**Universidad Central "Marta Abreu" de Las Villas**

**Facultad de Matemática, Física y Computación**

# TRABAJO DE DIPLOMA

# GENERACIÓN DE ESQUEMAS FÍSICOS EN EL DISEÑO DE BASES DE DATOS DISTRIBUIDAS EN SQL SERVER

**Autor:** Hector Baranda Somonte

**Tutores:** Dr. Abel Rodríguez Morffi

Dra. Luisa M. González González

Curso 2010-2011

Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencias de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del tutor Firma del Jefe del Seminario

# Dedicatoria

**A mi madre**, por su inigualable amor, apoyo e incalculable confianza.

**A mis abuelos**, por haber vivido para ver este momento.

**A mi familia**, por estar allí para guiarme.

**A mis amigos**, esos que siempre están ahí para mí.

# Agradecimientos

A Maritza por ser la persona número uno en mi vida, sin ti no habría podido ni comenzar. Te quiero mami.

A mis abuelitos, esos viejos míos que son lo más lindo y grande.

A Mario Ernesto que más que un primo es un padre.

A José, Arelis y Keilan que han sido otra familia para mí.

A José Luis y Marisol por el maravilloso tiempo compartido, los quiero mucho.

A mi tutor Abel por su confianza y probada paciencia.

A mis amigos Albanis, Polo, José Ramón que más que amigos son mis hermanos, gracias por la confianza, el cariño y su amistad, sin ustedes el camino hubiese sido más difícil.

A Sandro por su amistad y todas esas noches de incalculable ayuda, gracias por tu tiempo amigo.

A Donna por la ayuda en la confección de este documento, por su tiempo, paciencia y cariño, gracias mi vida.

A Waldy por esa amistad de toda una vida, y porque sé que hubieses dado cualquier cosa por estar aquí, te quiero mi herma.

En general a todas esas personas que han estado a mi lado todo este tiempo para ayudar, para brindar su amistad…….

A todos muchísimas gracias.

## Pensamiento

“Ya se han escrito todas las buenas máximas. Solo falta ponerlas en práctica.”

Blaise Pascal

# Resumen

El procesamiento de bases de datos distribuidas consiste en trabajar con bases de datos, en las cuales la ejecución de transacciones, la recuperación y actualización de los datos, acontece usualmente en dos o más computadoras independientes, por lo general separadas geográficamente (aunque conceptualmente basta una computadora y un acceso costoso a los datos para utilizar bases de datos distribuidas). Actualmente existen varios problemas relacionados con el diseño de bases de datos distribuidas ya que se han propuesto resultados teóricos que son difíciles de utilizar o no han solucionado totalmente el problema para el que se han desarrollado. Uno de ellos es la obtención de esquemas físicos de bases de datos distribuidas. Esto se debe fundamentalmente a la fuerte dependencia que tiene de los sistemas gestores de bases de datos.

Este trabajo muestra un detallado examen de las características que debe seguir un esquema de replicación para Microsoft SQL Server y ofrece una solución computacional al problema de generación de esquemas físicos mediante replicación a través de la creación de scripts en Transact-SQL, tomando como entrada un documento XML que contiene el resultado del diseño de una base de datos distribuida obtenido por la herramienta SIADBDD. Esta aplicación fue desarrollada en Visual C # como parte de Visual Studio 2010.

# Abstract

Distributed database is a collection of data that are connected by logic, stored separately and used together in a computer network. In other words, the distribution of this data allows other units can access data from a particular unit. In Distributed Database there are advantages and disadvantages.

There are several problems concerning to the design a distributed database. Some theoretical outcomes have been proposed but most of them are difficult to implement or to use, and none of them fully solve the problem for which they have created. One of these problems is the final allocation of data in which each fragment of data must be allocated to one or several sites, where the fragment will be saved. This is mainly due to the strong dependence of the database management systems.

The present work presents a detailed examination of the issues must follow a replicated schema generation in Microsoft SQL Server and provides a computational solution to the problem of physical schema generation by replication through the creation of scripts in Transact-SQL taking as input a XML document containing the SIADBDD tool design results obtained. This application was developed in Visual C # as part of Visual Studio 2010.

# Tabla de Contenidos

# Índice de Figuras

# Introducción

En los últimos años han surgido grandes cantidades de innovaciones tecnológicas, que unidas al descenso en los costos del hardware, han estimulado el desarrollo de los sistemas de bases de datos distribuidas (SBDD).

Según Őzsu y Valduriez (Őzsu and Valduriez, 1999), una base de datos distribuida (BDD) es una colección de múltiples bases de datos (BD), lógicamente interrelacionadas distribuidas sobre una red de computadoras. Así, el objetivo fundamental de los SBDD es integrar la manipulación de datos para que sean presentados al usuario como una única colección de datos global y coherente.

Por tanto, la distribución involucra el hecho de que los datos no residen necesariamente en el mismo sitio, pero poseen propiedades comunes que los vinculan, y se facilita su acceso a través de una interfaz común. Es necesario destacar que los enlaces entre los datos se llevan a cabo en una red de comunicación, lo que implica generalmente que los sitios estén localizados en diferentes áreas geográficas y con capacidad de procesamiento autónomo.

Existen varias razones técnicas para distribuir datos. En particular, los sistemas distribuidos se adaptan mejor a las necesidades de las organizaciones descentralizadas, ya que reflejan más adecuadamente su estructura y tienen importantes ventajas con respecto a los centralizados. La descentralización se justifica desde el punto de vista tecnológico, ya que permite autonomía local y promueve la evolución de los sistemas, así como los cambios en los requerimientos de los usuarios, proporciona una arquitectura de sistemas simple, flexible y tolerante a fallos, y ofrece buenos rendimientos. Estas razones a favor de la descentralización también añaden nuevas complejidades a su diseño e implementación.

Las diferentes técnicas de distribución de datos generalmente se basan en la semántica de ellos, y se rigen por principios idénticos que las BD centralizadas, incorporando otros detalles particulares, como la fragmentación de las entidades y su posterior localización en los diferentes sitios de la red. La fragmentación es muy útil para mejorar los tiempos

de respuesta y garantizar el paralelismo de un sistema (Baião et al., 2004, Coulon et al., 2005, Johansson et al., 2000, Lin et al., 2005).

En el diseño de la distribución influyen muchos factores relacionados con la BD, las aplicaciones que acceden a la misma, la comunicación de la red, y sobre el sistema de computadoras que la soporta; lo que hace que sea muy complicada la formulación de un problema de distribución. Por tanto es necesario decidir cómo fragmentar y distribuir los datos sobre los diferentes sitios y cuáles de estos datos deben ser replicados. Además, el diseño físico de una BDD exige decisiones y procesamientos complejos respecto a la ubicación de los fragmentos de datos.

Como **antecedentes** a este trabajo se tiene que en el Grupo de Bases de Datos del Centro de Estudios de Informática de la Universidad Central “Marta Abreu” de Las Villas, se desarrollaron herramientas de ayuda al diseño de BDD. La herramienta SIADBDD integra todas las herramientas creadas en un ambiente único de coordinación entre ellas a través de un catálogo compartido, ALLOCATOR es una de estas herramientas la cual se encarga de la asignación de fragmentos a los diferentes sitios de procesamiento identificados en la red donde se ubicará la BDD que se diseñe. Esta herramienta realiza la ubicación mediante la creación y posterior ejecución de un conjunto de scripts en cada uno de los sitios pero trabaja como una aplicación monolítica con todas las funcionalidades juntas. La solución obtenida está fuertemente ligada al uso del sistema de gestión de bases de datos (SGBD) Microsoft SQL Server. Por otra parte, surge la necesidad de obtener soluciones de diseño aplicables a otros SGBD. Consecuentemente el **problema de la investigación** que se presenta para este trabajo está dado por la necesidad de obtener una solución al problema de ubicación para ser adaptada al nivel físico soportado por SQL Server que sea independiente de SIADBDD y se integre dinámicamente a esta.

Como **hipótesis** se tiene que una herramienta para la generación de esquemas físicos de BDD mediante replicación soportado por SQL Server que no dependa directamente de las estructuras internas de SIADBDD, facilita el trabajo de los diseñadores de BDD y un potencial mantenimiento posterior.

El **objetivo** de esta investigación es diseñar e implementar una herramienta para la creación de los fragmentos físicos de una BDD en Microsoft SQL Server, con el fin de obtener una solución independiente que se integre dinámicamente con SIADBDD a partir de un repositorio compartido en formato XML.

Como **objetivos específicos** se tiene los siguientes:

- Identificar los aspectos de programación en Transact-SQL que permiten crear entornos de replicación de datos para materializar diseños de BDD a nivel físico en Microsoft SQL Server.
- Diseñar una herramienta que permita generar los esquemas físicos de ubicación usando los elementos programáticos de replicación identificados y use como entrada un documento XML con la información requerida.
- Implementar la herramienta diseñada usando como entrada un documento XML.
- Probar DISTRIBUTOR con un caso de estudio real.

Las **preguntas de la investigación** son las siguientes:

- ¿Qué elementos obtenidos como resultado del diseño de BDD en SIADBDD son necesarios para realizar la ubicación física de los fragmentos?
- ¿Qué características debe poseer un asistente para ubicación física de fragmentos de datos que resulte favorable al diseñador?
- ¿Qué aspectos de la replicación de datos se deben considerar al ubicar fragmentos físicos en SQL Server?

Relacionado con los antecedentes antes planteados, se puede decir que este estudio se **justifica** por su importancia desde el punto de vista práctico en la integración que tendrán los resultados esperados en una herramienta de ayuda al diseño de BDD.

Este trabajo es **viable**, ya que en el laboratorio de Bases de Datos existen posibilidades de realización de esta investigación, porque se cuenta con recursos materiales para ello, expertos para la consulta, y oportunidades de utilización de los resultados.

Los resultados obtenidos tienen **valor práctico** porque la herramienta ayuda en la creación de fragmentos físicos en Microsoft SQL Server que constituye una solución independiente que se integra dinámicamente con SIADBDD para lograr completar diseños de BDD.

La tesis está **estructurada** en tres capítulos de la siguiente forma:

El capítulo 1 trata sobre aspectos generales del diseño de sistemas de BDD y particulariza en la generación de esquemas físicos y en las etapas fundamentales de la replicación.

En el capítulo 2 se propone una solución concreta al problema planteado, mostrando su diseño y los aspectos más importantes de su implementación.

En el capítulo 3 se presenta un caso de estudio real y se muestran los resultados ofrecidos por la herramienta DISTRIBUTOR para dar solución a la problemática anterior.

Este documento culmina con las conclusiones, referencias bibliográficas y los anexos.

# Capítulo 1. Diseño de Bases de Datos Distribuidas

En este capítulo se introducen los elementos básicos para el diseño de SBDD, tanto lógico como físico, así como el estudio de la generación de esquemas y la replicación como una etapa esencial dentro de la creación de tales sistemas.

## 1.1 Generalidades sobre las bases de datos distribuidas

La utilización de las nuevas tecnologías así como el uso intensivo de Internet ha traído consecuencias importantes en las comunicaciones. Afectaciones como la distribución de los datos en localidades geográficamente distantes no escapan a este problema. Ante esta situación se buscan alternativas viables para integrar y compartir la información necesaria, por lo que surgen nuevas tecnologías capaces de interactuar ante dicho fenómeno y gestionar de manera estable la información de todas las organizaciones.

En muchas organizaciones geográficamente distribuidas, las vías centralizadas no representan una alternativa factible, y la migración hacia SBDD resulta natural. Muchos autores recalcan el principio de que un sistema distribuido muestra la estructura de la BD como un espejo de la estructura de la empresa, con lo cual se incrementa la localidad de referencia y se reduce drásticamente el tráfico en la red (Date, 2000).

Muchas han sido las acepciones para lograr la definición de un sistema distribuido, pero en ocasiones no convergen entre sí. La más viable en correspondencia con los principios básicos perseguidos en esta investigación es la siguiente: Un sistema distribuido es una colección de computadoras independientes, que aparecen ante los usuarios del sistema como una única computadora. El logro de elevados grados de independencia entre los programas de aplicación y los aspectos internos ha sido el resultado del perfeccionamiento, avance y progreso de las tecnologías de BD.

Como se expresó en la introducción a este trabajo, una BDD puede verse como una colección de datos que pertenecen lógicamente a un solo sistema, pero se encuentra

físicamente esparcida en varios sitios de la red, es decir, se tiene un conjunto de sitios conectados entre sí mediante algún tipo de red de comunicaciones en el cual cada sitio es un sistema de BD en sí mismo, y trabajan juntos con el fin de que un usuario pueda obtener acceso a los datos de cualquier punto de la red, como si estuvieran almacenados en el propio sitio del usuario.

El diseño de la BDD tiene como objetivo lograr un mejor desempeño; y requiere de su propia teoría, metodología, y herramientas de apoyo. El diseño de distribución incluye fragmentación y ubicación replicada o no de los fragmentos (Ceri et al., 1982). El término fragmentación se refiere a la división del esquema de una BDD de acuerdo con algún criterio, mientras que la replicación está relacionada con la existencia de copias de los fragmentos en varios sitios.

Los dos principios básicos de las BDD (Ceri et al., 1987, Ma et al., 2005, Őzsu and Valduriez, 1999) son reducir el intercambio de datos entre los sitios y eliminar datos irrelevantes en la ejecución de solicitudes. Ambos sirven de guía al proceso de diseño de BDD, el cual ha sido dividido en cuatro pasos fundamentales (Baião et al., 2002, Bellatreche et al., 2000, Ceri et al., 1987, Hababeh et al., 2004, Navathe et al., 1995, Őzsu and Valduriez, 1999, Schewe, 2002): (1) Diseño del esquema conceptual, (2) Diseño de la fragmentación, (3) Diseño de la asignación, y (4) Diseño de la BD física. Los problemas 1 y 4 son comunes a las BD centralizadas (BDC) y a las BDD, mientras que los problemas 2 y 3 caracterizan el diseño de BDD.

El diseño de un SBDD implica la toma de decisiones sobre la ubicación de los programas que accederán a la Base de Datos y sobre los propios datos que constituyen esta última, a través de los diferentes sitios de una red de computadoras.

## 1.2 Diseño lógico de distribución de datos

El diseño lógico se refiere a lo que hará el nuevo sistema, es una descripción de los requisitos funcionales de un sistema. En otras palabras, es la expresión conceptual de lo que hará el sistema para resolver los problemas identificados en el análisis previo. A falta de este paso, los aspectos técnicos del sistema, como los dispositivos de hardware

que deban adquirirse, con frecuencia afectan la solución. El diseño lógico incluye planear el propósito de cada elemento del sistema, sin relación con consideraciones de hardware y software.

El nivel conceptual describe la estructura lógica global de la base de datos mediante un modelo abstracto de datos comprensible por el SGBD. Se definen la descripción de atributos, de entidades, las conexiones y las restricciones de integridad asociadas a la semántica. Se puede decir que describe los datos que son almacenados realmente en la BD y las relaciones que existen entre los mismos, describe la BD completa en términos de su estructura de diseño.

El modelo lógico de las BDD fue formulado inicialmente por Ceri et al. en 1983 (Ceri et al., 1983), actualmente considerado como un modelo clásico consolidado. El mismo incluye el problema de cómo fragmentar y distribuir los datos óptimamente en los sitios de una red. Este problema tiene como principio clave alcanzar máxima localidad de los datos, ubicándolos tan cerca como sea posible de las aplicaciones que los utilizan, lo que permite reducir el tráfico de comunicaciones en la red. El diseño lógico se convierte en parte en la especificación funcional que se usa en el diseño físico. El diseño lógico es independiente de la tecnología.

Usualmente hay dos enfoques alternativos para el diseño de una BDD, uno es el descendente y el otro es el ascendente. La estrategia descendente es adecuada para el desarrollo inicial de un sistema de BDD sin tener restricciones de otros sistemas ya instalados y que deban ser integrados de alguna manera al sistema distribuido como corresponde a la estrategia ascendente. Bajo el enfoque descendente, en una primera fase los esquemas globales se dividen en subconjuntos llamados fragmentos; y en una segunda fase los fragmentos son ubicados en los diferentes sitios. Este trabajo está acorde con la estrategia descendente.

### 1.2.1 Fragmentación

Un sistema soporta fragmentación de datos si es posible dividir sus relaciones en pequeñas porciones o fragmentos. Existen, esencialmente, dos formas de realizar esta

división: horizontal o verticalmente. Existe otro tipo no básico de fragmentación que es generada por la combinación de fragmentos horizontales y verticales y es nombrada fragmentación mixta o híbrida. Aún cuando esta fragmentación no se considere un tipo primitivo, en muchas situaciones reales resulta imprescindible disponer de ella.

La fragmentación mixta puede realizarse de tres formas diferentes (véase la figura 1):

1. Desarrollando primero la fragmentación vertical y luego aplicar la fragmentación horizontal sobre los fragmentos verticales (llamada partición VH).
2. Desarrollando primero una fragmentación horizontal y luego aplicar la fragmentación vertical sobre los fragmentos horizontales (llamada partición HV).
3. Aplicando sobre una relación, de forma simultánea y no secuencial la fragmentación horizontal y vertical, generando una rejilla y los fragmentos formaran las celdas de dicha rejilla.

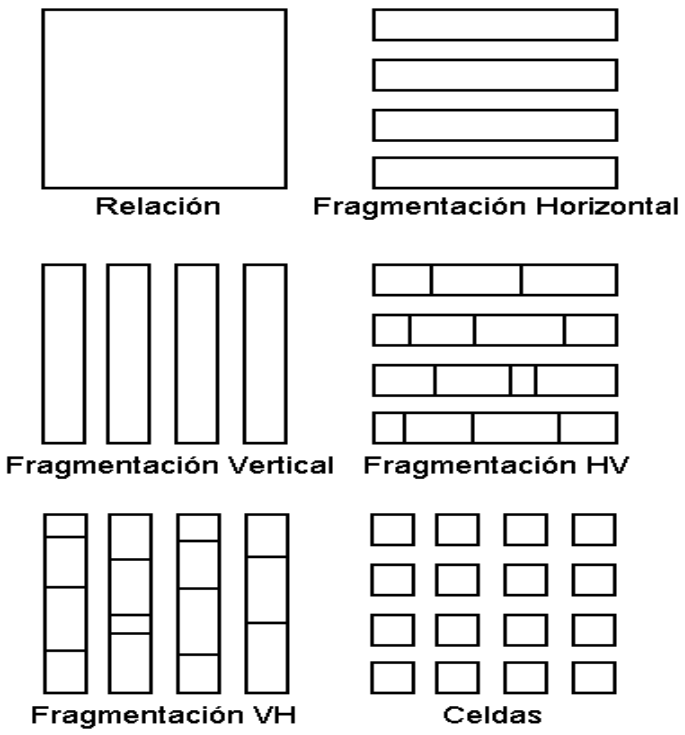



Figura 1. Variantes de la fragmentación.

Una decisión compleja e importante que afecta el rendimiento de las aplicaciones es determinar cuál relación debe ser fragmentada. Por supuesto que ésta no es una decisión arbitraria. En este sentido influyen determinados parámetros que deben caracterizar tanto a las aplicaciones como a la BD, pero es necesario resaltar que el grado de la fragmentación puede ir de un extremo en que no se fragmenta del todo, a otro donde se fragmenta en el ámbito de atributos individuales o al nivel de tuplas, para el caso de la fragmentación vertical y horizontal respectivamente.

Los efectos adversos de fragmentar unidades muy grandes pueden influir negativamente en relación con la réplica. Si el fragmento no es replicado se incrementan desmesuradamente los accesos remotos, y si es replicado en todos los sitios causa serios problemas con la actualización de los datos y es necesario incrementar la capacidad de almacenamiento de las computadoras personales. Por otra parte, si es necesario recuperar datos ubicados en más de un fragmento, la fragmentación de unidades muy pequeñas provoca el uso de operadores de acople lo cual es muy costoso, y además, el problema de la localización puede resultar inmanejable. Por tal motivo, el diseñador debe encontrar un nivel de fragmentación adecuado, que será un compromiso entre ambos extremos. Tal nivel puede ser definido con respecto a las características de las aplicaciones que se ejecutan sobre la base de datos.

El propósito del diseño de la fragmentación es determinar los fragmentos que constituyen unidades lógicas de asignación y consiste en agrupar tuplas o atributos con las mismas propiedades. Los fragmentos deben ser colecciones homogéneas de información desde el punto de vista de accesos de las aplicaciones, o sea, que todas las instancias de los fragmentos sean accedidas uniformemente por las aplicaciones que las usan.

El diseño de la fragmentación antecede al proceso de asignación de fragmentos a sitios de procesamiento, reflejando el criterio de mantener los datos locales en el sitio donde frecuentemente son accedidos por las aplicaciones; es por ello que los fragmentos constituyen una unidad apropiada de asignación. Si las aplicaciones que manipulan un esquema están ubicadas en sitios diferentes, se pueden seguir dos alternativas con

todo el esquema como unidad de distribución: replicado o no replicado (Bhalla and Hasegawa, 2005, Gançarski et al., 2002, Johansson et al., 2000).

Muchos de los problemas relacionados con la fragmentación y asignación en el diseño de BDD tienen una modelación matemática compleja y son considerados de la clase NP-Completo (Baião et al., 2003, Őzsu and Valduriez, 1999, Lee and Baik, 2004, Pérez et al., 2005), donde no hay garantía de encontrar una solución óptima con algoritmos determinísticos en un tiempo polinomial. Así, su complejidad es tal, que cualquier algoritmo que resuelve óptimamente cada uno de sus casos requiere un esfuerzo computacional que crece exponencialmente en función del tamaño del problema, en dependencia de la cantidad de fragmentos y de sitios.

Intentando dar solución a estos problemas se han presentado propuestas (Baião et al., 2004, Pérez et al., 2005, Savonnet et al., 1999, Tamhankar and Ram, 1998, Navathe et al., 1995, Mei and Sheng, 1992, Ceri et al., 1987, Ceri and Pernici, 1985) que sugieren el uso de heurísticas para algunos de ellos.

### 1.2.2 Ubicación

La ubicación de recursos a lo largo de una red de computadoras es un problema que ha sido estudiado extensivamente. Solo una pequeña proporción de los estudios realizados abarcan la distribución de datos de una BD (Bertone, 2004). El problema de la ubicación de los datos consiste en encontrar el esquema de distribución óptimo. La optimización puede ser definida en función de dos conceptos:

1. Costo mínimo: La definición de la función de costo consiste en evaluar cuánto costará, en tiempo de comunicación, consultar y modificar los datos en todos los lugares donde aparezcan. Adicionalmente deben considerarse otros factores como los que se señalan más abajo.
2. Desempeño: La estrategia de ubicación se diseña para mantener un determinado parámetro o nivel de desempeño del sistema. El desempeño se evalúa en función del tiempo de respuesta y del rendimiento del sistema en cada sitio de la red.

La construcción de un modelo que permita evaluar todos los aspectos que involucra la optimización del esquema de distribución es muy compleja. Las variables a tener en cuenta son varias y la incidencia de cada una de ellas también lo es, como por ejemplo el costo de replicar los fragmentos a lo largo de la red (Őzsu and Valduriez, 1991).

El modelo de ubicación de datos intenta, por tanto, minimizar el costo total de procesamiento y almacenamiento respetando las restricciones de tiempo establecidas. Algunas de las restricciones son tiempo máximo de ejecución, capacidad de almacenamiento, si bien este concepto ya no tiene la incidencia que tuvo anteriormente, y capacidad máxima de procesamiento. La ubicación de los datos es influenciada, además, por el esquema de replicación de información elegido.

En conjunto con este concepto, las técnicas que preservan integridad en la información y que aseguran la mejor actualización de los datos también inciden en la definición del mejor esquema. Como parte de este trabajo se presentan estudios realizados sobre los diferentes métodos y su incidencia en el comportamiento de la BDD.

## 1.3 Diseño físico de distribución de datos

El diseño físico es el proceso de escoger las estructuras de almacenamiento en disco y métodos de acceso a los datos más adecuada para lograr un buen rendimiento de la BD. En el momento del diseño físico es importante conocer la carga de trabajo, combinación de consultas y actualizaciones, que debe soportar la BD y los requerimientos del usuario. También es importante que el diseñador conozca las técnicas de procesamiento de consultas e indexación soportadas por el SGBD.

La clave de un buen diseño físico es una correcta descripción de la carga de trabajo: lista de consultas y actualizaciones, indicando sus frecuencias de operación y el resultado esperado. Para cada consulta es necesario indicar las relaciones a las que se accede, los atributos de salida y los que intervienen en filtros y condiciones. Igualmente para las actualizaciones deben conocerse los atributos sobre los que se expresan condiciones y el tipo de actualización y la relación y atributos actualizados.

El diseño físico traduce el diseño lógico en una solución implementable y económicamente efectiva. El diseño físico es el proceso de producir la descripción de la implementación de la BD en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos. Entre el diseño físico y el diseño lógico hay una realimentación, ya que alguna de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

El diseño físico está íntimamente ligado a una alternativa tecnológica. Ante la acelerada evolución tecnológica es importante considerar los estándares del momento y las tendencias ya que una mala decisión tendrá, inevitablemente, una implicación en los costos.

## 1.4 Generación de esquemas distribuidos en SQL Server

A continuación se analizan los aspectos relacionados con la distribución y replicación en Microsoft SQL Server, que es uno de los SGBD más usados en Cuba y para el cual se plantea una solución de generación de esquemas físicos mediante replicación.

### 1.4.1 Distribución versus replicación

Como se ha podido entender, los datos se distribuyen por las siguientes razones: mejoras del desempeño puesto que cada servidor solamente está manejando los datos asociados con ese servidor y las tablas son tan compactas como sea posible o por reducción de costos donde muchos servidores pequeños son menos costosos de mantener que un único servidor monolítico. Un único servidor representa un único punto de fracaso. Si un servidor falla en un escenario de servidores múltiples, los otros servidores pueden continuar con el servicio en sus ubicaciones.

Por otra parte, los datos se replican por las siguientes razones: para apoyar la distribución de otros datos ya que en muchas ocasiones los datos de tablas distribuidas pueden contener llaves de tablas replicadas, para mantener la integridad referencial

donde las tablas de apoyo deben ser replicadas al sitio de datos distribuidos, para permitir la accesibilidad a los mismos datos porque puede ser deseable replicar resúmenes de información a cada servidor y estas estadísticas pueden ser generadas desde cualquier servidor.

### 1.4.2 Replicación en Microsoft SQL Server

La replicación de datos permite que cierta información de la BD sea almacenada en más de un sitio y consiste en el transporte de esta entre dos o más servidores, permitiendo que ciertos datos de la BD estén almacenados en más de un sitio, y así aumentar la disponibilidad de los mismos y mejorar el rendimiento de las consultas globales (Morell, 2004).

La replicación en SQL Server consiste, en el transporte de datos entre dos o más instancias de servidores. Para ello SQL Server brinda un conjunto de soluciones que permite copiar, distribuir y posiblemente modificar datos de toda la organización. Se incluyen, además, varios métodos y opciones para el diseño, implementación, supervisión y administración de la replicación, que le ofrecen la funcionalidad y flexibilidad necesarias para distribuir datos y mantener su coherencia (Morell, 2004).

El modelo de replicación en SQL Server toma como referencia la metáfora de la industria de las publicaciones; está formado por: publicador, distribuidor, suscriptor, publicación, artículo y suscripción; y varios agentes responsabilizados de copiar los datos entre el publicador y el suscriptor. Estos agentes son: agente de instantáneas, agente de distribución, agente del lector del registro, agente del lector de cola y agente de mezcla (Morell, 2004). A los tipos básicos de replicación de instantáneas, transaccional y de mezcla se le incorporan opciones para ajustarse aún más a los requerimientos del usuario. A continuación se explican en detalle cada uno de estos componentes.

#### *1.4.2.1* Componentes del modelo de replicación

El publicador es un servidor que pone los datos a disposición de otros servidores para poder replicarlos. El distribuidor es un servidor que almacena la BD de distribución y almacena los datos históricos, transacciones y metadatos. Los suscriptores reciben los datos replicados.

Una publicación es un conjunto de artículos de una publicación de una BD. Esta agrupación de varios artículos facilita especificar un conjunto de datos relacionados lógicamente y los objetos de BD que desea replicar conjuntamente. Un artículo de una publicación puede ser una tabla de datos, la cual puede contar con todas las filas o algunas (filtrado horizontal) y simultáneamente contar de todas las columnas o algunas (filtrado vertical), un procedimiento almacenado, una definición de vista, la ejecución de un procedimiento almacenado, una vista, una vista indizada o una función definida por el usuario.

Una suscripción es una petición de copia de datos o de objetos de BD a replicar. Una suscripción define qué publicación se recibirá, dónde y cuándo. Las suscripciones pueden ser de inserción o de extracción; y una publicación puede admitir una combinación de suscripciones de inserción y extracción. Con una subscripción de inserción el Publicador propaga los cambios al Subscriptor sin petición del Subscriptor, los cambios pueden ser insertados bajo demanda, continuamente o bajo un horario programado, el agente de distribución o el agente de mezcla se ejecutan en el Distribuidor; se utiliza cuando las publicaciones requieren movimientos de datos en tiempo real, o cuando estos deban ser sincronizados de manera continua o bajo un repetido esquema programado, más comúnmente usada con replicación de instantáneas y replicación de transacciones. Con la subscripciones de extracción el Subscriptor solicita los cambios hechos en el Publicador, este permite a los usuarios en el Subscriptor determinar cuándo se sincronizarán los datos modificados, los agentes de distribución o de mezcla se ejecutan en el Subscritor. Se usa cuando los datos son sincronizados bajo demanda o a un horario programado de manera no tan continua, cuando las publicaciones tienen un gran número de subscriptores, habitualmente usado

con replicaciones de mezcla**.** El publicador (en las suscripciones de inserción) o el suscriptor (en las suscripciones de extracción) solicitan la sincronización o distribución de datos de una suscripción.

El publicador puede disponer de una o más publicaciones, de las cuales los suscriptores se suscriben a las publicaciones que necesitan, nunca a artículos individuales de una publicación. El publicador, además, detecta qué datos han cambiado durante la replicación transaccional y mantiene información acerca de todas las publicaciones del sitio.

La función del distribuidor varía según la metodología de replicación implementada. En ocasiones se configura como distribuidor el mismo publicador y se le denomina distribuidor local. En el resto de los casos el distribuidor será remoto, pudiendo coincidir en algún caso con un suscriptor.

Los suscriptores además de obtener sus suscripciones, en dependencia del tipo y opciones de replicación elegidas, pueden devolver datos modificados al publicador. Además puede tener sus propias publicaciones (Morell, 2004).

#### *1.4.2.2* Escenarios típicos de la replicación

En una solución de replicación pudiera ser necesario utilizar varias publicaciones en una combinación de metodologías y opciones. En la replicación los datos o transacciones fluyen del publicador al suscriptor pasando por el distribuidor. Por lo tanto, en su configuración mínima, una topología de replicación se compone de al menos dos o tres servidores SQL Server, que desempeñan los tres roles mencionados.

Variando la ubicación del servidor distribuidor se pudiera contar con las siguientes variantes:

1. El rol de distribuidor desempeñado por el publicador (véase la figura 2).
2. El rol de distribuidor desempeñado por el suscriptor (véase la figura 3).
3. Un servidor de distribución, independiente del publicador y del suscriptor (véase la figura 4).


Figura 2.Publicador-Distribuidor

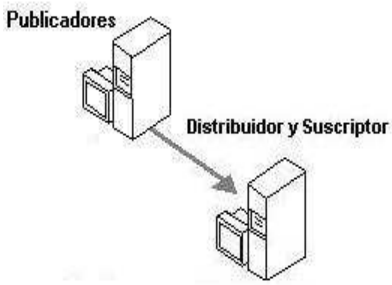



Figura 3. Distribuidor-Suscriptor

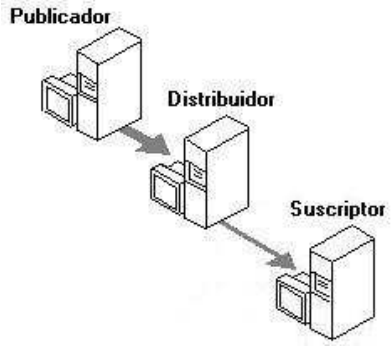



Figura 4.Distribuidor independiente

En la mayoría de las configuraciones, el peso fundamental de la replicación recae sobre el servidor de distribución. Por tanto, esto puede ser un criterio para determinar su ubicación, teniendo en cuenta las configuraciones o posibilidades físicas de los servidores, así como otras responsabilidades que pueden estar desempeñando: servidor de dominio, servidor de páginas Web entre otras (Morell, 2004).Existe la posibilidad de contar con un servidor que se suscriba a una publicación y a la vez la publique para el resto de los suscriptores; esto puede ser muy útil cuando se cuente con una conexión muy costosa con el publicador principal. Por ejemplo, el publicador principal en Madrid y los subscriptores en Ciudad Habana, Varadero, Cayo Coco, Cayo Largo, etc. En casos como este, se puede elegir un subscriptor como el servidor de Ciudad Habana el cual se suscribe al publicador en Madrid y a la vez actúa como servidor de publicación para los servidores de Varadero, Cayo Coco, Cayo Largo y demás. Evidentemente, en una configuración tal, pueden nuevamente combinarse la ubicación de los dos distribuidores y aumentar el número de variantes que pueden presentarse, pero las consideraciones para determinar la ubicación del servidor que fungirá como distribuidor son las ya mencionadas.

#### *1.4.2.3* Tipos de replicación

Replicación de instantáneas

En la replicación de instantáneas los datos se copian tal y como aparecen exactamente en un momento determinado. Por consiguiente, no requiere un control continuo de los cambios. Las publicaciones de instantáneas se suelen replicar con menos frecuencia que otros tipos de publicaciones y puede llevar más tiempo propagar las modificaciones de datos a los suscriptores.

Se recomienda utilizar este tipo de replicación cuando la mayoría de los datos no cambian con frecuencia, cuando se replican pequeñas cantidades de datos, cuando los sitios con frecuencia están desconectados y es aceptable un período de latencia largo, o sea, la cantidad de tiempo que transcurre entre la actualización de los datos en un sitio y en otro.

En ocasiones se hace necesario utilizarla cuando están involucrados algunos tipos de datos (text, ntext, e image) cuyas modificaciones no se registran en el registro de transacciones y por tanto no se pueden replicar utilizando la metodología de replicación transaccional (Morell, 2004).

Con la opción de actualización inmediata en el suscriptor, se permite a los suscriptores actualizar datos solamente si el publicador los va a aceptar inmediatamente. Si el publicador los acepta, se propagan a otros suscriptores. El suscriptor debe estar conectado de forma estable y continua al publicador, para poder realizar cambios en el suscriptor. Esta opción es útil en escenarios en los que tienen lugar unas cuantas modificaciones ocasionales en los servidores suscriptores.

Replicación transaccional:

En este caso se propaga una instantánea inicial de datos a los suscriptores, y después, cuando se efectúan las modificaciones en el publicador, las transacciones individuales se propagan a los suscriptores. SQL Server almacena las transacciones que afectan a los objetos replicados y propaga esos cambios a los suscriptores de forma continua o a intervalos programados. Al finalizar la propagación de los cambios, todos los suscriptores tendrán los mismos valores que el publicador.

Este tipo de replicación suele utilizarse cuando se desea que las modificaciones de datos se propaguen a los suscriptores normalmente pocos segundos después de producirse; se necesita que las transacciones sean atómicas, que se apliquen todas o ninguna al suscriptor; los suscriptores se conectan en su mayoría al publicador; su aplicación no puede permitir un período de latencia largo para los suscriptores que reciban cambios.

Esta es útil en escenarios en los que los suscriptores pueden tratar a sus datos como de sólo lectura, pero necesitan cambios a los datos con una cantidad mínima de latencia. Con el uso de la opción de actualización inmediata en el suscriptor se pierde aún más la autonomía de sitio, pero se reduce el tiempo en el cual los sitios actualizan sus copias de los datos. Para hacer modificaciones en la BD del suscriptor éstas se realizan o intentan también en la base de datos publicador en una confirmación de dos fases (2PC) por lo que si su modificación se confirma, indica que es válida y luego, en cuestión de minutos, o según la planificación hecha, estos cambios son duplicados a las demás BD suscriptoras.

Replicación de mezcla:

Esta permite que varios sitios funcionen en línea o desconectados de manera autónoma, y mezclar más adelante las modificaciones de datos realizadas en un resultado único y uniforme. La instantánea inicial se aplica a los suscriptores. A continuación SQL Server hace un seguimiento de los cambios realizados en los datos publicados en el publicador y en los suscriptores. Los datos se sincronizan entre los servidores a una hora programada o a petición. Las actualizaciones se realizan de manera independiente, sin protocolo de confirmación en más de un servidor; así el publicador o más de un suscriptor pueden haber actualizado los mismos datos.

Por lo tanto, pueden producirse conflictos al mezclar las modificaciones de datos. Cuando se produce un conflicto, el agente de mezcla invoca una resolución para determinar qué datos se aceptarán y se propagarán a otros sitios.

Este tipo de replicación es útil cuando varios suscriptores necesitan actualizar datos en diferentes ocasiones y propagar los cambios al publicador y a otros suscriptores, cuando los suscriptores necesitan recibir datos, cuando se necesita realizar cambios sin

conexión y sincronizar más adelante los cambios con el publicador y otros suscriptores y cuando la autonomía del sitio es un factor crucial.

La misma es útil en ambientes en los que cada sitio hace cambios solamente en sus datos pero que necesitan tener la información de los otros sitios. Para ajustarse aún más a los requerimientos de los usuarios se incorporan opciones como son la actualización inmediata en el suscriptor, la actualización en cola y la transformación de datos replicados (Morell, 2004).

#### *1.4.2.4* Factores para elegir el método de replicación a utilizar

En la elección de un método adecuado para la distribución de los datos en una organización influyen varios factores, los cuales se pueden agrupar en dos: factores relacionados con los requerimientos de la aplicación y factores relacionados con el entorno de red.

Dentro de los factores relacionados con los requerimientos de la aplicación, los fundamentales son (Morell, 2004): autonomía, consistencia transaccional y latencia.

La autonomía de un sitio da la medida de cuanto puede operar el sitio desconectado de la base de datos publicadora. La consistencia transaccional de un sitio viene dado por la necesidad de ejecutar o no inmediatamente todas las transacciones que se han ejecutado en el servidor, o si es suficiente con respetar el orden de las mismas. La latencia de un sitio se refiere al momento en que se deben sincronizar las copias de los datos (Morell, 2004).

Entre los factores relacionados con el entorno de red están la velocidad de transmisión de datos de la red y la confiabilidad de la misma. Por otra parte en el caso que los servidores SQL Server no permanezcan todo el día encendido, como pudiera suceder en algunas organizaciones, deben considerarse los horarios de disponibilidad de cada servidor (Morell, 2004).

La consideración de estos factores sirve de guía en la configuración del ambiente de replicación. Además debe considerar las siguientes preguntas: ¿Qué datos se van a publicar? ¿Reciben todos los suscriptores todos los datos o sólo subconjuntos de ellos?

¿Se deben particionar los datos por sitio? ¿Se debe permitir que los suscriptores envíen actualizaciones de los datos? Y en caso de permitirlas ¿Cómo deben implementarse? ¿Quiénes pueden tener acceso a los datos? ¿Se encuentran estos usuarios en línea? ¿Se encuentran conectados mediante enlaces caros?

#### *1.4.2.5* Fases generales para implementar y supervisar la replicación

A pesar de que existen varias formas de implementar y supervisar la replicación, el proceso de replicación es diferente según el tipo y las opciones elegidas. En general, la replicación se compone de las siguientes fases:

1. Configuración de la replicación.
2. Generación y aplicación de la instantánea inicial.
3. Modificación de los datos replicados.
4. Sincronización y propagación de los datos.

## 1.5 Consideraciones parciales

La configuración de la replicación en el diseño físico de BDD es un elemento importante a tener en cuenta. La topología más apropiada será configurar un servidor Distribuidor-Publicador y otro Subscriptor. La variante la replicación de mezcla es la más conveniente por las ventajas que ofrece, sobre todo estimula la autonomía como medida de cuánto puede operar el sitio desconectado de la BD publicadora. En este tipo de replicación se puede implementar la fragmentación a través del filtrado de datos horizontal y vertical. Se aplica el tipo de subscripción de extracción debido a que los datos serán típicamente sincronizados por petición o por una hora programada. Los subscriptores determinarán cuando se conectarán y sincronizarán los cambios.

# Capítulo 2. Diseño de la distribución física de datos mediante replicación en SQL Server

En este capítulo se abordan algunos aspectos relacionados con el desarrollo de la herramienta DISTRIBUTOR que se encarga de la generación de scripts para la creación de BDD mediante replicación en el SGBD Microsoft SQL Server. Para esto se utilizan algunos recursos del lenguaje UML.

## 2.1 Diseño de la herramienta DISTRIBUTOR

A continuación se detallan los elementos de análisis y diseño de la herramienta DISTRIBUTOR para la generación de esquemas físicos en SQL Server usando la notación de UML.

### 2.1.1 Diagrama de Clases

Un diagrama de clases es un esquema, patrón o plantilla para describir muchas instancias de datos posibles, que describen clases de objetos. Un diagrama de clases dado se corresponde con un conjunto infinito de diagramas de instancia.
Este tipo de diagrama muestra la relación existente entre las seis clases de este trabajo, sus nombres, atributos y métodos (véase la figura 5). La clase MainForm es la clase principal. Esta clase tiene servicios que utilizan los ofrecidos por las demás clases con las está asociada: StringFunctions, XMLReader, DBNameForm, FileHandler.


Figura 5. Diagrama de clases de DISTRIBUTOR.

### 2.1.2 Diagrama de Secuencia

El objetivo de este diagrama es indicar el orden temporal en el que se ejecutan los pases de mensajes entre las diferentes clases. Cada línea de vida está representada por una clase. El software comienza a ejecutarse por la clase MainForm que muestra la interfaz gráfica con la que el usuario va a trabajar. Primeramente se selecciona el archivo XML con el que se va a trabajar ya que esta herramienta toma como entrada una solución genérica en ese formato mediante el método openFileDialog. Este archivo es generado por servicios soportados en una biblioteca de enlace dinámico (DLL) que se desarrolla en Modelo genérico para la generación de esquemas físicos en el diseño de bases de datos distribuidas. Esta DLL posee servicios para la creación y validación de los archivos XML que sirven de entrada a la herramienta DISTRIBUTOR que es el resultado final de esta tesis. Una vez seleccionado el XML se realiza una copia del mismo en el directorio de trabajo con nombre Catálogo.txt para el trabajo temporal. Seguidamente se muestra la forma principal MainForm con páginas para cada sitio registrado en el catálogo; posteriormente el usuario debe elegir la opción de generar script; cuando esto se realiza se abre el archivo Catalogo.txt, se lee secuencialmente y se muestra la sentencia SQL correspondiente a esa línea.

Este método se auxilia de otros como configurarDistribuidor, createHorizontalPublication, createSubcriptions. También se auxilia de StringFunctions para obtener la información relevante de cada línea. Una vez terminada la generación del script, el usuario puede determinar guardarlo en la dirección escogida por él mismo. (Véase la figura 6).

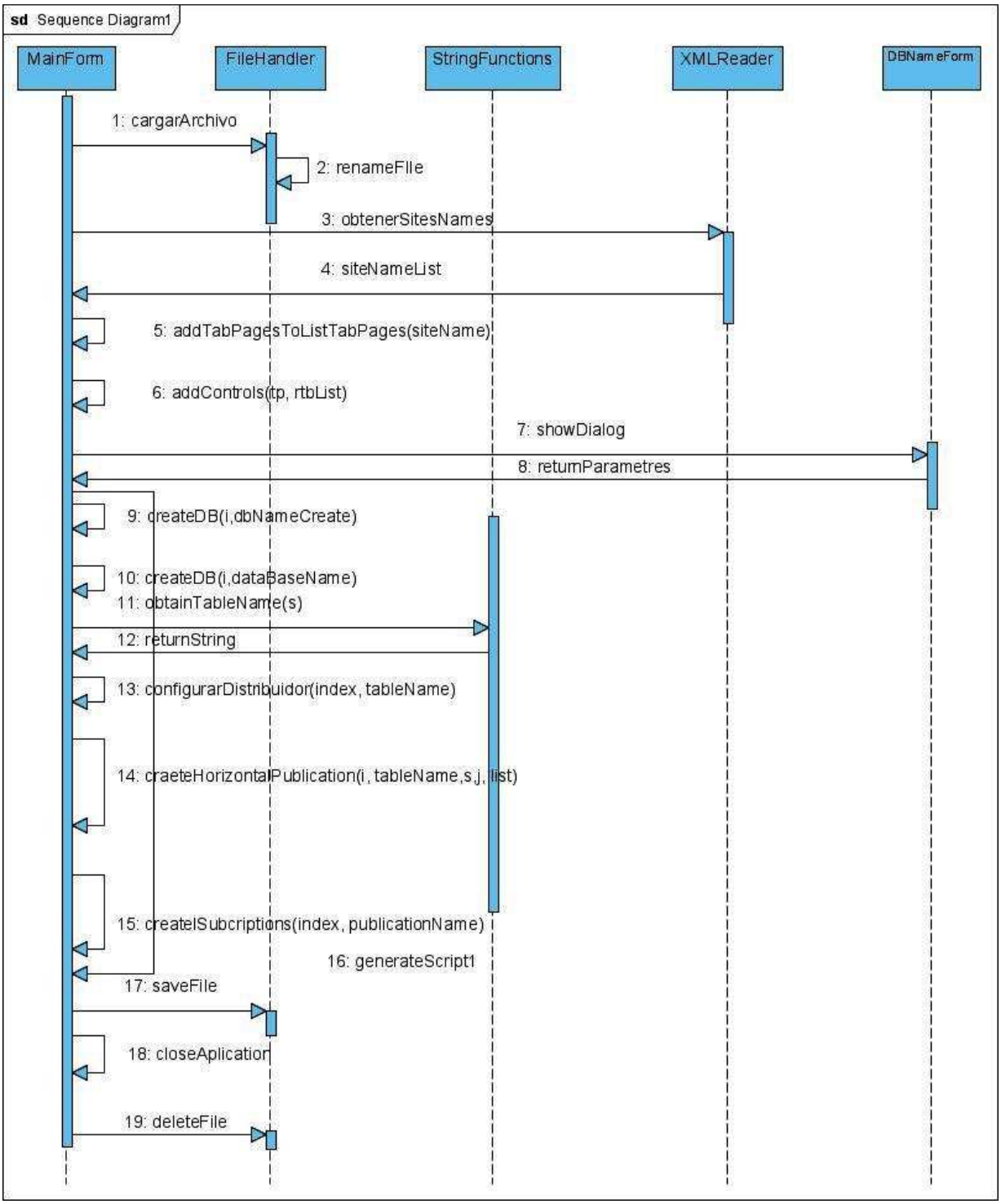



Figura 6. Diagrama de secuencia de DISTRIBUTOR.

## 2.2 Implementación

A continuación se explican brevemente las clases que están implementadas en el software, así como los métodos de cada una de ellas.

**Clase DBNameForm**

Esta clase se utiliza para recoger algunos parámetros que se necesitan para la configuración de la replicación como: nombre de las instancias del los servidores de publicación y subscripción, nombre de la base de datos de publicación y subscripción. Esta clase tiene un atributo por cada parámetro que necesita y utiliza una propiedad (property) para manejar estos valores.

**Clase FileHandler**

De manera general, esta clase se encarga de manipular lo referente al trabajo con archivos. Esta carga el archivo XML y crea un archivo temporal llamado catálogo.txt utilizado para la generación del script.

Métodos:

openFileToRead(): crea un objeto de tipo StreamReader. El constructor de esta clase tiene un parámetro con la dirección del archivo del que se va a leer. El objeto creado es el encargado de leer el archivo.

closeFile(): cierra el archivo.

deleteFile(): elimina el archivo catálogo.txt después que deja de ser útil.

cargarArchivo(): muestra un diálogo para seleccionar el archivo XML a cargar y pone el camino seleccionado en el atributo nombreArchivo; para ello usa la propiedad nombre_Archivo.

renameFile(): hace una copia del archivo.xml pero con la extensión .txt y lo pone en el directorio de trabajo para poder crear el script a partir de este archivo.

readOneLine(): lee una línea del documento y pone el cursor en la línea siguiente.

createDirectory(): crea un directorio en Mis documentos con el nombre MSSQL.

**Clase MainForm**

Esta es la clase principal del software que muestra el script generado para cada sitio en una interfaz gráfica al usuario con una colección de pestañas (TabPages). Para esto se

utilizan diferentes métodos. Primeramente se muestra la ventana con una lista de pestañas (TabPages) que tienen como nombre los nombres de los sitios (SitesNames) leídos del archivo XML. Esto se hace a través de los siguientes métodos:

addTabPagesToListTabPages(List<string>s): crea una pestaña (TabPages) y un cuadro de texto (RitchTextBox) por cada sitio; esto lo hace a partir del método obtener de la clase XmlReader que devuelve una lista con todos los sitios del XML la cual se pasa por parámetro a este método.

addControls(List<TabPage>tp, List<RichTextBox>rt): agrega el contenedor de pestañas (TabControl), a la ventana principal (Form); pone las propiedades necesarias para mostrarla y le agrega una pestaña (TabPages) por sitio que a su vez contiene un cuadro de texto (RitchTextBox), cada uno con un conjunto de propiedades para que se muestre en el contenedor de pestañas (TabControl).

openFile():este método se invoca desde la opción Abrir archivo XML de la barra de herramientas o desde el menú Archivo. Básicamente hace una llamada a los métodos cargarArchivo, obtener, addTabPagesToListTabPages, addControls y como resultado final se muestra el área de generación de script de la ventana.

generateScript1(): este método es el encargado de generar el script que se muestra en el cuadro de texto (RitchTextBox); esto se hace recorriendo el archivo catálago.txt y escribiendo la sentencia en Transact-SQL(T-SQL) correspondiente a cada línea leída. Este método utiliza fundamentalmente los métodos de la clase StringFunctions para obtener la información de cada línea leída y el método appendTextToRitchTextBox que imprime en el cuadro de texto (RitchTextBox) una cadena pasada como parámetro. También utiliza otros métodos cómo configurarDistribuidor, createHorizontalPublication, createSubcriptions.

obtainAEntireSite(string nameSite): obtiene todos las líneas de un sitio pasado por parámetro.

saveScript(): crea un archivo .txt por cada sitio con el script generado y lo guarda en la dirección seleccionada por el usuario a través de FolderBrowserDialog.

closeApplication(): cierra la aplicación y elimina el archivo catálogo.txt.

**Clase StringFunctions**

Esta clase cuenta con un conjunto de métodos estáticos por lo que no se necesita crear un objeto de la clase para poder utilizar sus métodos implementados para el trabajo con cadenas.

Métodos:

obtainTableName(string s): devuelve el nombre de una tabla a partir de la cadena <SchemaName>nombre-de-tabla</SchemaName>

obtainAttributeName(string s): devuelve el nombre de un atributo a partir de la cadena <AttName>nombre-de-atributo</AttName>

obtainAttributeNamePrimaryKey(string s): devuelve el nombre de un atributo en la llave primaria de una tabla a partir de la cadena <AttributePK> nombre-de-atributo-llave-primaria </AttributePK>.

obtainAttributeType(string s): devuelve el tipo de un atributo a partir de la cadena<AttType>tipo-de-atributo</AttType>.

obtainAttributeSize(string s): devuelve el tamaño de un atributo a partir de la cadena <AttSize>tamaño-de-atributo</AttSize>.

obtainFKRefTable(string s): obtiene el nombre de la tabla a la que hace referencia la llave foránea a partir de la cadena <RefTable> tabla-propietaria</RefTable>.

obtainFKOwner(string s): obtiene el propietario de la llave foránea a partir de la cadena <AttributeOwnerFK>atributo-llave-tabla-propietaria</AttributeOwnerFK>.

obtainSiteName(string s): devuelve el nombre del sitio a partir de la cadena <SiteName>sitio</SiteName>.

obtainHorizontalExpression(string s): devuelve el valor de la expresión horizontal a partir de la cadena <ExpressionHorizontal>minterm</ExpressionHorizontal>.

obtainDerivedExpression(string s): devuelve el valor de la expresión derivada a partir de la cadena <ExpressionDerivada>tabla-propietaria </ExpressionDerivada>.

obtainAttributeVerticalExpression(string s): devuelve el valor del atributo de la expresión vertical a partir de la cadena <AttributteVertical>atributo-proyectado</AttributteVertical>.

obtainHorizontalExpressionTerm(string s): devuelve el valor del atributo del término de la expresión vertical.

deleteLastColon(string s): elimina la última coma de una línea en la creación de las llaves primarias. Esto es necesario acorde con la sintaxis del T-SQL.

deleteLastColonFK(string s): elimina la última coma de una línea en la creación de las llaves foráneas. Esto es necesario acorde con la sintaxis del T-SQL.

**Clase XMLReader**

En esta clase se lee el archivo XML y se obtienen todos los nombres de los sitios donde se ubicarán los datos. Este nombre se utiliza para crear una pestaña (TabPages) por sitio.

Método:

obtener(stringns, string s, stringpath): este método crea un iterador que recorre el archivo XML guardando los nombres de todos los sitios en una lista. Este recibe como parámetro el sitio (Site), el nombre del sitio (SiteName) y el nombre del archivo. El nombre es la dirección completa incluyendo el camino.

## 2.3 Generación de scripts en SQL Server 2008

Anterior a DISTRIBUTOR se contaba con un software que se encargaba de la asignación de fragmentos a los diferentes sitios de procesamiento identificados en la red donde se ubicará la BDD diseñada. Se decidió despojar a ALLOCATOR de la responsabilidad de generar esquemas físicos y dejarle solamente la de realizar la asignación lógica de fragmentos a sitios. De esta forma, se podrán generar soluciones independientes para múltiples SGBD sin necesidad de dar mantenimiento a ALLOCATOR, mediante la creación de las mismas de forma independiente y acoplándolas con SIADBDD de forma dinámica sin necesidad de recompilar, evitando así problemas de integración.

En este acápite se explica la forma en que se generan los scripts que van a ser usados por Microsoft SQL Server 2008 para la configuración de la replicación. Para esto la herramienta DISTRIBUTOR comienza con la lectura de un archivo XML anteriormente validado. Posteriormente este archivo se manipula con los métodos descritos anteriormente en el acápite de Implementación. Una vez que se ha obtenido la información necesaria de este archivo, se comienzan a generar los scripts y se colocan en pestañas (TabPages) con el título del nombre del sitio donde se ubicará cada script.

Estos scripts crean las tablas cuyas definiciones se obtienen del archivo XML, incluyendo las definiciones de sus atributos llaves y descriptores. Una vez terminada la creación de las tablas se pasa a configurar la replicación la cual se explica a continuación:

Para configurar la replicación en Microsoft SQL Server se inicia con la configuración del servidor de distribución, luego el de publicación, y a continuación el subscriptor. Para este trabajo se escogió la arquitectura de un servidor Distribuidor-Publicador por lo que una vez acabada la configuración del distribuidor se configura la publicación en el mismo, luego en el servidor de subscripción se configura una subscripción para la publicación anterior.

Una vez creada la instantánea inicial por el agente de la publicación (SnapshotAgent) se trata de establecer una sincronización por el agente de mezcla de la subscripción (MergeAgent). Luego de realizados estos procesos se replica lo filtrado por el publicador hacia el sitio donde defina el subscriptor.

A continuación se explican los procedimientos almacenados fundamentales utilizados:

Distribuidor:

sp_adddistributor (T-SQL): configura el servidor como un distribuidor al que se le pasa como parámetro el nombre del servidor de distribución. En este caso se configuró como el nombre del sitio y el nombre de la instancia del servidor (por ejemplo: WOLF\SQLServerMyInsta). Los otros parámetros que son opcionales se dejaron con el valor por defecto.

sp_adddistributiondb (T-SQL): crea una nueva BD de distribución e instala el esquema del distribuidor. La BD de distribución almacena procedimientos, esquemas y metadatos usados en la replicación. Este procedimiento almacenado se ejecuta en el distribuidor sobre la BD master en orden para crear la BD de distribución, e instala las tablas necesarias y procedimientos almacenados requeridos para habilitar la distribución de la replicación. Se le pasa el nombre de la BD de distribución como parámetro. En este caso se configuró con el nombre estático “distributor”, entre otros parámetros que son opcionales y se dejaron con el valor por defecto, como por ejemplo el @security_mode

que tiene valor 1 que significa que el modo de autentificación usado va a ser el de Windows.

sp_adddistpublisher (T-SQL): configura un publicador para usar una BD de distribución específica. Este procedimiento almacenado se ejecuta en el distribuidor sobre cualquier BD. Es importante señalar que los procedimientos almacenados sp_adddistributor (T-SQL) y sp_adddistributiondb (T-SQL) tienen que ser ejecutados antes de este; se le pasan como parámetros el nombre del servidor de publicación (Publisher) que en este caso coincidirá con el servidor de distribución ya que se está configurando un Distribuidor-Publicador; además del nombre de la BD de distribución, el @security_mode se mantiene en 1 debido a la autentificación. Es importante señalar que el @working_directory con que se va a trabajar es el nombre de la computadora seguido del directorio repldata (por ejemplo: \\WOLF\repldata).

Publicación:

sp_replicationdboption(T-SQL): pone las opciones de la BD de replicación para la BD especificada. Este procedimiento almacenado se ejecuta en el publicador o subscriptor en cualquier BD; se le pasan como parámetros: @dbname que es el nombre de la base de datos de la que se va a publicar, @optname que se le especifica la opción mergepublishlo cual significa que la BD puede ser usada para publicaciones de mezcla. Por último se pone @value=true porque si el valor se pone falso y además el @optname= mergepublish significa que las subscripciones de las BD de publicación de mezcla son además eliminadas.

sp_addmergepublication (T-SQL): crea una nueva publicación de mezcla. Este procedimiento almacenado es ejecutado por el publicador en la BD que está siendo publicada; se le pasan como parámetros el nombre de la publicación, el resto se puede dejar con el valor por defecto, aclarando que al menos el @allow_pull=true debido a que la subscripción que se va a utilizar es de tipo pull.

sp_addpublication_snapshot (T-SQL): crea el agente de instantánea para una publicación; se le pasan como parámetros el nombre de la publicación y otro conjunto de entrada que se dejan con el valor por defecto para cada uno.

sp_addmergearticle (T-SQL): añade un artículo a una publicación de mezcla existente. Este procedimiento almacenado es ejecutado por el publicador en la BD de publicación; se le pasan como parámetros el nombre de la publicación, nombre del articulo @article, objeto de la BD que va a ser publicada @source_object, entre otros que se mantuvieron con el valor por defecto.

Subscripción:

sp_addmergesubscription (T-SQL): crea una subscripción de mezcla de tipo push o pull. Este procedimiento almacenado se ejecuta por el publicador en la BD de publicación; se le pasan como parámetros el nombre de la publicación, el @subscriber que es el nombre del servidor de subscripción, @subscriber_db que es el nombre de la BD de subscripción, @subscription_type que es el tipo de subscripción a utilizar el cual será pull, entre otros que se mantendrán con los valores por defecto.

sp_addmergepullsubscription (T-SQL): agrega una subscripción de extracción (pull) a una publicación de mezcla. Este procedimiento almacenado se ejecuta en el subscriptor sobre la BD de subscripción; se le pasan como parámetros @publication que es el nombre de la publicación, @publisher que es el nombre del publicador, @publisher_db que es el nombre de la BD del publicador y otros con sus valores por defecto.

sp_addmergepullsubscription_agent (T-SQL): adiciona un nuevo agente de trabajo usando el esquema de sincronización de subscripciones de extracción (pull) para publicaciones de mezcla. Este procedimiento se ejecuta por el subscriptor sobre la BD de subscripción; se le pasan como parámetros @publisher que es el nombre del publicador, @publisher_db que es el nombre de la BD de publicación, @publication que es el nombre de la publicación, @distributor que es el nombre del distribuidor, entre otros.

De esta manera queda creado el script para cada sitio. Una vez visualizados en la ventana de la aplicación DISTRIBUTOR, se pueden ejecutar en Microsoft SQL Server y efectuar así la replicación.

## 2.4 Conclusiones parciales

En este capítulo se describieron algunas de las fases del desarrollo del software DISTRIBUTOR; puntualizándose en las partes del diseño de mayor interés como son los diagramas de clases y de secuencia.

En el acápite de Implementación se explicaron todas las clases del proyecto, así como la función de cada uno de sus métodos.

Tomando en consideración los resultados derivados del seguimiento de estos pasos se logró el diseño e implementación de un asistente para la creación de los fragmentos físicos de BDD para el SGBD SQL Server.

# Capítulo 3. Aplicación para generar esquemas físicos distribuidos en SQL Server

En este capítulo se muestra una guía para el uso de la herramienta DISTRIBUTOR. Se enfatiza en el uso de los diferentes elementos presentes en la interfaz gráfica de usuario.

## 3.1 Presentación del Distributor

Este software cuenta con una ventana principal y otras secundarias. Es completamente operable como un editor de texto que posee varios menús como Archivo, Editar y Ayuda.

El menú Archivo brinda las opciones de:

- Abrir archivo XML (Ctrl+A): Abre un archivo XML.
- Guardar Script (Ctrl+G): Guarda en una carpeta seleccionada varios archivos en texto plano, uno por cada pestaña generada en el software.
- Generar Script (Ctrl+S): Genera el script SQL que posteriormente va a ser utilizado en el Microsoft SQL Server.
- Salir: Cierra o termina la aplicación.

El menú Editar que muestra las opciones:

- Deshacer (Ctrl+Z): Revierte el último cambio realizado en el script.
- Rehacer (Ctrl+Y): Rehace el último cambio realizado.
- Cortar (Ctrl+X): Corta el texto seleccionado en el editor.
- Copiar (Ctrl+C): Copia el texto seleccionado en el editor.
- Pegar (Ctrl+V): Pega el texto anteriormente cortado o copiado en el lugar seleccionado.
- Seleccionar Todo (Ctrl+E): Selecciona todo lo que está en la pestaña.

El menú Ayuda ofrece la opción Acerca de…para mostrar información acerca del DISTRIBUTOR.

En la figura 7 se muestra la ventana principal del software DISTRIBUTOR.

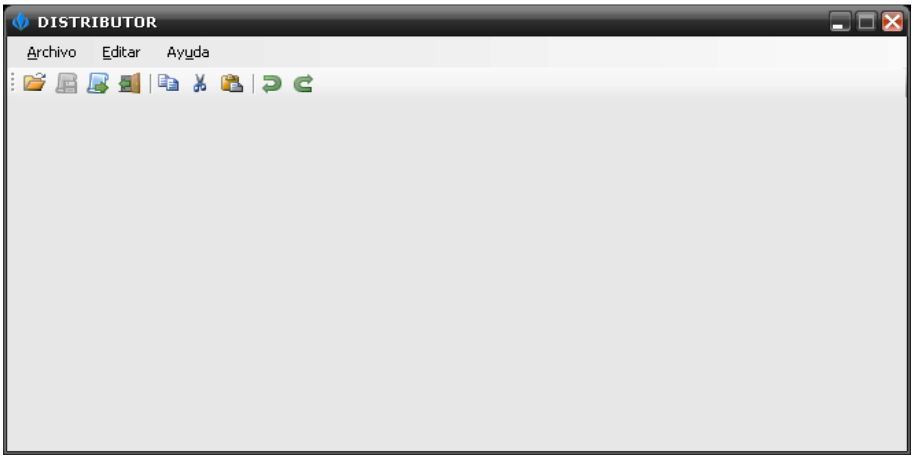



Figura 7. Ventana principal de la herramienta DISTRIBUTOR.

Para trabajar con un archivo XML se debe abrir con la opción Abrir archivo XML o mediante las teclas de acceso directo Ctrl+A. Este mostrará un diálogo con el nombre Abrir archivo XML que tiene filtrado para archivos XML (véase la figura 8).

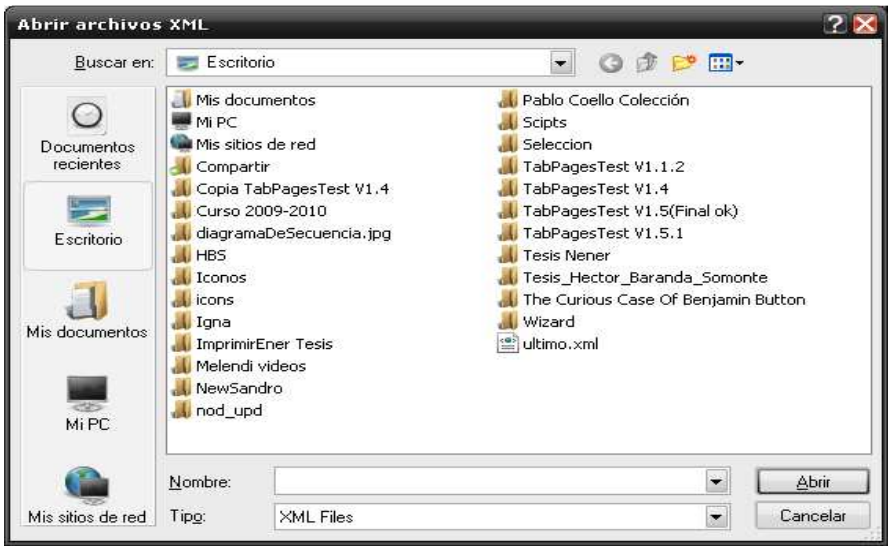



Figura 8. Ventana de diálogo para abrir archivo XML.

Una vez cargado el archivo, se generan los nombres de los sitios leídos del XML como pestañas como se muestra en la figura 9.

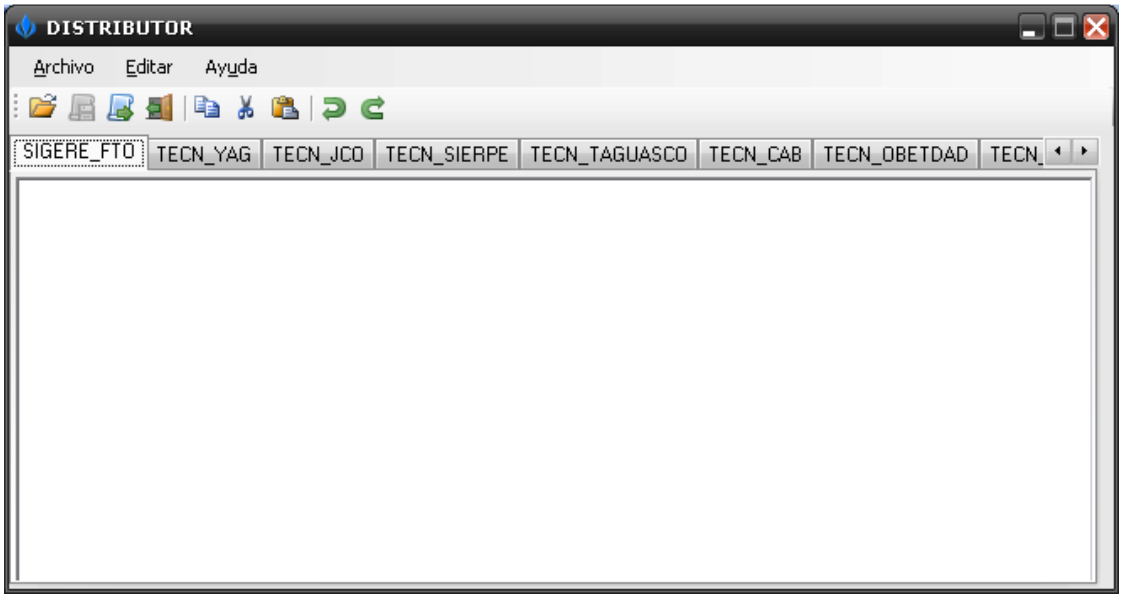



Figura 9. Sitios cargados.

En este momento se puede pasar a la opción Generar Script del menú Archivo o utilizar las teclas de acceso directo Ctrl+S. Se necesita obtener la información para la configuración del servidor de publicación, subscripción, así como la BD de publicación y subscripción (véase la figura 10).

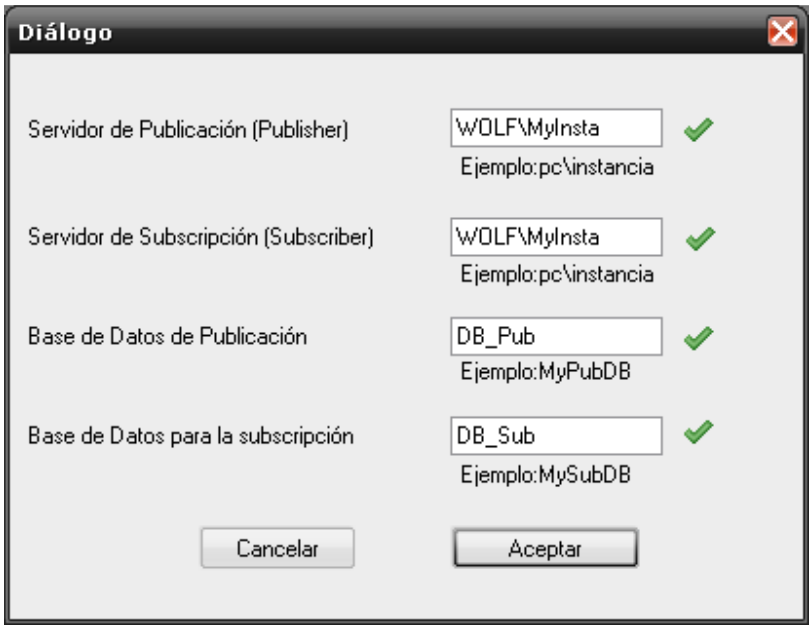



Figura 10. Configuración de parámetros de replicación.

Al presionar el botón Aceptar, el software crea los scripts para cada sitio; dentro de estos se crean las bases de datos, las tablas, integridad referencial, así como las configuraciones para la replicación (véase la figura 11).

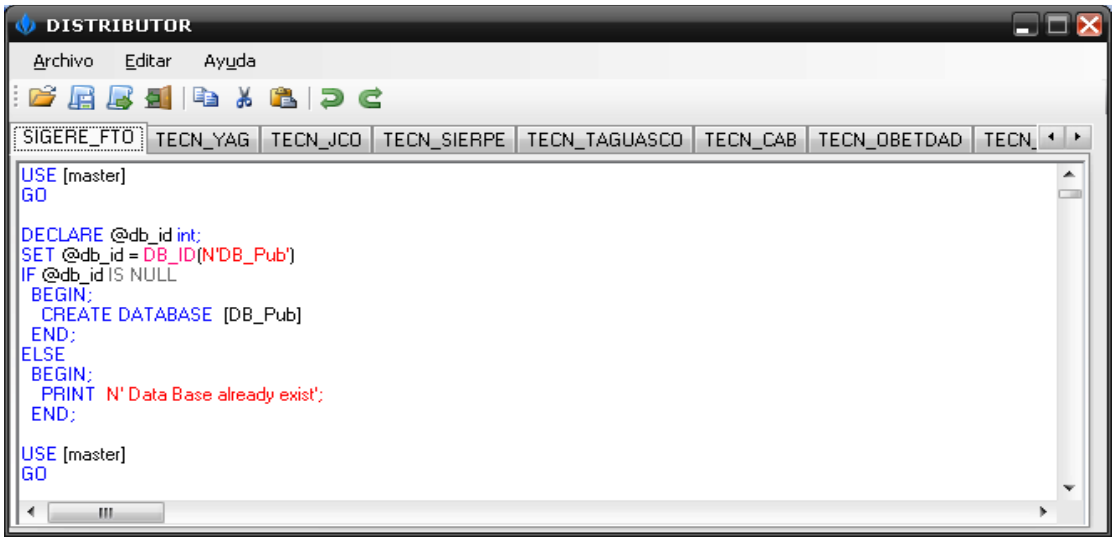



Figura 11. Generación de scripts.

## 3.2 Descripción de un caso de estudio

La Organización Básica Eléctrica (OBE) de Sancti Spíritus se dio a la tarea de implementar un módulo para el control de los transformadores instalados y las operaciones relacionadas en la explotación de los mismos, reflejando la estructura inherentemente distribuida de las diferentes unidades de la Empresa Eléctrica, teniendo autonomía local y ofreciendo buen rendimiento.

En Cuba, durante los últimos años, la distribución ha sido el área de trabajo menos atendida de la Unión Eléctrica, presentándose altos índices de pérdidas e interrupciones, un creciente número de transformadores dañados, y un escaso nivel de informatización y automatización.

Como respuesta a los problemas informáticos detectados, se identifica como necesidad el desarrollo de un sistema que permita controlar los transformadores instalados con una reducción de los gastos totales de explotación de la distribución, influyendo en un mejor servicio a los clientes. El transformador es el equipo más cercano al cliente que más abunda en las redes eléctricas cubanas, con una distribución espacial muy variada y el mayor índice de fallas; de ahí la importancia de minimizar sus averías. Esta es otra razón por la que se necesita desarrollar un sistema capaz de automatizar toda la información relacionada con ellos, poder seguir sus ciclos de mantenimiento, su estado de carga; así como ser capaz de prevenir las fallas.

Para diseñar el sistema hay que tener en cuenta la estructura geográficamente distribuida de las Empresas Eléctricas y sus propias características de comunicaciones; reconociendo los cuatro niveles bien definidos: nacional, provincial, territorial y de sucursal, además de un quinto nivel que es el del taller, al mismo nivel de la provincia.

La comunicación entre la OBE Provincial y las OBEs territoriales se hace a través de líneas telefónicas de baja velocidad; aunque en los últimos tiempos se han introducido gradualmente las redes inalámbricas, aún no se cuenta con este sistema en todas las provincias, ni es política de la Unión Eléctrica hacerlo extensivo en poco tiempo. Como la topología de las redes informáticas no es la más adecuada para contar con una BDC

provincial donde accedan todos los usuarios, ya sean de la provincia, taller o territorio, es necesario diseñar una BDD, para poder seguir el ciclo de vida completo de cada transformador manteniendo datos históricos de todas las operaciones que se realizan.

Esto le da la facilidad de hacer numerosos estudios comparativos del comportamiento de las diferentes instalaciones, equipos, así como prever posibles fallas en ellos. El transformador tiene un ciclo de vida muy difícil de seguir ya que pasa por diferentes estados en diferentes ubicaciones.

El control de los transformadores se hace en el ámbito provincial, por lo que sólo intervienen la provincia, el territorio y el taller como sitios del sistema. El ciclo de vida del transformador comienza en el taller mediante la verificación de un transformador nuevo.

Después pasa a formar parte de los transformadores disponibles en la OBE Provincial donde aparecerán todos los datos de su ficha. Allí, dependiendo de sus características y las necesidades de las diferentes OBEs territoriales, es asignado a una de estas, y de ahí hacia el banco que lo necesita. En el municipio sólo se encuentran los datos pertenecientes a las instalaciones y equipos que atiende este territorio, mientras que en la provincia está centralizada toda la información de las OBEs territoriales.

En la OBE territorial se realizan todas las operaciones sobre el transformador que van conformando una historia. Existen dos formas para que un transformador retorne hacia la provincia: moviéndolo directo al taller o mediante un evento de Reporte de inspección al transformador dañado que lo retira automáticamente del Banco y lo envía al taller si se comprueba que está dañado, a la vez que se crea un evento de Necesidad de transformador para sustituir este averiado. Luego de llegar el transformador y sus datos al taller, se realiza la Defectación o Diagnóstico en el Taller. Esta operación es la que dice si el transformador se retira definitivamente por su daño o si su problema es solucionable. En este caso, el transformador no pierde su historia, se da como disponible, pasa a la provincia y vuelve a fluir, dependiendo de sus características técnicas y las necesidades de las OBEs territoriales. Nótese que en todo este ir y venir

no pierde los datos de todas las pruebas que se le hayan realizado, aunque sea situado en una OBE territorial diferente.

Para este caso de estudio se tienen los siguientes sitios, donde para cada uno de ellos se va a generar un script mediante el DISTRIBUTOR:

- UEB Subcentro Fomento (Sigere_Fto)
- UEB OBE Sancti Spíritus (Tecn_ssp)
- UEB Subcentro La Sierpe (Tecn_Sierpe)
- UEB Subcentro Taguasco (Tecn_Taguasco)
- UEB OBE Trinidad (Tecn_Obetdad)
- UEB OBE Yaguajay (Tecn_Yag)
- UEB OBE Cabaiguán (Tecn_Cab)
- UEB OBE Jatibonico (Tecn_Jco)
- Taller (Transformadores)

A continuación se muestra un fragmento del script generado por el software para uno de los sitios (Transformadores) del archivo XML utilizado de prueba:

```
…..
use[DB_Pub]
CREATE TABLE EstructuraAdministra
(
        CALCULADO          int NOT NULL,
        CENTRO_DE_COSTO      int NOT NULL,
        CODIGO       int NOT NULL,
        CODIRECCION       int NOT NULL,
        E_MAIL         varchar(50) NOT NULL,
        ID_EADIRECCION  int NOT NULL,
        ID_EADMINISTRATIVA      int NOT NULL,
        JEFE  int NOT NULL,
        KM_LINEA    varchar(30) NOT NULL,
        KVAINSTALADOS  int NOT NULL,
```

```
	N_CONSUMIDORES		int NOT NULL,
	N_FAX		varchar(30) NOT NULL,
	N_TELEFONO		varchar(25) NOT NULL,
	NOMBRE		varchar(30) NOT NULL,
	SUBORDINADA		varchar(40) NOT NULL,
	TIPO	varchar(30) NOT NULL,
CONSTRAINT		PK_EstructuraAdministra	PRIMARY KEY CLUSTERED
	(
	ID_EADMINISTRATIVA		ASC,
	JEFE	ASC,
	KVAINSTALADOS	ASC,
	NOMBRE		ASC
	)
)
-- Enabling the replication database
use master
exec sp_replicationdboption @dbname = N'DB_Pub', @optname = N'merge publish',
@value = N'true'
GO
-- Adding the merge publication
use [DB_Pub]
exec sp_addmergepublication @publication = N'PublicationHorizontal', @description =
N'Merge publication of database ''DB_Pub'' from Publisher ''WOLF\MyInsta''.',
@sync_mode = N'native', @retention = 14, @allow_push = N'true', @allow_pull =
N'true', @allow_anonymous = N'true', @enabled_for_internet = N'false',
@snapshot_in_defaultfolder = N'true', @compress_snapshot = N'false', @ftp_port = 21,
@ftp_subdirectory = N'ftp', @ftp_login = N'anonymous', @allow_subscription_copy =
N'false', @add_to_active_directory = N'false', @dynamic_filters = N'false',
@conflict_retention = 14, @keep_partition_changes = N'false', @allow_synctoalternate
= N'false', @max_concurrent_merge = 0, @max_concurrent_dynamic_snapshots = 0,
```

@use_partition_groups = null, @publication_compatibility_level = N'100RTM', @replicate_ddl = 1, @allow_subscriber_initiated_snapshot = N'false', @allow_web_synchronization = N'false', @allow_partition_realignment = N'true', @retention_period_unit = N'days', @conflict_logging = N'both', @automatic_reinitialization_policy = 0

……

## 3.3 Solución

Para utilizar en SQL Server los resultados obtenidos por DISTRIBUTOR se debe abrir el administrador SQL Server Management Studio. Posteriormente se copia el script hacia una nueva consulta y se ejecuta. Una vez realizada esta acción se crean las bases de datos así como las tablas referentes a ellas; se configura el servidor de Distribución, Publicación y Subscripción, además de las publicaciones y subscripciones que contiene el script para ese sitio, quedando como se muestra en la figura 12.

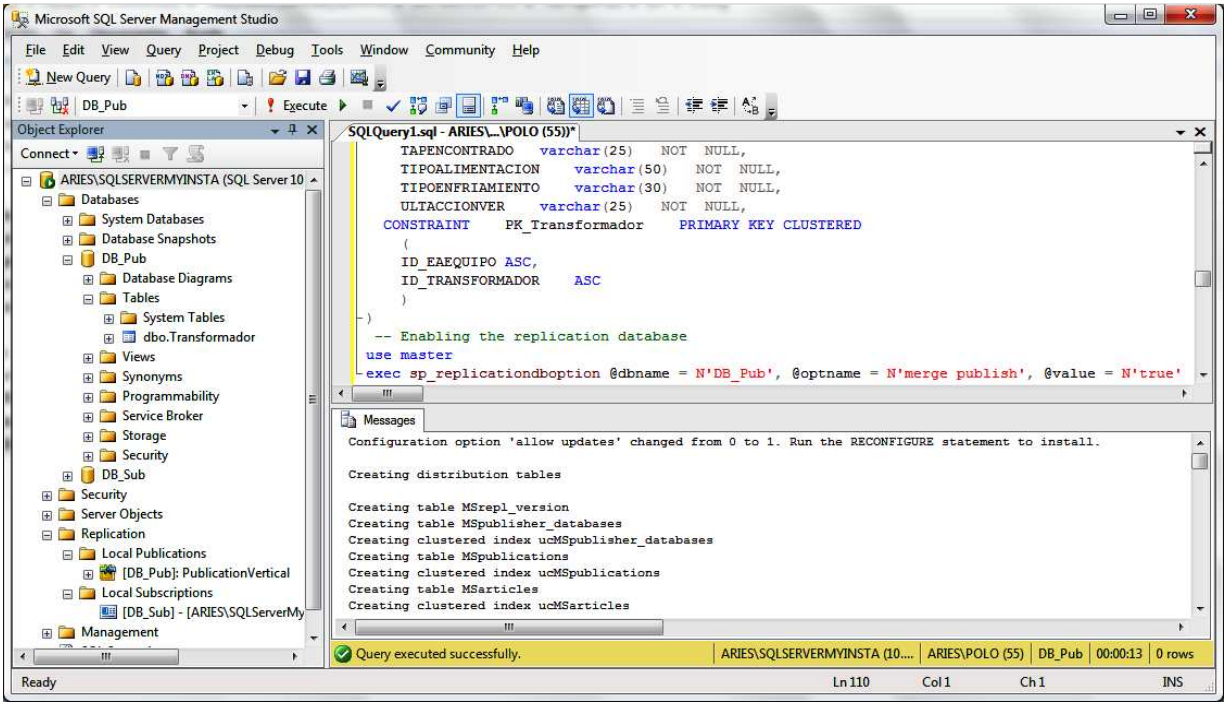



Figura 12. Generación de scripts.

Luego de ser ejecutado el script se crea la instantánea inicial por el agente de instantáneas (SnapshotAgent). Para ello se debe hacer clic derecho sobre el nombre de la publicación ([DB_Pub]: PublicationVertical) y elegir la opción View SnapshotAgent Status para ver el estado.


Figura 13. Agente de instantáneas.

Seguidamente se puede comprobar que fue realizada la sincronización haciendo clic derecho sobre la subscripción ([DB_Sub]-[WOLF\MyInsta].[DB_Pub]: PublicationVertical) y elegir la opción View Synchronization Status, el cual brinda la información que se observa en la figura 14.

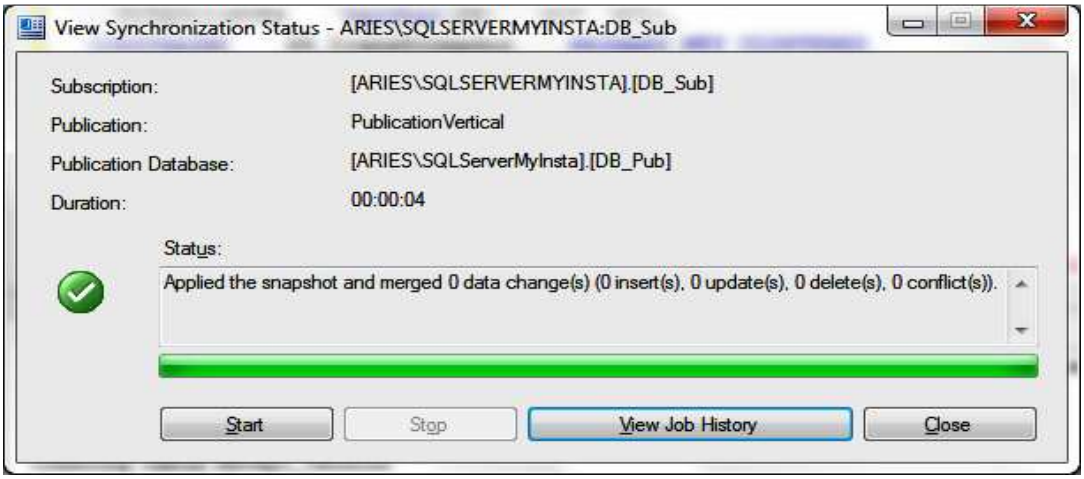


Figura 14. Estado de sincronización.

Una vez comprobadas las acciones anteriores se puede probar que se haya efectuado la replicación, para lo que se debe expandir la base de datos de subscripción en el panel, con la finalidad de chequear si se escribieron los cambios en la misma; en este caso se ejecutó perfectamente el modelo de réplica, de las 21 columnas que posee la tabla Transformadores de la base de datos DB_Pub se filtraron 12 y se escribieron en la base de datos de subscripción DB_Sub (véase la figura 15).

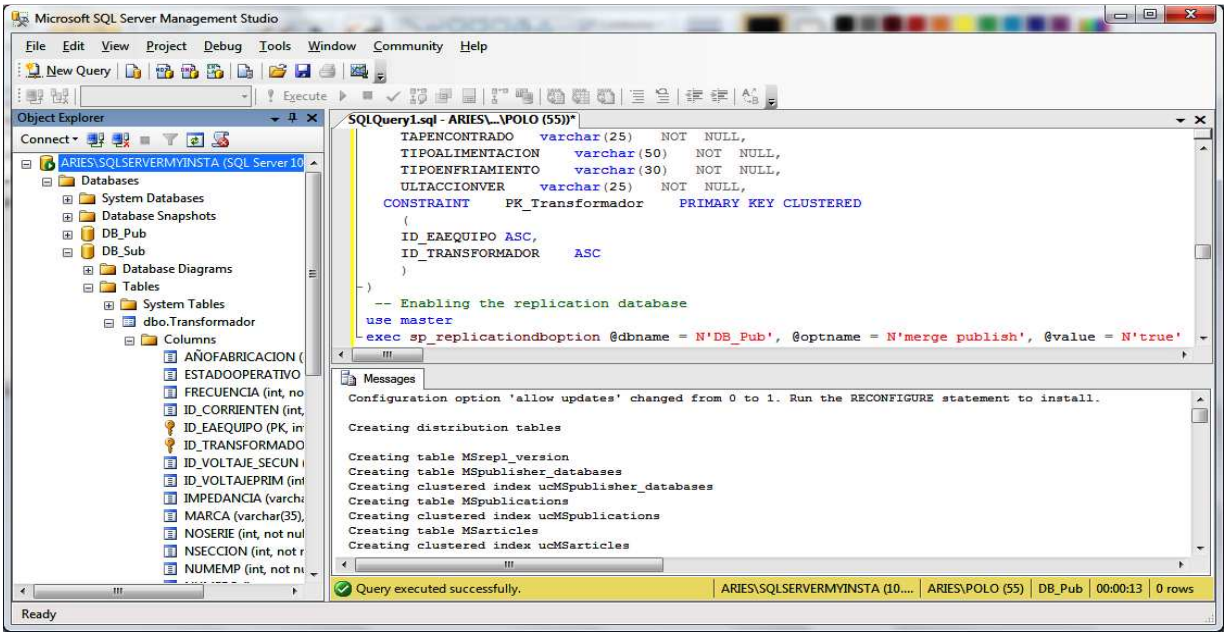



Figura 15. Estado de replicación.

De esta manera se ven los resultados del filtrado aplicado a la tabla de publicación, o sea, la base de datos de subscripción solo contendrá tablas con algunas de las columnas, debido que para este caso se está en presencia de un filtrado vertical.

## 3.4 Consideraciones Parciales

En este capítulo se ha asistido a la prueba del software DISTRIBUTOR, tomando como punto de partida el archivo XML generado para un caso de estudio sobre

transformadores de distribución de energía eléctrica. A partir de este archivo se obtuvo un conjunto de scripts para la generación de esquemas físicos en cada sitio. Estos fueron ejecutados en Microsoft SQL Server lográndose la replicación de datos.

# Conclusiones

1. Se determinaron elementos de configuración de los servidores, tipo de replicación, publicación y suscripción apropiados para el diseño de BDD.
2. Se identificaron los aspectos de programación en T-SQL mediante llamadas a procedimientos almacenados para configurar un entorno de replicación completo que permitió materializar diseños de BDD a nivel físico en Microsoft SQL Server.
3. Se diseñó una herramienta que permita generar los esquemas físicos de ubicación usando los elementos programáticos de replicación identificados que usa como entrada un documento XML con toda la información requerida para esto.
4. Se implementó una herramienta denominada DISTRIBUTOR que sigue el diseño realizado, es fácil de usar y representa una solución al problema de ubicación que es independiente de las estructuras internas manejadas por SIADBDD.
5. Se probó la herramienta con un caso de estudio que hizo posible su validación.

# Recomendaciones

Implementar la sincronización de los cambios. Mediante la utilización de procedimientos almacenados predefinidos en MS SQL Server.

# Referencias Bibliográficas

BAIÃO, F. A., MATTOSO, M., SHAVLIK, J. W. & ZAVERUCHA, G. Year. Applying Theory Revision to the Design of Distributed Databases. *In:* HORVÁTH, T. & YAMAMOTO, A., eds. Proceedings of the 13th International Conference on Inductive Logic Programming ILP 2003, LNAI 2835, September 29-October 1 2003 Szeged, Hungary. Springer, 57-74.

BAIÃO, F. A., MATTOSO, M. & ZAVERUCHA, G. Year. A Framework for the Design of Distributed Databases. *In:* LITWIN, W. & LÉVY, G., eds. Records of the 4th International Meeting on Distributed Data & Structures 4 (WDAS 2002), 2002 Paris, France. Carleton Scientific, 29-36.

BAIÃO, F. A., MATTOSO, M. & ZAVERUCHA, G. 2004. A Distribution Design Methodology for Object DBMS. *Distributed and Parallel Databases,* 16**,** 45-90.

BELLATRECHE, L., KARLAPALEM, K. & SIMONET, A. 2000. Algorithms and support for horizontal class partitioning in object-oriented databases. *Distributed and Parallel Databases,* 8**,** 155-179.

BERTONE, R. 2004. Métricas de Performance en Administración de BD Distribuidas en redes LAN y WAN.

BHALLA, S. & HASEGAWA, M. Year. Parallelizing serializable transactions within distributed real-time database systems *In:* Proceedings of the International

Conference on Embedded and Ubiquitous Computing (EUC 2005). Lecture Notes in Computer Science, 2005. Springer, 203-213

CERI, S., MARTELLA, G. & PELAGATTI, G. 1982. Optimal File Allocation in a Computer Network: a Solution Method Based on the Knapsack Problem. *Computer Networks,* 6**,** 345-357.

CERI, S., NAVATHE, S. B. & WIEDERHOLD, G. 1983. Distribution Design of Logical Database Schemas. *IEEE Trans. Software Eng.,* 9**,** 487-504.

CERI, S. & PERNICI, B. 1985. *DATAID-D: Methodology for Distributed Database Design,* North-Holland.

CERI, S., PERNICI, B. & WIEDERHOLD, G. 1987. Distributed Database Design Methodologies. *IEEE Database Eng. Bull.,* 7**5,** 533-546.

COULON, C., PACITTI, E. & VALDURIEZ, P. Year. Consistency Management for Partial Replication in a High Performance Database Cluster. *In:* IEEE International Conference on Parallel and Distributed Systems (ICPADS2005), 2005 Fukuoka, Japan.

DATE, C. J. 2000. *Introducción a los Sistemas de Bases de Datos, Séptima edición,* México, Addison-Wesley.

GANÇARSKI, S., NAACKE, H., PACITTI, E. & VALDURIEZ, P. Year. Parallel Processing with Autonomous Databases in a Cluster System. *In:* MEERSMAN, R. & TARI, Z., eds. Proceedings of the Confederated International Conferences On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE, October 30- November 1, 2002 2002 Irvine, California, USA. Springer, 410-428.

HABABEH, I. O., BOWRING, N. & RAMACHANDRAN, M. Year. A Method for Fragment Allocation in Distributed Object Oriented Database Systems. *In:* Proceedings of the 5th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking & Broadcasting (PGNet), June, 2004 2004 Liverpool, UK. 54 - 59.

JOHANSSON, J. M., MARCH, S. T. & NAUMANN, J. D. Year. The effects of parallel processing on update response time in distributed database design. *In:* ANG, S., KRCMAR, H., ORLIKOWSKI, W. J., WEILL, P. & DEGROSS, J. I., eds. Proceedings of the Twenty-First International Conference on Information Systems ICIS, December 10-13, 2000 2000 Brisbane, Australia. ACM, 187-196.

LEE, J. W. & BAIK, D. K. 2004. Database allocation modeling for optimal design of distributed systems. *IEICE Transactions on Information and Systems,* E87D**,** 1795-1804.

LIN, Y., KEMME, B., PATIÑO-MARTÍNEZ, M. & JIMÉNEZ-PERIS, R. Year. Middleware based Data Replication providing Snapshot Isolation. *In:* Proceedings of the ACM SIGMOD International Conference on Management of Data., June 14-16, 2005 2005 Baltimore, Maryland, USA. ACM Press, 419-430.

MA, H., SCHEWE, K.-D. & WANG, Q. 2005. Distribution design for higher-order data models. Palmerston North, New Zealand: Massey University, Department of Information Systems.

MEI, H. & SHENG, O. Year. A Semantic Based Methodology for Integrated Computer-Aided Distributed Database Design. *In:* The 25th Hawaii International Conference on System Sciences, 1992. 288- 299.

MORELL, D. E. C. 2004. Replicación de Datos en SQL Server.

NAVATHE, S. B., KARLAPALEM, K. & RA, M. 1995. A mixed fragmentation methodology for initial distributed database design. Atlanta, GA: College of Computing, Georgia Institute of Technology.

ŐZSU, M. T. & VALDURIEZ, P. 1991. *Principles of Distributed Database Systems*, Prentice-Hall.

ŐZSU, M. T. & VALDURIEZ, P. 1999. *Principles of Distributed Database Systems, Second Edition,* Upper Saddle River, New Jersey., Prentice-Hall.

PÉREZ, J., PAZOS, R. A., FRAUSTO-SOLIS, J., REYES, G., SANTAOLAYA, R., FRAIRE, H. J. & CRUZ, L. Year. An approach for solving very large scale instances of the design distribution problem for distributed database systems. *In:* Proceedings of the 4th International School and Symposium on Advanced Distributed Systems (ISSADS2005). Lecture Notes in Computer Science, 2005. Springer, 33-42.

SAVONNET, M., TERRASSE, M.-N. & YÉTONGNON, K. Year. FRAGTIQUE: An OO Distribution Design Methodology. *In:* CHEN, A. L. P. & LOCHOVSKY, F. H., eds. Proceedings of the Sixth International Conference on Database Systems for Advanced

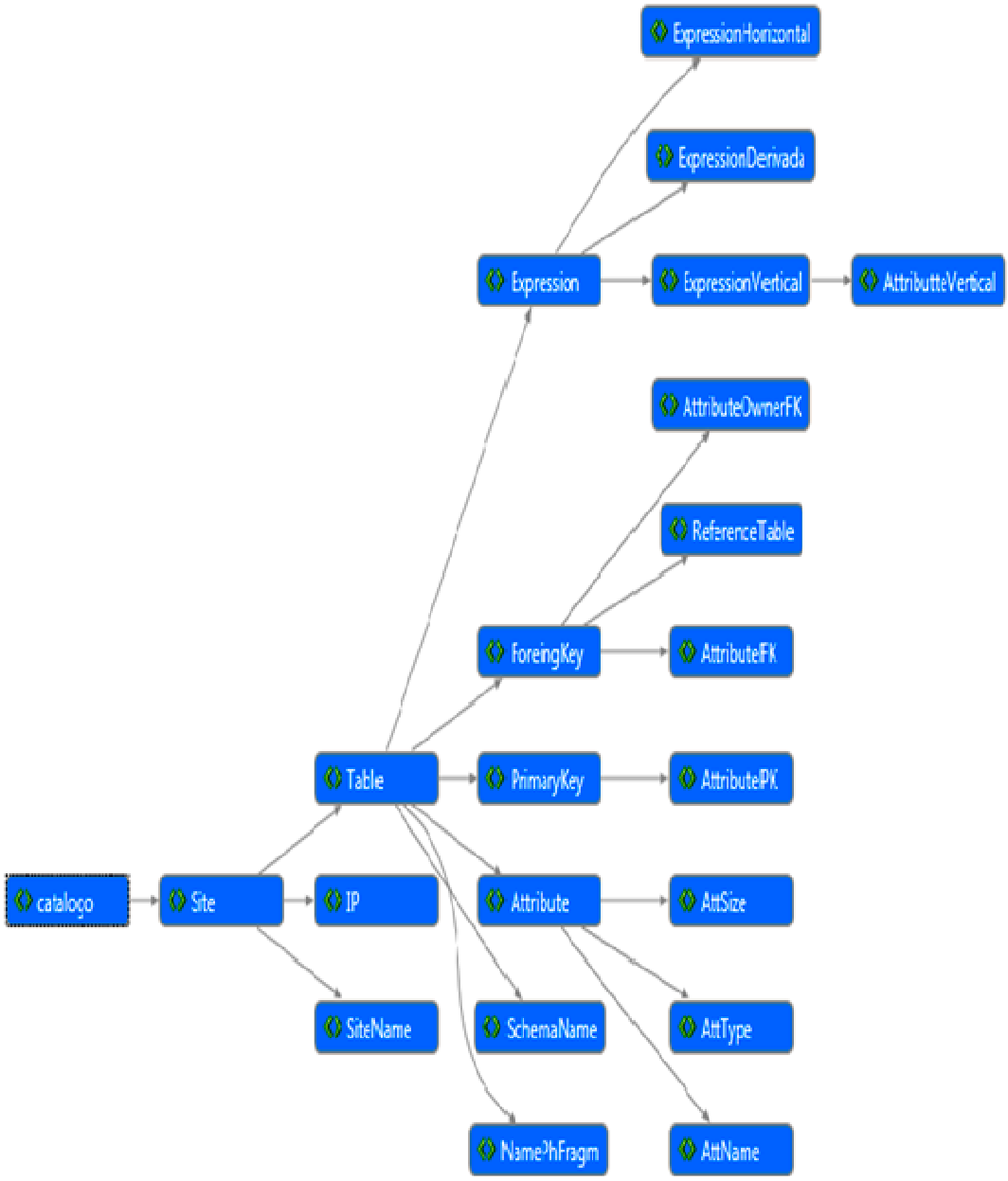
Applications (DASFAA), April 19-21 1999 Hsinchu, Taiwan. IEEE Computer Society, 283-290.

SCHEWE, K.-D. Year. Fragmentation of object oriented and semi-structured data. *In:* HAAV, H.-M. & KALJA, A., eds. Databases and Information Systems II, 2002. Kluwer Academic Publishers, 1-14.

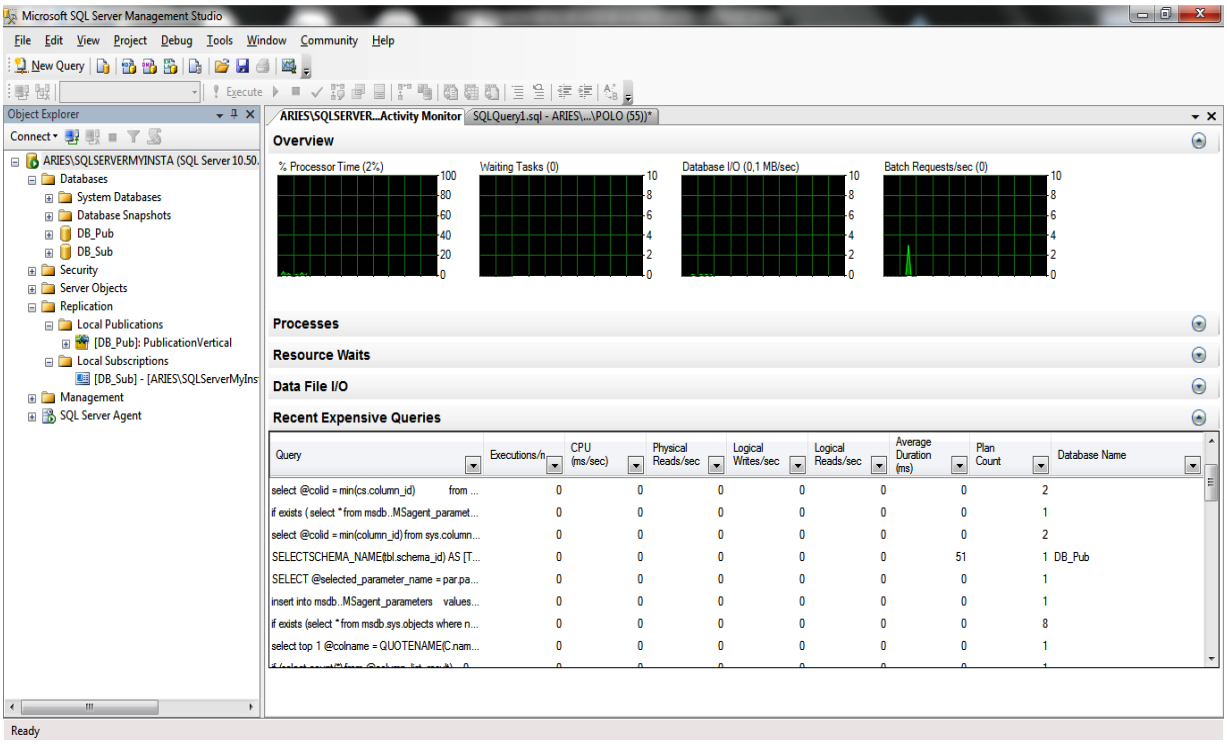
TAMHANKAR, A. M. & RAM, S. 1998. Database fragmentation and allocation: an integrated methodology and case study. *IEEE Transactions on Systems, Man, and Cybernetic Part A,* 28**,** 288-305.

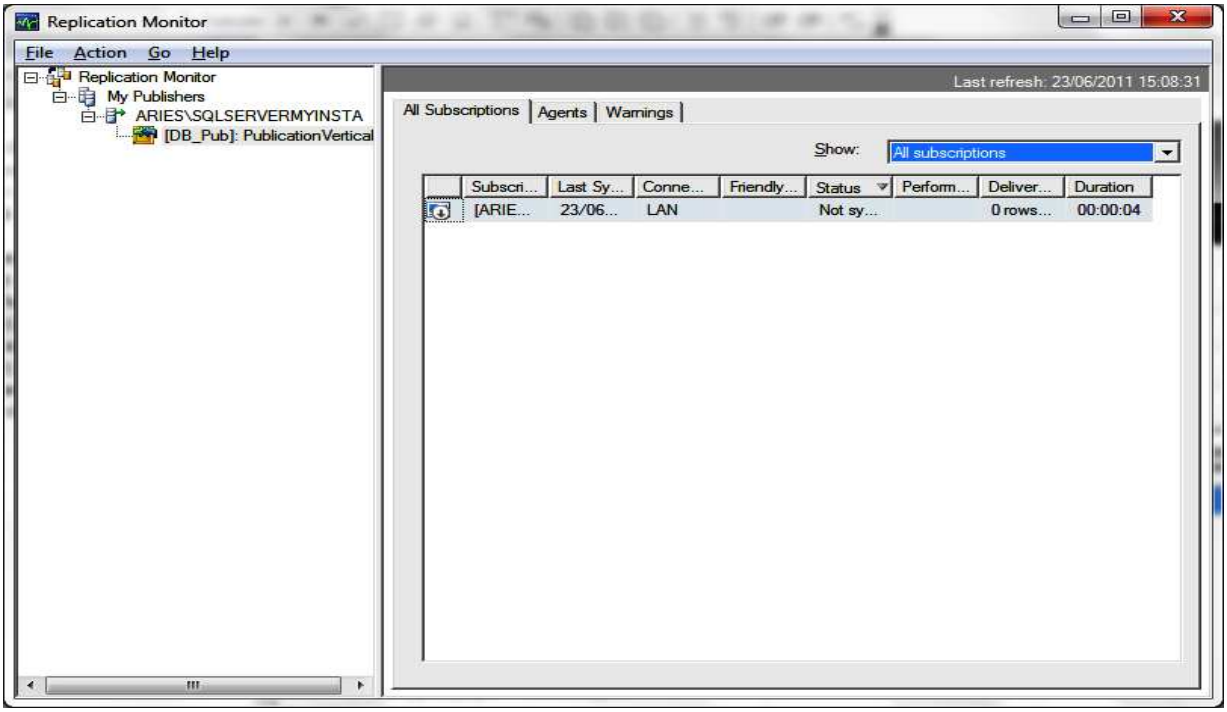
# Anexos

## **Anexo 1.** Esquema XML

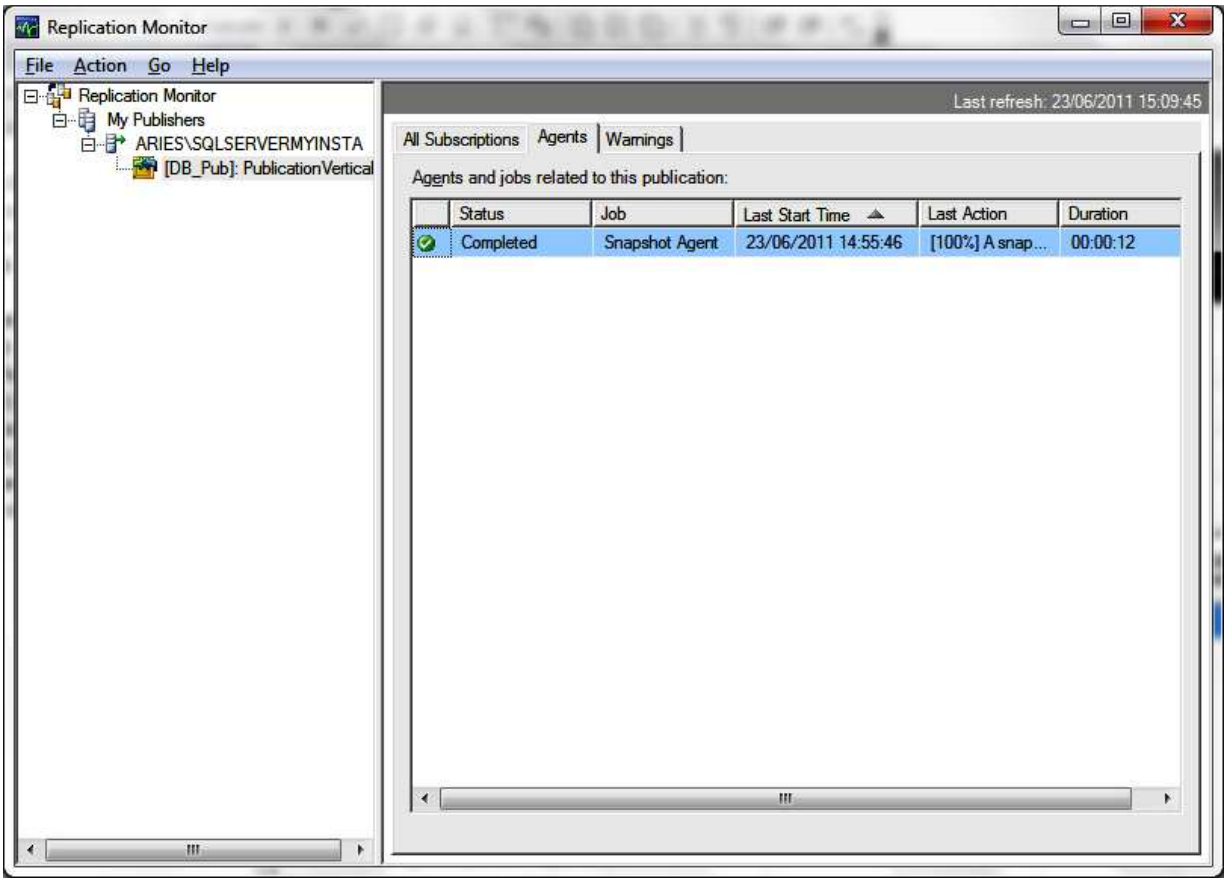


# **Anexo 2.** Monitor de replicación



## **Anexo 2.** Monitor de replicación (Continuación)




## **Anexo 3.** Script generado por DISTRIBUTOR

```
USE [master]
GO

DECLARE @db_id int;
SET @db_id = DB_ID(N'DB_Pub')
IF @db_id IS NULL
BEGIN;
CREATE DATABASE[DB_Pub]
END;
ELSE
BEGIN;
```

```
PRINT N' Data Base already exists';
END;

USE [master]
GO

DECLARE @db_id int;
SET @db_id = DB_ID(N'DB_Sub')
IF @db_id IS NULL
BEGIN;
CREATE DATABASE[DB_Sub]
END;
ELSE
BEGIN;
PRINT N' Data Base already exists';
END;

use master
exec sp_adddistributor @distributor = N'WOLF\MyInsta'
GO
exec sp_adddistributiondb @database = N'distribution', @data_folder = N'C:\Archivos
de programa\Microsoft SQL Server\MSSQL10_50.MyInsta\MSSQL\Data', @log_folder =
N'C:\Archivos de programa\Microsoft SQL Server\MSSQL10_50.MyInsta\MSSQL\Data',
@log_file_size = 2, @min_distretention = 0, @max_distretention = 72,
@history_retention = 48, @security_mode = 1

GO
use [distribution]
if (not exists (select * from sysobjects where name = 'UIProperties' and type = 'U '))
        create table UIProperties(id int)
```

```
if (exists (select * from ::fn_listextendedproperty('SnapshotFolder', 'user', 'dbo', 'table',
'UIProperties', null, null)))
        EXEC sp_updateextendedpropertyN'SnapshotFolder', N'\\WOLF\repldata', 'user',
dbo, 'table', 'UIProperties'
else
        EXEC sp_addextendedpropertyN'SnapshotFolder', N'\\WOLF\repldata', 'user',
dbo, 'table', 'UIProperties'
GO
exec sp_adddistpublisher @publisher = N'WOLF\MyInsta', @distribution_db =
N'distribution', @security_mode = 1, @working_directory = N'\\WOLF\repldata',
@trusted = N'false', @thirdparty_flag = 0, @publisher_type = N'MSSQLSERVER'
GO
use[DB_Pub]
CREATE TABLE EstructuraAdministra
(
        CALCULADO       int NOT NULL,
        CENTRO_DE_COSTO     int NOT NULL,
        CODIGO      int NOT NULL,
        CODIRECCION     int NOT NULL,
        E_MAIL      varchar(50) NOT NULL,
        ID_EADIRECCION  int NOT NULL,
        ID_EADMINISTRATIVA      int NOT NULL,
        JEFE  int NOT NULL,
        KM_LINEA    varchar(30) NOT NULL,
        KVAINSTALADOS  int NOT NULL,
        N_CONSUMIDORES      int NOT NULL,
        N_FAX       varchar(30) NOT NULL,
        N_TELEFONO      varchar(25) NOT NULL,
        NOMBRE      varchar(30) NOT NULL,
        SUBORDINADA     varchar(40) NOT NULL,
```

```
	TIPO  varchar(30) NOT NULL,
CONSTRAINT	PK_EstructuraAdministra  PRIMARY KEY CLUSTERED
	(
	ID_EADMINISTRATIVA	ASC,
	JEFE  ASC,
	KVAINSTALADOS  ASC,
	NOMBRE	ASC
	)
)
-- Enabling the replication database
use master
exec sp_replicationdboption @dbname = N'DB_Pub', @optname = N'merge publish',
@value = N'true'
GO
-- Adding the merge publication
use [DB_Pub]
exec sp_addmergepublication @publication = N'PublicationHorizontal', @description =
N'Merge publication of database ''DB_Pub'' from Publisher ''WOLF\MyInsta''.',
@sync_mode = N'native', @retention = 14, @allow_push = N'true', @allow_pull =
N'true', @allow_anonymous = N'true', @enabled_for_internet = N'false',
@snapshot_in_defaultfolder = N'true', @compress_snapshot = N'false', @ftp_port = 21,
@ftp_subdirectory = N'ftp', @ftp_login = N'anonymous', @allow_subscription_copy =
N'false', @add_to_active_directory = N'false', @dynamic_filters = N'false',
@conflict_retention = 14, @keep_partition_changes = N'false', @allow_synctoalternate
= N'false', @max_concurrent_merge = 0, @max_concurrent_dynamic_snapshots = 0,
@use_partition_groups = null, @publication_compatibility_level = N'100RTM',
@replicate_ddl = 1, @allow_subscriber_initiated_snapshot = N'false',
@allow_web_synchronization = N'false', @allow_partition_realignment = N'true',
@retention_period_unit = N'days', @conflict_logging = N'both',
@automatic_reinitialization_policy = 0
```

```
GO
exec sp_addpublication_snapshot @publication = N'PublicationHorizontal',
@frequency_type = 4, @frequency_interval = 14, @frequency_relative_interval = 1,
@frequency_recurrence_factor = 0, @frequency_subday = 1,
@frequency_subday_interval = 5, @active_start_time_of_day = 4100,
@active_end_time_of_day = 235959, @active_start_date = 0, @active_end_date = 0,
@job_login = null, @job_password = null, @publisher_security_mode = 1
use [DB_Pub]
exec sp_addmergearticle @publication = N'PublicationHorizontal', @article =
N'EstructuraAdministra', @source_owner = N'dbo', @source_object =
N'EstructuraAdministra', @type = N'table', @description = null, @creation_script = null,
@pre_creation_cmd = N'drop', @schema_option = 0x000000010C034FD1,
@identityrangemanagementoption = N'manual', @destination_owner = N'dbo',
@force_reinit_subscription = 1, @column_tracking = N'false', @subset_filterclause =
N'[Id_EAdministrativa ] = 22', @vertical_partition = N'false', @verify_resolver_signature
= 1, @allow_interactive_resolver = N'false', @fast_multicol_updateproc = N'true',
@check_permissions = 0, @subscriber_upload_options = 0, @delete_tracking = N'true',
@compensate_for_errors = N'false', @stream_blob_columns = N'false',
@partition_options = 0
GO

exec sp_startpublication_snapshot@publication ='PublicationHorizontal', @publisher
=null
-----------------BEGIN: Script to be run at Publisher 'WOLF\MyInsta'-----------------
use [DB_Pub]
exec sp_addmergesubscription @publication = N'PublicationHorizontal', @subscriber =
N'WOLF\MyInsta', @subscriber_db = N'DB_Sub', @subscription_type = N'pull',
@subscriber_type = N'local', @subscription_priority = 0, @sync_type = N'Automatic'
GO
-----------------END: Script to be run at Publisher 'WOLF\MyInsta'-----------------
```

```
----------------BEGIN: Script to be run at Subscriber 'WOLF\MyInsta'----------------

use [DB_Sub]
exec sp_addmergepullsubscription @publisher = N'WOLF\MyInsta', @publication =
N'PublicationHorizontal', @publisher_db = N'DB_Pub', @subscriber_type = N'Local',
@subscription_priority = 0, @description = N'', @sync_type = N'Automatic'

exec sp_addmergepullsubscription_agent @publisher = N'WOLF\MyInsta',
@publisher_db = N'DB_Pub', @publication = N'PublicationHorizontal', @distributor =
N'WOLF\MyInsta', @distributor_security_mode = 1, @distributor_login = N'',
@distributor_password = null, @enabled_for_syncmgr = N'False', @frequency_type =
1, @frequency_interval = 0, @frequency_relative_interval = 0,
@frequency_recurrence_factor = 0, @frequency_subday = 0,
@frequency_subday_interval = 0, @active_start_time_of_day = 0,
@active_end_time_of_day = 0, @active_start_date = 0, @active_end_date =
19950101, @alt_snapshot_folder = N'', @working_directory = N'', @use_ftp = N'False',
@job_login = null, @job_password = null, @publisher_security_mode = 1,
@publisher_login = null, @publisher_password = null, @use_interactive_resolver =
N'False', @dynamic_snapshot_location = null, @use_web_sync = 0
GO
----------------END: Script to be run at Subscriber 'WOLF\MyInsta'----------------

use[DB_Pub]
exec sp_resyncmergesubscription @publisher =WOLF\MyInsta, @publisher_db =
'DB_Pub', @publication ='PublicationHorizontal', @resync_type = 0
```

Como se ha podido observar, DISTRIBUTOR es capaz de generar scripts en T-SQL para la generación de esquemas físicos de BDD mediante replicación. A continuación

se explica la utilización de resultados intermedios para su utilización en Microsoft SQL Server.