad, el proceso de replicación de datos [1] es responsabilidad principal del motor de base de datos, que normalmente corresponde a versiones empresariales de los productos comerciales de mayor uso en el mercado. En este escenario, la aplicación depende de los servicios de distribución que ofrece el motor, y normalmente estos servicios "avanzados" tienen un costo adicional muy elevado. Por lo anterior, las empresas enfrentan un problema económico, ya que deben pagar un valor adicional para adquirir "*in house*" servicios que no están incluidos en las versiones estándar de los actuales motores de base de datos. En otro contexto, cuando las empresas despliegan sus aplicaciones bajo el esquema de *hosting*, el problema es aún mayor, ya que los proveedores de servicios de *Internet* (*Internet Service Provider*, ISP) o los proveedores de servicios de aplicación (*Application Service Provider*, ASP), normalmente no pueden cubrir necesidades tan específicas en cuanto a tecnología y en el caso de hacerlo, el costo es muy elevado. Además, en muchos casos las empresas terminan pagando por un servicio que es mucho más de lo que realmente necesitan, es decir, en muchos casos sólo se necesita un método específico de replicación y no todo el conjunto general de posibilidades ofrecidas, y por las que se paga en conjunto.

Por otro lado, si bien la interoperabilidad entre motores de bases de datos se ha hecho un requerimiento necesario, pocos de ellos ofrecen verdaderos servicios que permitan replicar información entre distintos motores. Normalmente, en estos casos se recurre a productos de terceros que también tienen un costo elevado o bien no soportan la cantidad de motores de base de datos que se necesitan.

En nuestro caso [2-4] la solución propuesta para este problema consiste en asignar la responsabilidad de la replicación de datos a un modelo de servicios *Web* que soporte la replicación (hasta el momento sólo esta probado el método de replicación por instantáneas) sobre múltiples motores de base de datos. Este modelo va acompañado de un catálogo maestro de replicación (CMR) que se usa para dar persistencia a la información de cada uno de los nodos que forman parte de la red de replicación y un conjunto de servicios del sistema operativo (agentes) que revisan la agenda de creación de publicaciones y la recepción de las mismas en los suscriptores.

A continuación (sección 2) se presentan los trabajos de investigación y/o desarrollo relacionados y las diferencias con esta investigación. En la sección 3 se describen los elementos más importantes del modelo y su dinámica de operación en el manejo de replicación de instantáneas con suscripción por inserción y por extracción. Finalmente, se presentan las principales conclusiones del trabajo y las próximas labores a desarrollar.

Trabajos Relacionados

Mercury [5] es una herramienta de *software* que ayuda a los usuarios a realizar un monitoreo acerca del desempeño, estado y configuración de aplicaciones con bases de datos distribuidas. Empresas como Oracle Corp., Informix Software, Inc. y el ASK Group, incluyeron agentes *Mercury* en sus productos de base de datos. Es una herramienta que ayuda a monitorear el desempeño de aplicaciones que involucran bases de datos distribuidas. Se destaca en este trabajo, la administración y monitoreo de desempeño de bases de datos distribuidas con capacidad de monitorear diferentes motores de base de datos. Sin embargo, esta herramienta parte de que los servicios de replicación ya estén definidos en el motor de base de datos.

Peer Direct Distributed Enterprise [6] es la *suite* empresarial distribuida de *PeerDirect*, y permite a las compañías descentralizar sus aplicaciones de negocios automáticamente y hacer una sincronización bi–direccional, distribuyendo y replicando datos corporativos y aplicaciones a través de múltiples ubicaciones, con dispositivos de escritorio y móviles, mientras se provee herramientas locales para una fácil administración. Su principal aporte, consiste en que permite la sincronización de aplicaciones y datos indepen-

diente del dispositivo y da soporte a varios motores de base de datos. Sin embargo, requiere que la empresa que lo adopta se una a la plataforma *Progress OpenEdge* con un costo elevado.

Progress DataXtend Enterprise [7] ofrece la integración de datos con posibilidad de realizar sincronización entre diferentes fuentes de datos con aplicaciones distintas y diferentes motores de base de datos, incluso usando diferentes esquemas, permite además lectura y escritura en tiempo real de cualquier fuente de datos, manteniendo la consistencia de los datos, no requiere cambios en la forma en que las aplicaciones acceden a los datos, manteniendo niveles de desempeño y escalabilidad. Sin embargo, el costo de la inclusión de este producto es muy elevado. Además, la herramienta no esta bajo un modelo abierto para ser adoptado y utilizado.

Microsoft SQL Server 2005 [8] brinda servicios de replicación de datos en diferentes topologías de replicación: replicación Transaccional (*transactional)*, replicación por Fusión (*Merge*) y replicación por instantáneas (*snapshot*). Permite además la suscripción de un suscriptor *Oracle*. Su principal aporte es su claro modelo de replicación que permite apropiar de mejor forma los conceptos relacionados a la replicación de datos. Sin embargo, las tareas de replicación son propietarias y hacen parte del motor, lo cual no permite trasmitir estos servicios de replicación a otros Sistemas Manejadores de Bases de Datos Relacionales (*Relational Database Management System*, RDBMS) u Objeto Relacionales (ORDBMS).

GlobeDB: Autonomic Data Replication for Web Applications [9]: Es un sistema *hosting* de aplicaciones *Web* que realiza replicación autónoma de datos. El proceso de replicación y distribución de datos lo maneja el sistema de manera automática con pequeñas tareas de administración manual. Su principal aporte, es el modelo de arquitectura planteado, que incorpora al modelo de las aplicaciones un *middleware* que se encarga de las tareas de la replicación y sincroniza los datos entre los servidores a través de *Internet*. Sin embargo soporta un solo modelo de replicación (*master/slave*) y usa un *driver* entre la aplicación y el acceso a datos, lo cual en cierto modo cambia la arquitectura de las aplicaciones que deseen implementar replicación.

Las características principales del modelo propuesto en este trabajo y su implementación son: 1) El servicio de replicación de instantáneas no se implementa para un motor de bases de datos específico. 2) El servicio se puede usar para replicar instantáneas en motores que soporten o no el servicio de replicación, incluido además un ambiente heterogéneo de dichos motores. 3) El servicio sigue siendo independiente de la aplicación que replica los datos. 4) Con el uso de un bloque de aplicación de consultas distribuidas, las instantáneas se pueden consultar más eficientemente por la aplicación (este aspecto se explica al final de la sección 3).

Modelo Propuesto

Para describir de mejor manera la interacción del modelo de replicación propuesto, se hace uso de la metáfora de la publicación de una revista en el mundo real, muy usada por varios autores y en especial por Microsoft [8]. Se entiende entonces la utilización de tres roles claramente identificados, ellos son: Publicador, entidad encargada de la emisión de la publicación de la revista. Suscriptor, entidad(es) (o personas) que consumen o leen la publicación emitida por dicho publicador. Distribuidor, entidad encargada de la entrega de las publicaciones emitidas por el publicador a todos los suscriptores.

Además, se deben manejar los conceptos básicos de una publicación, ellos son: Publicación, uno de los ejemplares de la revista emitida por el publicador. La revista está conformada por varios artículos que hacen parte de un ejemplar en particular de la revista. Artículo, la unidad mínima de información que conforma una publicación.

Dentro del ejemplo de la publicación de una revista, una vez un ejemplar de la revista (publicación) es emitida, esta lista para ser distribuida a las entidades (o personas) suscriptoras. Existen

dos tipos de suscripciones posibles, o dicho de mejor manera, existen dos formas de obtener un ejemplar de la revista, la primera a través de una suscripción directa a la revista, lo que implica que cada vez que sale una nueva edición de la misma, esta se entrega a cada uno de los suscriptores (suscripción por inserción). La segunda forma de adquirirla, es solicitándola a cualquiera de las entidades que se encargan de la distribución de la misma (suscripción por extracción).

Esta metáfora permite adaptar los conceptos al contexto de las bases de datos, redefiniendo cada uno de los conceptos de la siguiente forma: Publicador, equipo servidor de base de datos que genera una publicación definida previamente por el administrador de los servicios de replicación de datos. Suscriptor, equipo(s) servidor(es) de base de datos que consumen una publicación existente. Distribuidor, equipo servidor que conoce a los publicadores, sus publicaciones y los respectivos suscriptores de cada publicación, además se encarga de entregar las publicaciones a cada uno de ellos.

Por tanto, una Publicación es un conjunto de artículos generados por un publicador. La publicación representa una unidad de información que lógicamente representa un único concepto. Artículo, es un conjunto de información que representa un objeto o bien un subconjunto de datos de ese objeto. Por ejemplo, un artículo puede estar definido por una sentencia *SELECT* de todas las columnas y filas de una tabla, es decir la tabla o vista en sí. O bien se puede definir solamente como unas cuantas columnas y/o filas de una tabla o vista.

El modelo de replicación propuesto está compuesto por: 1) un conjunto de servicios *Web XML* (*eXtensible Markup Language*) [10] que expone la lógica de replicación, 2) dos agentes que usan los servicios *Web* que a su vez pueden llamar a otros servicios *Web* ubicados en otros equipos (servidores) de la red y 3) un catálogo maestro de replicación que da la persistencia a los datos de los servidores, roles que desempeñan en la red, publicaciones, suscripciones, entre otros.

Tipos de suscripción

Los tipos de suscripción que se han modelado son: Suscripción por inserción *(Push),* en donde el publicador genera la publicación y esta se entrega a todos los suscriptores en forma inmediata. La tarea de distribución de la publicación puede estar asignada a la misma entidad (servidor) publicadora o bien puede estar asignada a otra entidad (servidor). Suscripción por Extracción *(Pull)***.** El suscriptor, solicita la última publicación generada por el publicador. Esta petición se hace directamente al distribuidor, y es la expresión más alta de asincronía.

La figura 1 muestra la dinámica del modelo cuando se habla de una suscripción por inserción (*Push*). La descripción del proceso de entrega de las publicaciones es el siguiente: 1) El agente publicador cada cierto intervalo de tiempo (5 segundos) está verificando a través del servicio *Web* de publicación si dentro de la agenda hay una nueva publicación que generar. 2) El servicio *Web* de publicación genera en la carpeta de publicaciones, la publicación en su respectiva carpeta y los respectivos artículos, repartidos en diferentes archivos *XML*. 3) Se genera un resumen criptográfico a través de una función *hash* (*Message-Digest Algorithm 5, MD5* [11]) de cada uno de los artículos en un archivo que también se guarda en la carpeta de la publicación. 4) Se comprime toda la publicación, esto implica todos los artículos y el archivo de seguridad. 5) Ese archivo se pasa al servicio *Web* de distribución y se almacena en una carpeta de publicaciones a distribuir. 6) El servicio *Web* de distribución hace un llamado a los servicios *Web* de suscripción de los suscriptores por inserción (*push*) a esa publicación, pasándole el archivo comprimido que recibió en el llamado anterior a través de una capa de comuniones segura (*Secure Socket Layer, SSL* [12]). 7) El servicio *Web* de suscripción descomprime el archivo que le llega usando un mecanismo de control de redundancia cíclica (*Cyclic Redundancy Check, CRC* [13]) del mismo y saca un MD5 a cada artículo comparándolo con el archivo de seguridad que

viene dentro del archivo comprimido. 8) La publicación se sincroniza con la base de datos de suscripción. En esta figura cada rectángulo hace referencia a un nodo de la red, que puede corresponder a un equipo (servidor) físico distinto o a un equipo que cumple con los tres roles.

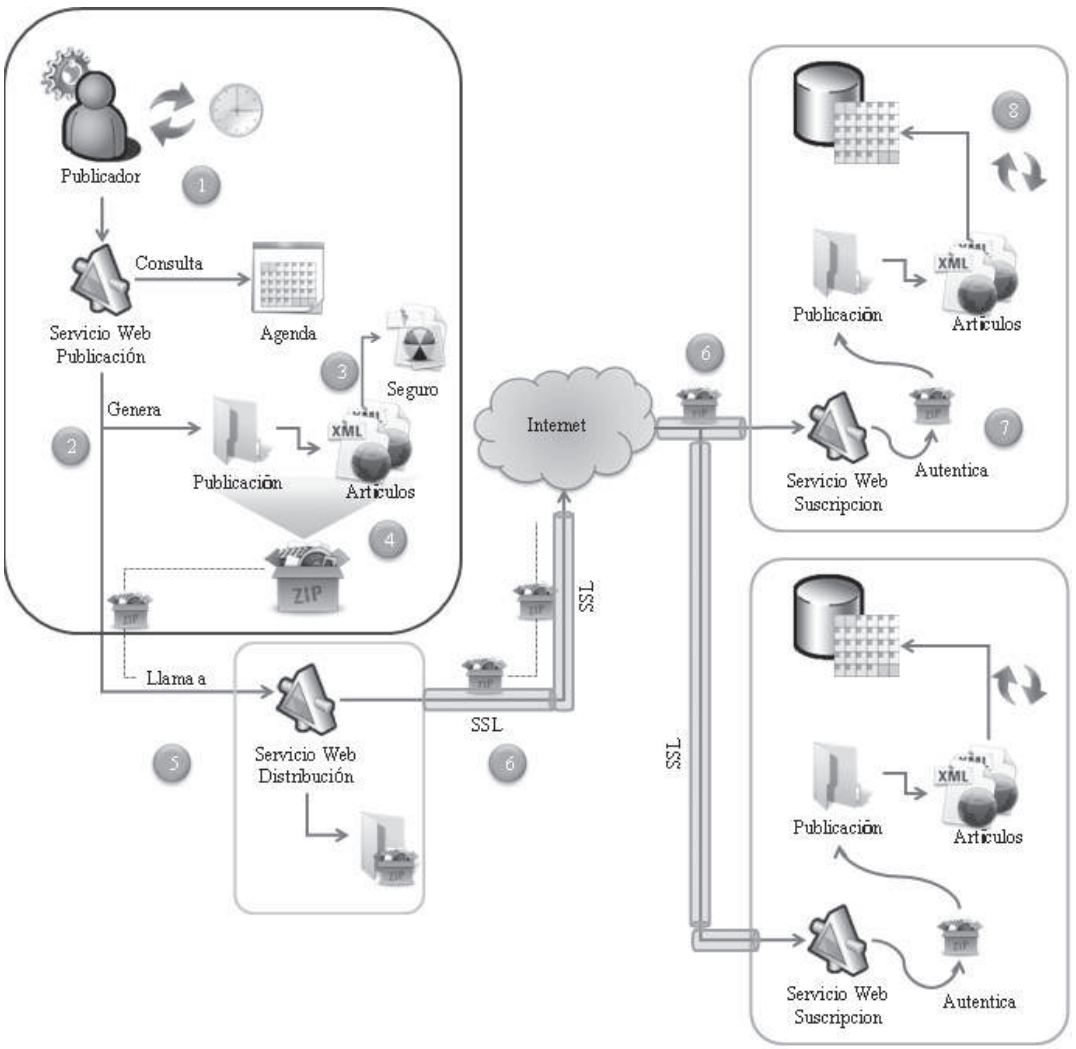

Figura 1 Publicación con suscripción por inserción (*Push*)

En una suscripción por extracción el comportamiento es diferente, básicamente en cuanto a los momentos (t) en los que ocurre el evento de creación de la publicación y el de solicitud de la misma por parte del suscriptor. Además, porque el origen del segundo evento lo genera el propio suscriptor con base en una agenda propia (ver figura 2). En este sentido, el momento (t_0) en el que se realiza la publicación es "lógicamente" anterior al momento (t_1) en que los suscriptores solicitan la publicación. Si esta regla no se cumple, el suscriptor puede recibir una excepción, porque no se ha realizado la publicación, o una versión vieja de la publicación. Es labor del administrador del sistema organizar adecuadamente las agendas. En la figura 2, los pasos 1, 2, 3, 4, y 5 del lado del publicador son iguales a la suscripción por inserción. Luego en el lado del

suscriptor, los pasos son: 1) El agente suscriptor se activa cada determinado intervalo de tiempo (5 segundos) y a través del servicio *Web* consulta la agenda de suscripciones. 2) El servicio *Web* de suscripción solicita al servicio *Web* de distribución la publicación correspondiente a la suscripción activa. 3) El servicio *Web* de distribución entrega la última publicación generada correspondiente al *Id* de publicación que solicitó el suscriptor y la entrega al servicio *Web* de suscripción a través de SSL. 4) El servicio Web de suscripción descomprime el archivo que le llega verificando el CRC del mismo y saca un MD5 a cada artículo comparándolo con el archivo de seguridad que viene dentro del archivo comprimido. 5) El servicio *Web* de suscripción sincroniza la publicación con la base de datos de suscripción.

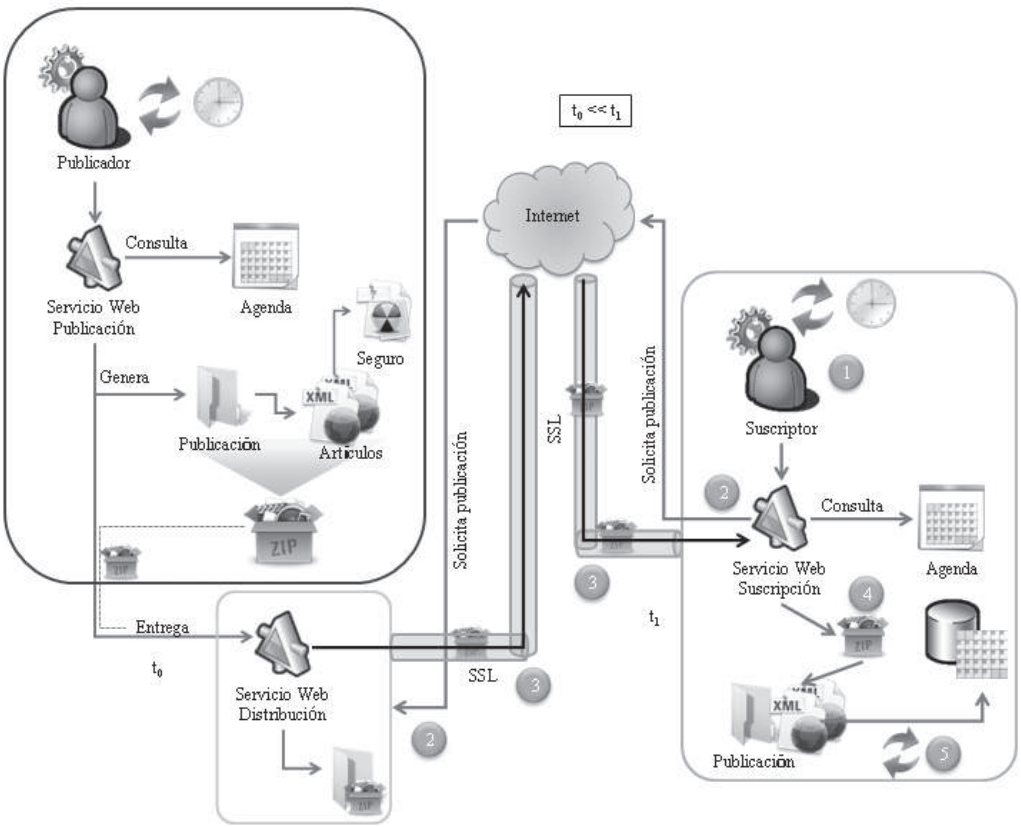

Figura 2 Publicación con suscripción por extracción (Pull)

Roles de los Servidores

Se debe tener en cuenta que los equipos involucrados en la red de replicación, pueden asumir cualquier de los tres roles o los tres roles al mismo tiempo. Esto implica que un publicador puede ser distribuidor y suscriptor a su vez. La configuración de los roles que cada uno de los equipos debe desempeñar es una labor del administrador del sistema de replicación. Cada uno de los equipos contendrá entonces un modelo completo del CMR, sin importar el rol que desempeñen dentro del sistema de replicación. Cada uno de los roles utiliza una porción del CMR y usa uno o varios de los agentes y servicios Web. (Ver figura 3).

A continuación se especifican cada uno de los objetos utilizados por cada uno de los roles. Si un equipo desea configurarse como publicador, distribuidor y suscriptor, este utilizará todos los agentes y servicios involucrados para cada labor.

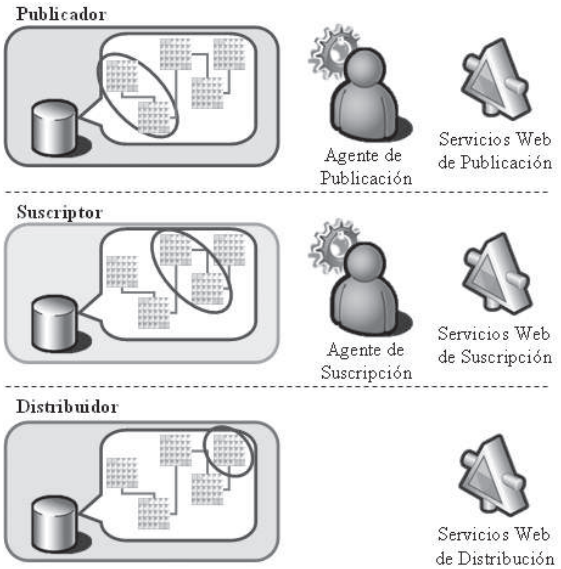

Figura 3 Recursos usados en cada rol dentro del modelo de replicación

El rol de Publicador como se muestra en la figura 3 activa el agente de publicación y este a su vez usa los servicios *Web* de publicación. Dentro del modelo relacional propuesto cuando un servidor se configura como publicador, los objetos del CMR utilizados por este rol son: Publication, almacena la información de la publicación, tal como el nombre de la publicación, la base de datos de publicación y los identificadores de los artículos involucrados en la publicación. Articles, almacena la información de los artículos que componen una publicación en particular, los datos más importantes a tener en cuenta aquí son la tabla o vista que define el artículo, las columnas que lo conforman y de ser necesario una condición que permita filtrar datos que requiera el administrador del sistema de replicación. *Execution Plan*, almacena información de la siguiente ejecución de una publicación (esta tabla se actualiza después de que una publicación se realiza con base a su programación y periodicidad). Esta tabla la consulta el agente de publicación, para determinar qué publicaciones se deben llevar a cabo en una determinada hora y fecha. *Programation*, almacena la información de la hora y fecha en las cuales la publicación debe llevarse a cabo. Dado que esta tarea generalmente puede tener intervalos de repetición y diferentes tipos de periodicidad, las tablas descritas a continuación ayudan a soportar este tipo de comportamientos. *Repetition,* almacena información acerca de qué tipo de periodicidad tiene una publicación en particular. Los tipos de periodicidad son: Diaria, Semanal, y Mensual. *Daily*, almacena información acerca de la periodicidad en días de una programación en particular. Es decir, cada cuántos días debe llevarse a cabo una publicación. *Weekly,* almacena información acerca de la periodicidad en semanas de una programación en particular. Es decir cada cuántas semanas una publicación en particular debe llevarse a cabo y explícitamente, qué días de la semana. *Montlhy,* almacena información acerca de la periodicidad en meses de una programación en particular. En la figura 4 se muestra la parte del CMR que se usa para el rol de publicador.

El rol de Suscriptor como se muestra en la figura 3 activa el agente de suscripción y este a su vez el servicio *Web* de suscripción. Dentro del CMR propuesto cuando un servidor se configura como suscriptor, los objetos de la base de datos utilizados por este rol son: *Suscription,* almacena la información relacionada con una suscripción en particular. Si esta suscripción es por inserción (*push*) esta no contará con programación independiente, lo que implica que la sincronización de los datos se llevará a cabo una vez se emita la publicación. Si la suscripción es por extracción (*pull*), la suscripción contará con una programación independiente a la publicación. Es labor del administrador del sistema de replicación que exista coherencia entre las dos tareas. *Execution Plan*, almacena información de la siguiente ejecución de una suscripción (esta tabla se actualiza después de que una suscripción por extracción se realiza con base a su programación y periodicidad). Esta tabla la consulta el agente de suscripción, para determinar qué suscripciones se deben llevar a cabo en una determinada hora y fecha. *Programation,* guarda la información acerca de la hora y fecha en las cuales la suscripción por extracción (*pull*) debe llevarse a cabo. Esta programación también se soporta en las tablas: *Repetition, Daily, Weekly* y *Montlhy*. En la figura 5 se muestra la parte del CMR que se usa para el rol de suscriptor.

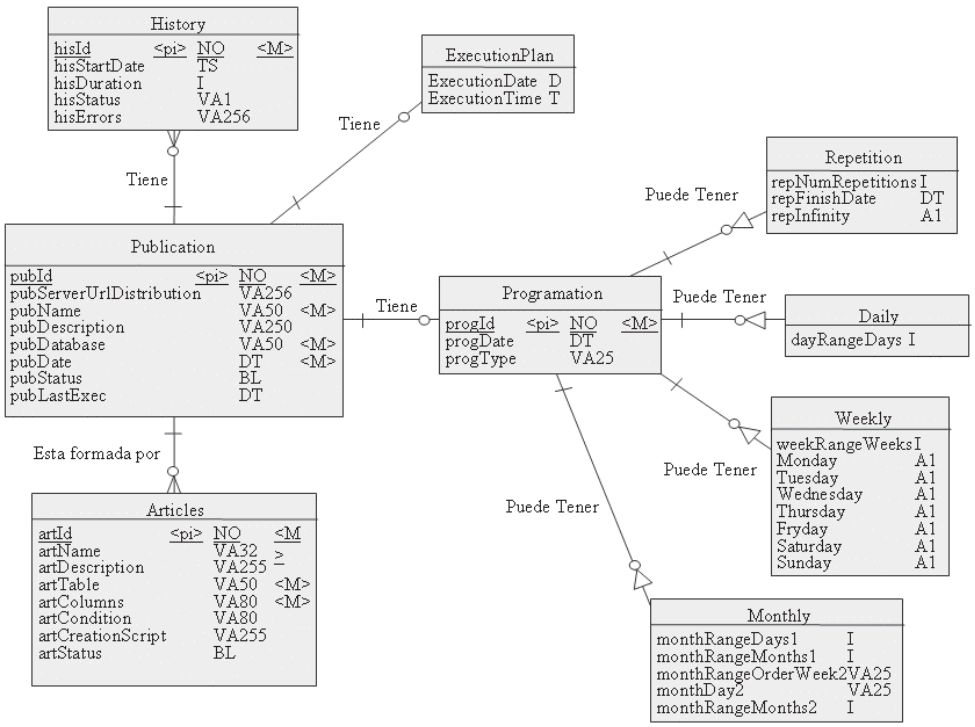

Figura 4 Catálogo maestro para la publicación de instantáneas

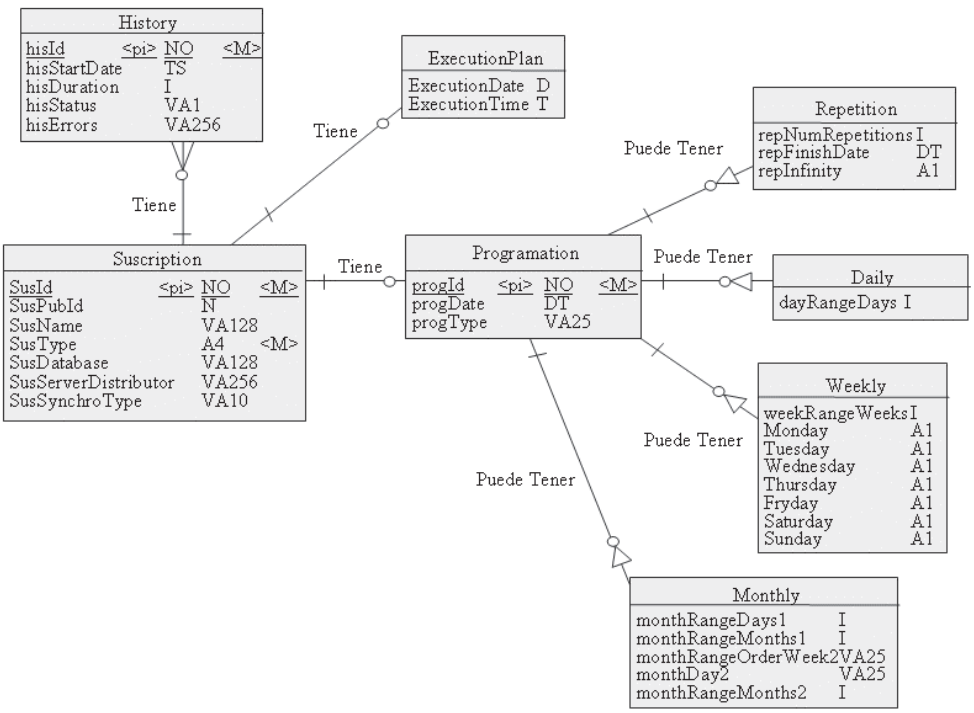

Figura 5 Catálogo maestro para la gestión de suscripciones de instantáneas

El rol de Distribuidor como se muestra en la figura 3 activa el servicio *Web* de distribución. Dentro del modelo relacional propuesto cuando un servidor se configura como distribuidor, los objetos de la base de datos utilizados por este rol son: *ServerRole*, almacena la información relacionada con los servidores que hacen parte de la red de replicación y el rol que desempeñan dentro de la misma. *PubToDistribute,* almacena información relacionada con las publicaciones que se deben distribuir. La información más importante acerca de las publicaciones a distribuir son: el nombre del servidor donde se encuentra la publicación y el *id* de esa publicación para ese servidor en particular, así como una contraseña para poder verificar que un suscriptor determinado puede suscribirse a dicha publicación. *SusToDistribute,* almacena información acerca de las suscripciones hechas a las publicaciones que el distribuidor debe realizar. La información más importante a tener en cuenta en esta tabla, es el nombre del equipo suscriptor, el *id* de la publicación a distribuir (*Id* de *PubToDistribut*e), el *Id* del suscriptor y el tipo de suscripción. En la figura 6 se muestra la parte del CMR que se usa para el rol de distribuidor.

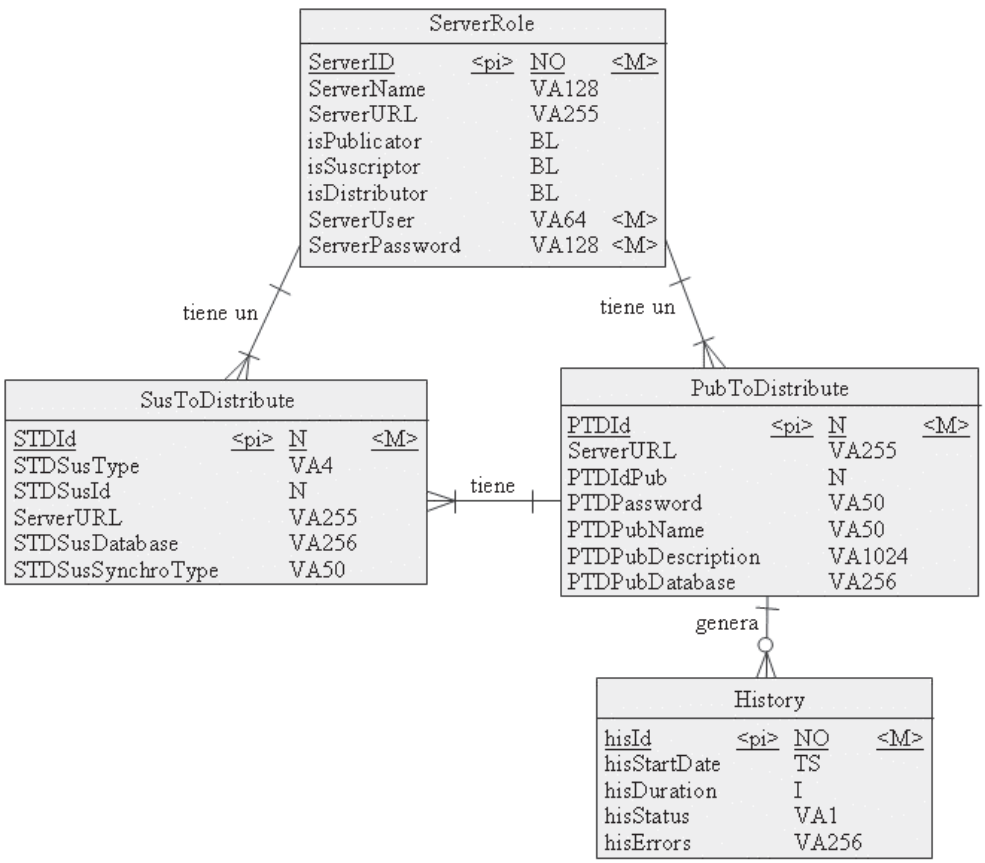

Figura 6 Catálogo maestro para la distribución de instantáneas

Consola de Administración

El modelo anterior ilustra cómo los servicios *Web* y los agentes de replicación (publicación y suscripción) interactúan para llevar a cabo las tareas de replicación. Las tareas de administración del sistema de replicación se llevan a cabo a través de una consola centralizada de administración realizada con una arquitectura de cliente inteligente [14], partiendo de la lógica

de servicios que se encuentra en los servicios *Web* de replicación explicados anteriormente, pasando por la lógica de negocios que prepara y procesa los datos y llegando a la interfaz que fue desarrollada con aplicación de escritorio. La figura 7 muestra cómo se construyó la consola de administración y la forma como se integra al modelo.

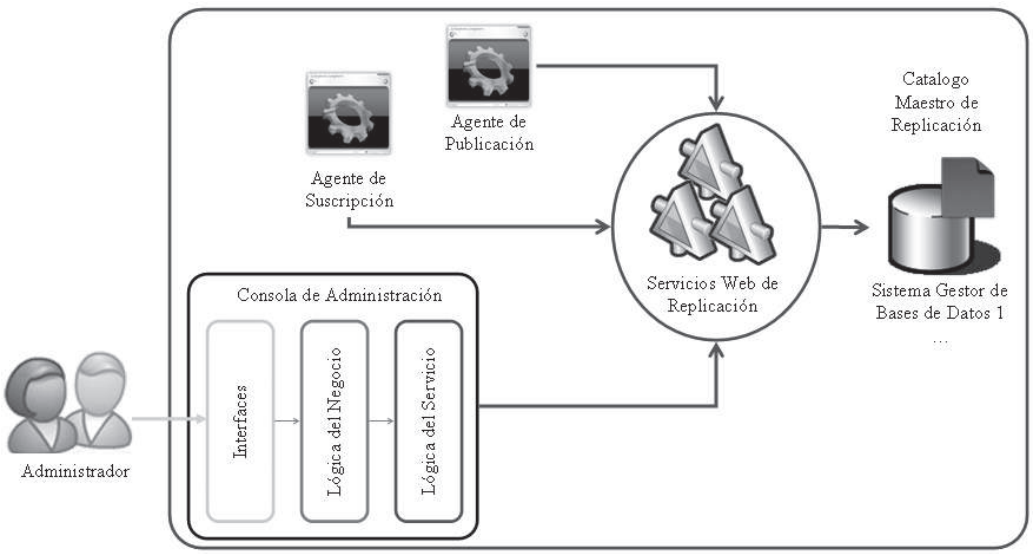

Figura 7 Arquitectura global de la consola de administración y componentes generales del modelo

En la figura 8 se puede apreciar una parte de la interfaz de la consola de administración, específicamente la funcionalidad relacionada con la creación de una nueva publicación con el asistente. Dicho asistente permite establecer el nombre de la publicación, los artículos a publicar y en el caso de la figura, muestra como establecer la programación de la ejecución de la publicación.

Una de las grandes ventajas [1] que tiene el modelo propuesto es la capacidad de poder realizar replicación a múltiples RDBMS, lo cual permite desarrollar sistemas más robustos e interoperables. La base para que el modelo permita la interacción con diferentes motores se encuentra dentro de un archivo de configuración XML que esta dentro de cada nodo de la red. Es decir, dentro de la red de replicación, cada servidor tiene un CRM completo y los agentes de replicación, estos a su vez tienen un archivo de configuración que definen los tipos de motores hasta ahora soportados por el modelo y las sentencias SQL necesarias para la consulta de tablas y vistas del sistema. El esquema que representa este archivo de configuración se ilustra en la figura 9. A medida que se necesita incorporar más motores se incluyen los datos correspondientes en este archivo de configuración; esto hace que el modelo sea flexible y se pueda usar en más motores de bases de datos.

Para una correcta representación de los datos entre motores, es requisito que las tablas a replicar cumplan con el estándar SQL 99 [15] en la definición de sus datos para conservar la representación de los datos en diferentes motores de base de datos. Para facilitar esta labor se investigó acerca de herramientas que desarrollan este tipo de validación que están disponibles de manera gratuita y se recomienda usar *SQL Validator for SQL99* [16]. Junto a esta herramienta se encuentra también un cuadro comparativo [17] de compatibilidad de tipos de datos y operaciones entre diferentes RDBMS basado en el estándar SQL 99.

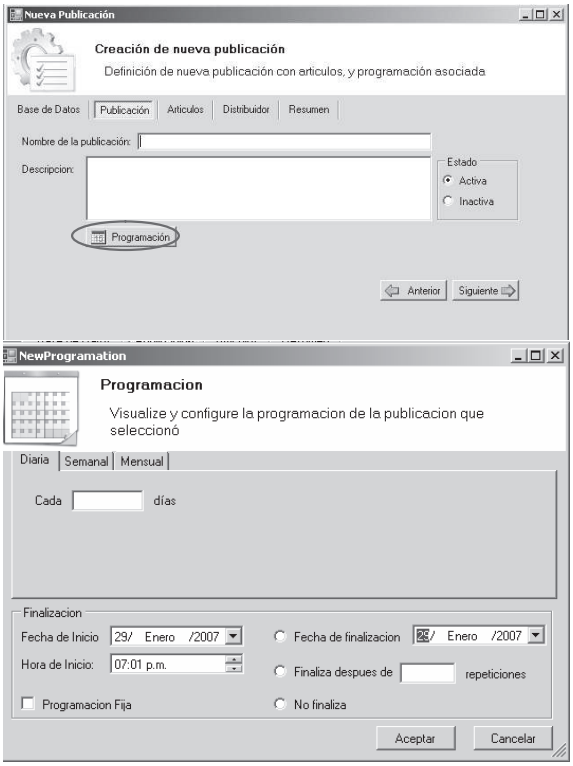

Figura 8 Interfaz de creación de una nueva publicación y de configuración de la programación de la misma

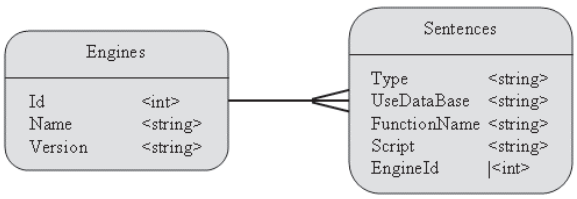

Figura 9 Esquema del archivo de configuración

Bloque de Replicación de Datos

Como se explicó en la sección anterior del modelo, toda la lógica de replicación se encuentra en los servicios *Web*. Estos servicios se pueden exponer a través de una fachada para que sean usados por cualquier aplicación que necesiten implementar y extender la replicación realizada por el modelo. Esto permite que el modelo sea flexible y se pueda ajustar a la medida de las necesidades de un aplicativo en particular. Con el modelo propuesto hasta este momento, se permite que las aplicaciones cuenten con el soporte para el desarrollo de consultas distribuidas sin que esto afecte su desempeño y sin usar productos de terceros al momento de replicar los datos [18].

Con el objetivo de probar el modelo de replicación y su implementación, se desarrolló una aplicación de consulta de información turística móvil, denominada *Easy Tour Guide* (ETG). En la figura 10 se muestran algunas interfaces de la aplicación.

Figura 10 Algunas interfaces de *Easy Tour Guide*

Esta aplicación cuenta con una base de datos compatible con SQL 99, en este caso *SQL Server* 2005 y tablas que almacenan información relacionada con localidades (ubicaciones geográficas), hoteles, iglesias, eventos, agencias, itinerarios de vuelos, restaurantes, entre otros. Usando la consola de administración se realizó la replicación de la tabla de localidades (por ser la más usada y además, cambia lentamente, es decir, cuenta con una baja tasa de modificaciones, entre otras) a una base de datos *Oracle 9i* ubicada en otro equipo.

Para el desarrollo de esta aplicación se implementó además un servicio "ingenuo" de consultas de datos distribuidos. Este servicio es ingenuo, porque selecciona al azar la fuente de datos para las

localidades (de la base de datos con SQL *Server* o con *Oracle*). En este sentido, cada cliente que usa ETG consulta una localidad, que puede haber sido consultada de un motor o del otro. En la figura 11 se aprecia la arquitectura de ETG, el servicio de consultas distribuidas denominado "servicio de acceso aleatorio" y la interacción con el modelo de replicación de instantáneas. En la figura 11 sólo aparece una consola de administración, puesto que este software se comunica con los dos servidores y los administra, el único requisito para poder realizar esto, consiste en contar con las credenciales de administrador (usuario y clave) para autenticar y autorizar las operaciones en los servicios web de replicación del servidor que se va a administrar.

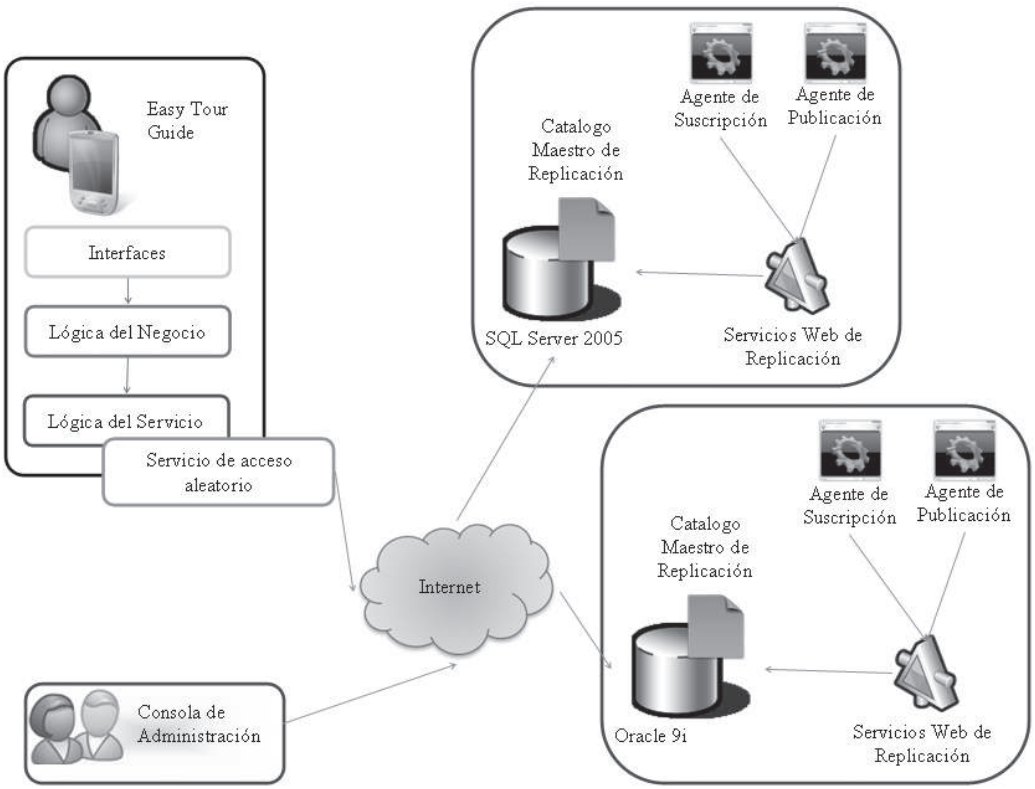

Figura 11 Arquitectura de Easy Tour Guide e interacción con el modelo de replicación de instantáneas

Conclusiones y trabajo futuro

Se logró modelar y desarrollar una herramienta de replicación de instantáneas sobre múltiples motores de bases de datos, basado en cuatro componentes principales: un conjunto de servicios *Web*, dos agentes, un catálogo maestro de replicación y una consola de administración que trabaja de manera independiente a las aplicaciones que deseen implementar replicación de datos.

Para poder hacer uso de este modelo y su implementación, se debe seguir un conjunto de lineamientos que permitan el modelado de tablas de fácil replicación, con tipos de datos soportados por el estándar SQL99 para asegurar un correcto mapeo de los datos a replicar por los servicios de replicación.

La implementación y apropiación de este tipo de servicios *Web* para replicación, permite el desarrollo de sistemas cada vez más robustos e inte-

roperables y facilitan las tareas de integración de datos cuando existen diferentes motores de base de datos involucrados, sin incurrir en los altos costos que hoy las compañías de *software* sostienen. La posibilidad de tener acceso al modelo de replicación propuesto, permite realizar implementaciones en distintos lenguajes y/o entornos de programación e irlo complementando a través del tiempo. La construcción de sistemas distribuidos en la academia y la investigación es uno de los principales propósitos de la industria [2].

El grupo de investigación espera desarrollar en el futuro cercano las siguientes actividades: Incluir meta datos que soporten más motores de base de datos. Incorporar el servicio para la solución de conflictos y manejo de concurrencia en sistemas de tiempo real que requieren que todos los participantes en la replicación puedan realizar lectura y escritura de datos. Soportar la sincronización de información con dispositivos móviles. Soportar transacciones distribuidas usando protocolos *Two-Face-Commit* y *Paxos Commit*. Finalmente, exponer servicios de replicación transaccional y *Merge*.

Referencias

1. C. J. Date. *Introducción a los sistemas de bases de datos: Bases de datos distribuidas*. 7ª ed. Prentice Hall. México. 2001. p. 651.
2. S. D. Michael. *Using Sharing to Simplify System Management. In Computer Systems: Theory, Technology, and Applications*. A. Herbert, K. Sparck Jones Eds. Springer-Verlag. New York. 2004. pp. 259-268.
3. K. Bettina. *Implementing Database Replication based on Group Communication. School of Computer Science*. McGill University. Montreal. Published in Fudico.2002. pp.1-3. http://citeseer.ist.psu.edu/kemme02implementing.html. Consultada octubre 20 de 2006.
4. P. A. Bernstein, E. Newcomer. *Principles of Transaction Processing*. The Morgan Kaufmann Series in Data Management Systems. pp. 293-310.
5. Mercury. IBM – LEGENT. *IBM-LEGENT make distributed database management splash*. http://www.findarticles.com/p/articles/mi_qa3649/is_199403/ai_n8719514. Consultada enero 18 de 2007.
6. PeerDirect Distributed Enterprise. *PeerDirect Announces Native Linux Support for PeerDirect Distributed Enterprise*. http://www.peerdirect.com/news/pressitem/pressrelease_197440/index.ssp. Consultada julio 12 de 2006.
7. Progress DataXtend Enterprise. *Progress DataXtend* . http://www.progress.com/dataxtend/dataxtend_enterprise/index.ssp. Consultada julio 15 de 2006.
8. SQL Server 2005. SQL Server Replication. *Types of Replication*. Microsoft Corp 2007. http://msdn2.microsoft.com/en-us/library/ms151198.aspx. Consultada abril 4 de 2006.
9. S. Swaminathan, A. Gustavo, P. Guillaume, V. S. Maarten, Globe DB. "Autonomic Data Replication for Web Applications*". Proceedings of the 14th international conference on World Wide.* Web. WWW '05. ACM press. May 2005. p. 33
10. M. Arenas, L. Libkin. "A normal form for XML documents". *ACM Trans. Database Systems*. Vol. 29. 2004. pp.195-232.
11. B. Byfield. "Take command: password's progress". *Linux J*. Vol 89. 2001, p. 6.
12. D. Wagner, B. Schneier. "Analysis of the SSL 3.0 protocol". *Proceedings of the 2nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce* - Volume 2 (Oakland, California, November 18 - 21, 1996). USENIX Association, Berkeley, CA, p. 4-4.
13. T. Henriksson, D. Liu. "Implementation of fast CRC calculation". *Proceedings of the 2003 Conference on Asia South Pacific Design Automation* (Kitakyushu, Japan, January 21 - 24, 2003). ASPDAC. ACM Press, New York, NY, pp. 563-564. DOI= http://doi.acm.org/10.1145/1119772.1119892
14. Smart Client Technology. *Microsoft smart clients: Power, performance, flexibility*. Micro-soft Corp. March 2005. http://www.microsoft.com/net/SmartClient.mspx. Consultada febrero 14 de 2006.
15. ANSI/ISO/IEC International Standard. *SQL/Framework: Data types specified in ISO/IEC 9075-2*. ANSI/ISO/IEC International Standard. 1999. p. 29.
16. SQL Validator for SQL99. *Mimer Developer* http://developer.mimer.com/validator/parser99/index.tml. Consultada abril 8 de 2006.
17. Core SQL 1999: feature comparison chart. Mimer SQL Developers: http://developer.mimer.com/validator/comparison/upd_comparison_chart.tml?pf=true. Consultada abril 10 de 2006.
18. Application Blocks Definition. *Patterns and Practices: Application Blocks*. Microsoft Patterns and Practices developer center. http://msdn.microsoft.com/practices/guidetype/AppBlocks. Consultada junio 14 de 2006.