Universidad de las Ciencias Informáticas

Facultad 6

Título: "Propuesta de solución de réplica multi-maestra asincrónica para bases de datos"

Trabajo de diploma

Presentado para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Roxana Silva Cobas.

Tutor: Ing. Leonel Fuentes Marrero.

Cotutor: Ing. Alberto Garnache Mendoza.

Ciudad de La Habana, Cuba.

"Año 52 de la Revolución"

“Solo se fracasa cuando se deja de intentar”

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de esta tesis y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Tesista.

Roxana Silva Cobas

_____ _____

Firma del Tutor. Firma del Cotutor.

Leonel Fuentes Marrero Alberto Mendoza Garnache

DEDICATORIA

♥♥ A toda mi familia ♥♥

AGRADECIMIENTOS

♥♦♥ Gracias por brindarme su apoyo y amor:

A Rasiel.

A mis padres.

A todas mis tías y tíos.

A mis abuelas y abuelos.

A mis primas y primos.

A mis amigas y amigos.

A mis tutores.

A todos los que me ayudaron.

A todos los que se preocupan por mí.

A la Revolución.

♥♦♥ Los quiero mucho ♥♦♥

RESUMEN

Las bases de datos, se encuentran ampliamente difundidas en la sociedad actual de la informática y el desarrollo de software, ya que permiten el almacenamiento y acceso a la información de manera eficiente, práctica y confiable. Cuando existen varios servidores de bases de datos, ubicados en distintos lugares geográficos, y se necesita mantener la misma información, actualizada y disponible en cada uno de ellos, se utiliza la replicación. Con las técnicas de replicación se transportan datos entre dos o más servidores.

En el presente trabajo se propone una solución de réplica multi-maestra asincrónica, que permitirá realizar de manera efectiva la replicación de datos a través de servidores ubicados en varios niveles y para entornos donde la red es inestable y pueden pasar largos períodos sin conexión. Se podrá llevar a cabo la replicación cuando existan las condiciones adecuadas. Se asegura una alta disponibilidad de los datos, para que los procesos sean atendidos en una razón de tiempo razonable. Se admite la tolerancia a fallos, sin alterar el correcto comportamiento del sistema. Se garantiza la coordinación entre cada una de las partes que intervienen en el proceso de replicación. Se mantiene la confidencialidad de la información, sólo los nodos autorizados podrán tener acceso, una vez que se ha hecho el registro, protegiéndose así contra entradas no autorizadas.

Palabras clave: bases de datos, herramienta, replicación de datos, Symmetric.

ÍNDICE

INTRODUCCIÓN

El uso extensivo y cada vez más integrado de las tecnologías de la información y las comunicaciones (TIC) es una característica y factor de cambio de nuestra sociedad actual. Provocan continuas transformaciones e inciden en casi todos los aspectos de nuestra vida. En los últimos años, estas tecnologías han experimentado un acelerado avance, por lo que hoy día representan una herramienta clave en los proyectos, empresas y organizaciones.

Cuba cuenta con varios centros de desarrollo y profesionales capacitados en la rama de la informática. El Ministerio de la Informática y las Comunicaciones (MIC), que tiene como misión impulsar el proceso de informatización de la sociedad cubana (1), ha creado estrategias para mejorar la industria de software. Como parte de esas estrategias, se encuentra la Universidad de las Ciencias Informáticas (UCI), que es la primera universidad creada sobre el concepto de universidad productiva. La UCI incluye entre sus objetivos, aparte de formar ingenieros informáticos, la creación de productos, servicios y soluciones informáticas, para ser brindados al país y al mundo.

Dentro de la UCI se encuentra el Centro de Tecnologías de Gestión Datos (DATEC), que es un centro especializado en el desarrollo de tecnologías de bases de datos. Entre los servicios que oferta está la provisión de soluciones y consultorías relacionadas con tecnologías de bases de datos y la implantación de soluciones de réplica de datos.

Las bases de datos, se encuentran muy difundidas en el mundo actual de la informática y el desarrollo de software, ya que permiten almacenar y acceder a la información eficientemente. Entre los aspectos más avanzados que se manejan en ese sentido, se encuentran las bases de datos distribuidas, cuya información está almacenada de manera física en diferentes sitios de una red, pero de manera lógica pertenece a un mismo sistema distribuido, así el usuario lo percibe como un único sistema. (2)

Para mejorar el rendimiento de las consultas realizadas por el usuario al sistema, se utilizan las técnicas de replicación de datos entre servidores. La replicación de datos es el proceso de compartir objetos y datos de una base de datos a múltiples bases de datos, en diferentes localizaciones. (3) Su principal utilidad es que aumenta la disponibilidad de los datos. Permite crear copias de respaldo ante fallos y aumenta la escalabilidad, que es la capacidad del sistema para mantener su rendimiento conforme aumenta el número de usuarios. (4)

Existen dos tipos de entornos para la replicación, son los siguientes:

- Maestro-Esclavo (Master-Slave), que permite a un sólo servidor maestro realizar consultas de lectura/escritura sobre los demás esclavos que sólo pueden realizar consultas de lectura.
- Multi-maestro (Multi-master), que permite a múltiples servidores maestros realizar consultas de lectura/escritura entre ellos.

Según el tiempo de latencia, la replicación puede ser:

- Sincrónica: cuando se replican los cambios en tiempo real.
- Asincrónica: cuando se replican los cambios en intervalos regulares de tiempo.

Las organizaciones que tienen varios niveles de dirección ubicados en distintos lugares geográficos, necesitan tener la información actualizada y disponible en todos los niveles, para manejar y coordinar operaciones. Continuamente se realizan registros administrativos y otras transacciones, arrojando un gran flujo de información, que precisa ser procesada y revisada, por lo tanto se necesita transmitir y actualizar esa información en cada una de las direcciones y departamentos. El proceso de actualización de datos, muchas veces se realiza mediante la red, con la utilización de correo electrónico, o la copia en servidores ftp. Por tanto, la información que se envía está propensa a pérdidas y a que se encuentre repetida, porque no hay constancia de que haya sido guardada en algún lugar. Toda esa labor se efectúa prácticamente de forma manual en muchos casos, lo que no garantiza ninguna seguridad. La gestión de la información puede ser un proceso complejo, que depende principalmente de la actualización que tengan los datos que se manejan. Si se gestiona información no actualizada se tiene que repetir el proceso. Lo que trae consigo que no se cumpla con los plazos establecidos para las actualizaciones, y provoca demora en la gestión de la información, dificultando la toma eficiente y rápida de decisiones.

Teniendo en cuenta la situación problémica anteriormente descrita se plantea como **problema científico** ¿Cómo lograr la disponibilidad de la información actualizada en cada una de las oficinas o filiales de una empresa, institución u organización dada, en sus distintos niveles de dirección?

El **objeto de estudio** de la investigación son las soluciones de réplica de datos, enmarcando el **campo de acción** a las soluciones de réplica de datos multi-maestra asincrónica.

Se trazó como **objetivo general** desarrollar una propuesta de solución de réplica multi-maestra asincrónica para bases de datos.

A partir del objetivo general se derivan los siguientes **objetivos específicos**:

1. Analizar tendencias referentes a las soluciones de réplica de datos.
2. Realizar el análisis y diseño de la propuesta de solución de réplica multi-maestra asincrónica para bases de datos.
3. Validar la solución.

Para cumplir con los objetivos planteados se proponen las siguientes **tareas de investigación**:

1. Caracterización de las soluciones de réplica de datos.
2. Establecimiento de comparaciones entre soluciones de réplica de datos.
3. Identificación de los requerimientos necesarios para el desarrollo de la propuesta.
4. Realización del análisis.
5. Realización del diseño.
6. Instalación y configuración.
7. Validación de la solución.

El documento está estructurado esencialmente en: Resumen, Introducción, Desarrollo y Conclusiones. El desarrollo está compuesto por 3 capítulos:

En el **capítulo 1** se abordan los conceptos relacionados con bases de datos, bases de datos distribuidas y replicación de datos. Se ofrece una visión general del contenido concerniente al desarrollo de soluciones de réplica de datos, facilitando una mejor comprensión del mismo. Se realiza un estudio de las herramientas de réplica multi-maestra asincrónica existentes y se da a conocer la solución propuesta.

En el **capítulo 2** se realiza el análisis y diseño de la solución. Se identifican los requerimientos y pasos a seguir, puntualizando en la configuración y opciones de instalación. Se implementa una herramienta de apoyo para la configuración inicial en la instalación de la herramienta seleccionada, que facilitará dicho proceso.

En el **capítulo 3** se valida la solución, realizándole pruebas, para verificar su eficiencia y garantizar su funcionamiento, asegurando la integridad, disponibilidad y seguridad de los datos. Las pruebas están enfocadas esencialmente a los requerimientos y son efectuadas en 2 entornos distintos. Los resultados son observados, comparados y evaluados.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se tratan los principales conceptos relacionados con las bases de datos, bases de datos distribuidas y replicación de datos. Se realiza un estudio de los aspectos concernientes al desarrollo de soluciones de réplica de datos, ofreciendo una visión general de las mismas. Se aborda el estado del arte de las herramientas de réplica con el objetivo de tener un acercamiento a ellas, centrando la atención en las que son del tipo multi-maestra y asincrónica. Cada una de las herramientas candidatas será sometida a un proceso de estudio y análisis, para seleccionar la más apropiada a utilizar en la solución. Se incluyen las tendencias actuales, así como las tecnologías más empleadas en el tema. Finalmente, se da a conocer la solución propuesta que incluye la herramienta, el gestor de bases de datos y las tecnologías a utilizar, justificando la elección.

1.1 Bases de datos

Una base de datos es una colección de datos persistentes que son usados por sistemas de alguna empresa. El término empresa convenientemente se refiere a una organización. Una empresa pudiera ser un individuo (con una pequeña base de datos), o una corporación o similar (con una gran base de datos compartida). (5)

Por persistentes se entiende, intuitivamente, que los datos de la base de datos difieren de otros tipos de datos, tales como datos de entrada, datos de salida, consultas SQL, resultados intermedios y otros datos más generales. Pero precisamente se dice que los datos en una base de datos "persisten" porque, una vez que han sido aceptados por el sistema de base de datos para entrar en la base de datos, estos pueden subsecuentemente ser borrados de la misma solamente por una petición explícita del sistema de base de datos. (5)

Un sistema de base de datos es básicamente un sistema informático de almacenamiento de datos; en otras palabras es un sistema informático cuyo propósito general es almacenar información y permitir que los usuarios recuperen y actualicen en demanda esta información. La información en cuestión puede ser cualquier cosa que le interese a un individuo u organización. Los componentes principales de un sistema de base de datos son los datos, el hardware, el software y los usuarios. (5) La Figura 1: Sistema de bases de datos, muestra la relación entre los usuarios, la base de datos, y los datos.

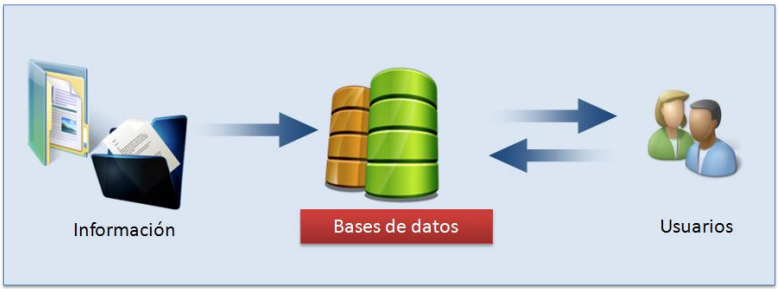

Figura 1: Sistema de bases de datos.

1.1.1 Bases de datos distribuidas

Un modelo de bases de datos distribuidas consiste en un conjunto de localidades, cada una de las cuales mantiene un sistema de bases de datos local. Cada localidad puede procesar transacciones locales, o bien transacciones globales entre varias localidades, requiriendo para eso comunicación entre ellas. (6)

En un modelo de bases de datos distribuidas, los datos se almacenan en varias computadoras. Las computadoras de un sistema distribuido se comunican entre sí a través de diversos medios de comunicación, tales como cables de alta velocidad o líneas telefónicas. No comparten la memoria principal ni el reloj. (6)

La diferencia principal entre los sistemas de bases de datos centralizados y distribuidos es que, en los primeros, los datos residen en una sola localidad, mientras, en los últimos, se encuentran en varias localidades. (6) La Figura 2: Sistema de bases de datos distribuidas muestra la estructura de un sistema de bases de datos distribuidas.

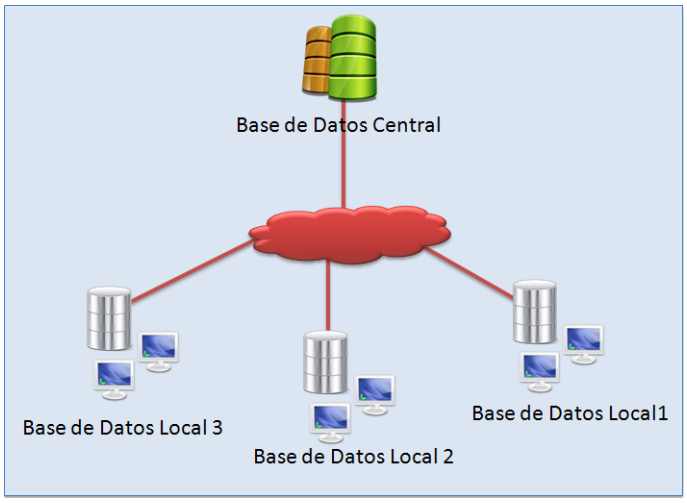

Figura 2: Sistema de bases de datos distribuidas.

Objetivos del diseño de la distribución de los datos

En el diseño de la distribución de los datos, se deben de tomar en cuenta los siguientes objetivos:

- **Procesamiento local:** La distribución de los datos, para maximizar el procesamiento local corresponde al principio simple de colocar los datos tan cerca como sea posible de las aplicaciones que los utilizan.
- **Distribución de la carga de trabajo:** La distribución de la carga de trabajo sobre los sitios, es una característica importante de los sistemas de cómputo distribuidos. Esta distribución de la carga se realiza para tomar ventaja de las diferentes características o utilizaciones de las computadoras de cada sitio, y maximizar el grado de ejecución de paralelismo de las aplicaciones.
- **Costo de almacenamiento y disponibilidad:** La distribución de la base de datos refleja el costo y disponibilidad del almacenamiento en diferentes sitios. Para eso, es posible tener sitios especializados en la red para el almacenamiento de datos.

Ventajas y desventajas de los sistemas bases de datos distribuidas.

La principal ventaja en la construcción de sistemas bases de datos distribuidas es que garantizan la fiabilidad y disponibilidad de los datos. Cuando se produce un fallo en una localidad, las demás localidades pueden seguir trabajando, y esto no implica necesariamente la desactivación del sistema. Se agiliza el procesamiento de consultas, al permitir que una consulta que solicite datos de varias localidades, sea dividida en varias subconsultas para ser ejecutadas paralelamente.

No obstante, tienen sus desventajas, por la difícil estructura de los sistemas bases de datos distribuidas, se necesitan más recursos, implicando un mayor costo de desarrollo de software. Puesto que las localidades del sistema distribuido operan paralelamente, es más difícil garantizar que los algoritmos estén correctos, lo que provoca una mayor posibilidad de errores. Además, necesitan un mayor tiempo extra de procesamiento, debido al intercambio de mensajes y los cálculos adicionales.

1.2 Sistemas gestores de bases de datos

El software que permite la utilización y actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema gestor de bases de datos. El objetivo fundamental de un sistema gestor de bases de datos consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o

sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado. (7)

Los programas de aplicación operan sobre los datos almacenados en la base utilizando las facilidades que brindan los sistemas gestores de bases de datos, los que en la mayoría de los casos, poseen lenguajes especiales de manipulación de la información que facilitan el trabajo de los usuarios. (7)

El más utilizado de estos lenguajes, que es soportado por la mayoría de los sistemas gestores de bases de datos, es el Lenguaje Estructurado de Consulta (SQL), que está compuesto por:

- El Lenguaje de Definición de Datos (Data Definition Language, DDL), usado para crear objetos en la base de datos.
- El Lenguaje de Control de Datos (Data Control Languaje, DCL), usado para controlar los permisos de los objetos de la base de datos.
- El Lenguaje de Manipulación de datos (Data Manipulation Languaje, DML), usado para consultar y modificar los datos.

Cada sistema gestor de bases de datos determina qué partes del SQL implementará y como lo hará. Existen muchas formas de organizar las bases de datos, pero hay un conjunto de objetivos generales que deben cumplir todos los sistemas gestores de bases de datos, de modo que faciliten el proceso de diseño de aplicaciones y que los tratamientos sean más eficientes y rápidos, dando la mayor flexibilidad posible a los usuarios. Los objetivos fundamentales de los sistemas gestores de bases de datos son: (7)

- Independencia de los datos y los programas de aplicación.
- Minimización de la redundancia.
- Integración y replicación de las bases de datos.
- Integridad de los datos.
- Seguridad y recuperación.
- Facilidad de manipulación de la información.
- Control centralizado.

En síntesis, un sistema gestor de bases de datos debe permitir definir la base de datos, especificando tipos, estructuras y restricciones de datos. Construir la base de datos, guardando los datos en algún medio controlado por el mismo sistema gestor de bases de datos y manipular la base de datos, realizando consultas, actualizando los datos y generando informes.

SQLServer

Microsoft SQL Server es un sistema gestor de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL. Entre sus características se encuentra el soporte de transacciones, la escalabilidad, estabilidad y seguridad. Soporta procedimientos almacenados. Incluye un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente. Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información. Además, permite administrar información de otros servidores de datos. (8)

Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos Windows. Usa Address Windowing Extension (AWE) para hacer el direccionamiento de 64-bit, esto le impide usar la administración dinámica de memoria y sólo le permite alojar a lo más 64Gb de memoria compartida. Permite aproximadamente 16 instancias distintas concurrentes en una máquina. No maneja compresión de datos, por tanto ocupa mucho espacio en disco. (8)

En sus últimas versiones permite acceder a los datos con las herramientas que se utilizan habitualmente como Microsoft Office 2007. Ayuda a la toma de decisiones con análisis predictivos generados mediante un entorno de minería de datos. Aumenta la disponibilidad con las Tecnologías “AlwaysOn” de SQL Server 2008, que aportan toda una familia de opciones destinadas a minimizar las pérdidas de servicio. Introduce mejoras orientadas a lograr una gestión eficaz de la configuración de seguridad y protección de la información, con un mecanismo potente de autenticación y control de accesos, cifrado de datos y gestión de claves, y un entorno de auditoría avanzado. (8)

Oracle

Oracle es un sistema gestor de bases de datos desarrollado por Oracle Corporation. Utiliza el lenguaje para consultas PL/SQL. Se considera como uno de los sistemas de bases de datos más completos. Contiene soporte de transacciones, estabilidad, escalabilidad y es multiplataforma. (9)

La seguridad de la plataforma y las políticas de suministro de parches de seguridad que incrementan el nivel de exposición de los usuarios han sido criticadas por algunos especialistas. La tecnología Oracle se encuentra prácticamente en todas las industrias alrededor del mundo. Oracle es la primera compañía de software que desarrolla e implementa software para empresas 100 por ciento activado por Internet a través de toda su línea de productos: bases de datos, aplicaciones comerciales y herramientas de desarrollo de aplicaciones y soporte de decisiones. A partir de la versión 10g Release 2, cuenta con 6 ediciones:

- Oracle Database Enterprise Edition(EE).
- Oracle Database Standard Edition (SE).
- Oracle Database Standard Edition One (SE1).
- Oracle Database Express Edition (XE).
- Oracle Database Personal Edition (PE).
- Oracle Database Lite Edition (LE).

La única edición gratuita es la Express Edition, que es compatible con las demás ediciones de Oracle Database 10gR2 y Oracle Database 11g.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[1] y con su código fuente disponible libremente. Sus características más importantes son la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez a el sistema. (10)

PostgreSQL implementa un subconjunto extendido de los estándares SQL92 y SQL99. Tiene funciones y procedimientos almacenados en numerosos lenguajes de programación. Entre ellos: PL/pgSQL, PL/Perl,

[1] *Licencia BSD: Es una licencia permisiva de software libre. Muy cercana al dominio público. Permite el uso del código fuente en software no libre.*

PL/Python y PL/Tcl C, C++, Java, Delphi, Python, Perl, PHP y Bash. Tiene Interfaz de Programación de Aplicaciones (APIs)[2] para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, Qt y muchos otros. (10)

Las transacciones permiten el paso entre dos estados consistentes manteniendo la integridad de los datos. Para garantizar la validez de los datos de la base de datos ofrece varios modos de bloqueo, para controlar el acceso concurrente a los datos en tablas. Los constraints[3] y disparadores tienen la función de mantener la integridad y consistencia en la base de datos. Soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario. Incorpora una estructura de datos Array. La conectividad puede ser TCP/IP, JDBC y ODBC. (10)

1.3 Replicación de datos

La replicación de datos es el proceso de compartir objetos y datos de una base de datos a múltiples bases de datos, en diferentes localizaciones. (11) En la Figura 3: Replicación de datos se muestra el proceso de intercambio de datos, el cual consiste en capturar la modificación, y transmitirla a todas las instancias de la base de datos, para luego aplicarla en cada una de ellas.

Figura 3: Replicación de datos.

1.3.1 Entornos de replicación

La replicación de datos se puede apreciar en diversos entornos, dependiendo del gestor de base de datos y el sistema operativo que se utilice en cada uno de los servidores fuentes y destino involucrados, los cuales pueden ser entorno homogéneo y entorno heterogéneo. (3)

[2] *API: Es el conjunto de funciones y procedimientos o métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas.*

[3] *Constraints: Son restricciones para limitar el tipo de dato que puede ingresarse en una tabla. Pueden especificarse cuando la tabla se crea por primera vez o luego de crear la tabla.*

Entorno Homogéneo

La replicación se ejecuta entre servidores de datos con el mismo gestor y sobre el mismo sistema operativo, o entre servidores de datos con el mismo gestor pero donde los sistemas operativos son diferentes.

Ejemplo1: Entre plataformas Windows y Linux con PostgreSQL instalado.

Ejemplo2: Entre plataformas Linux con PostgreSQL instalado.

Entorno Heterogéneo

La replicación se ejecuta entre servidores de datos con diferentes gestores de datos y el mismo sistema operativo, o el caso de tener una réplica de datos entre servidores con diferentes gestores y diferentes sistemas operativos.

Ejemplo 1: PostgreSQL vs Oracle o SQL Server vs PostgreSQL ambos corriendo en Windows.

Ejemplo 2: PostgreSQL vs Oracle o SQL Server vs PostgreSQL uno en Windows y otro en Linux.

1.3.2 Sentido de la Réplica

La réplica de datos puede aplicarse de dos maneras diferentes, de acuerdo con las necesidades propias de las personas o instituciones que la llevan a cabo, estas son:

Maestro-esclavo

La replicación Maestro-esclavo permite que un sólo servidor maestro pueda recibir consultas de lectura/escritura, mientras que los servidores esclavos sólo pueden realizar consultas de lectura**.** (10) Los datos pueden ser modificados en múltiples ubicaciones, entonces la replicación debe procesar los cambios realizados en cada uno de los sitios de forma coordinada. El servidor maestro es el que se encarga de distribuir los cambios a todos los sitios. Los cambios realizados en los destinos fluyen hacia los otros sitios a través del servidor maestro.

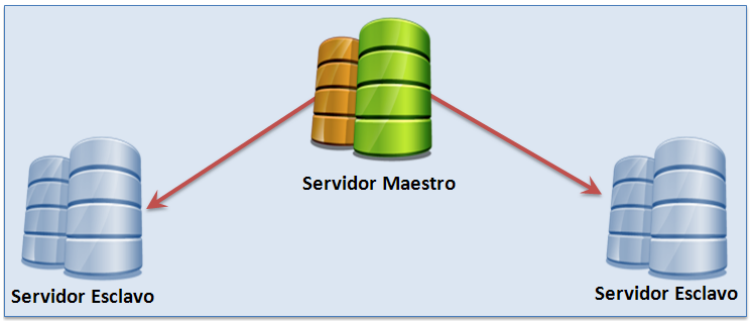

Figura 4: Maestro-Esclavo.

Los datos se replican entre servidores y clientes para admitir las siguientes aplicaciones:

- Intercambiar datos con usuarios en otras localizaciones.
- Aplicaciones de punto de venta para el consumidor.
- Integrar datos de varios sitios.

Multi-maestro

La replicación Multi-maestro permite enviar consultas de lectura/escritura a múltiples servidores replicados. Esta capacidad tiene un considerable impacto en el rendimiento debido a la necesidad de replicar los cambios entre servidores. (10) Esta réplica no tiene designado un servidor maestro, cada ubicación copia los cambios desde todos los otros sitios directamente. Cada sitio es un sitio maestro, y se comunica con otros sitios maestros. Puede ser usada para mantener sitios recuperables ante posibles desastres o caídas, así como para proveer sistemas con alta disponibilidad y para balancear la carga de consultas a través de las distintas ubicaciones.

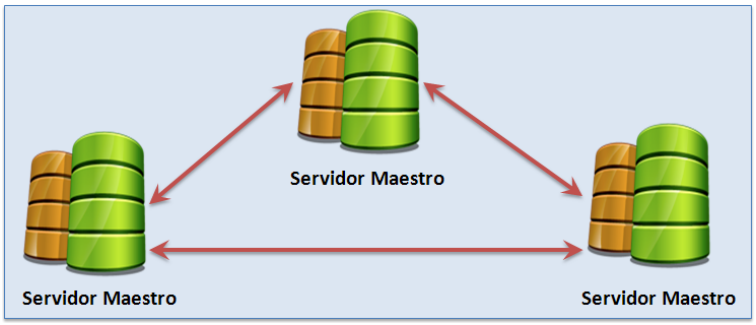

Figura 5: Multi-Maestro.

Los datos se replican entre servidores para:

- Mejorar la escalabilidad y la disponibilidad.

- Almacenar datos e informes.
- Integrar datos de varios sitios.
- Integrar datos heterogéneos.

1.3.3 Tiempo de Latencia

Aparte de los datos necesarios para la administración es interesante que las réplicas sepan la medida de latencia con respecto a la fuente primaria. La medida de latencia es la cantidad de tiempo que una réplica puede estar inconsistente hasta llegar a estar consistente con la fuente primaria designada. (11)

Replicación sincrónica

La replicación sincrónica es llamada de "consistencia fuerte" entre los datos almacenados, donde el grado de desactualización de los datos replicados respecto a los datos originales es casi nula, es decir, la latencia es aproximadamente cero. Estas transacciones son globales, también llamadas "todo o nada". En ellas, una transacción no debería ser aceptada si alguno de los sitios que intervienen no está de acuerdo. Respetan las propiedades ACID[4], pues son atómicas, consistentes, aisladas y los cambios son durables.

El proceso de actualización se dispara en el momento en que una réplica es modificada, se establece contacto con el servidor responsable de la réplica y se le informa de la actualización. El servidor actualiza la copia primaria del dato modificado y la información es enviada a todos los sitios donde existe una copia de esa réplica. (12)

La replicación sincrónica se caracteriza porque permite mantener un alto nivel de consistencia entre las réplicas. Genera un alto grado de sobrecarga en la red y no permite que los usuarios puedan trabajar desconectados. Un sistema que opere con este mecanismo de replicación sólo soporta clientes tradicionales ya que no permite manejar clientes móviles. (12) Si el procedimiento falla en cualquier sitio, entonces la transacción entera se anula.

Replicación asincrónica

La replicación asincrónica es llamada de "consistencia débil" entre los datos almacenados, ya que existe una desactualización de la copia replicada respecto a la original, es decir, existe una cierta latencia. Generalmente la latencia se mide en segundos si la infraestructura apropiada de recursos es suficiente.

[4] *ACID: expresa la función que las transacciones desarrollan en aplicaciones críticas para una misión. Responde a los términos atomicidad (atomicity), coherencia (consistency), aislamiento (isolation) y permanencia (durability).*

El proceso de actualización de réplicas se dispara a intervalos regulares de tiempo. Requiere que todos los servidores almacenen localmente una bitácora con las modificaciones que se van realizando sobre las réplicas. Una vez que inicia el proceso de replicación con los servidores primarios, las bitácoras son enviadas a dichos servidores para que realicen el proceso de reconciliación de réplicas para determinar sus valores actuales. (12)

La replicación asincrónica se caracteriza porque permite un alto nivel de autonomía en los sitios. Un usuario puede trabajar sobre las réplicas sin estar conectado a la red. Las modificaciones se siguen registrando en la bitácora local y una vez que se reconecta se incorporan sus modificaciones locales en el siguiente período de replicación. Por esa misma razón el nivel de consistencia de los datos es bajo. En un momento determinado dos usuarios distintos pueden usar diferentes valores para el mismo dato. (12)

Típicamente el intervalo de tiempo entre períodos de replicación es un parámetro del sistema. Si el intervalo es muy corto, se simula el funcionamiento de la replicación sincrónica. Si se usa un intervalo grande, se tiene entonces un ambiente con una mayor autonomía en los sitios de trabajo. En general, la longitud del intervalo se debe acondicionar a las características de la carga de trabajo, estimando el volumen de usuarios tradicionales y móviles del sistema. (12)

1.3.4 Tipos de replicación asincrónica

Basada en control de cambios

Se crea una estructura, principalmente formada por disparadores, funciones y tablas, donde se almacenan datos de control. Esta estructura es capaz de controlar los cambios que se realizan en los datos de las bases de datos en cuestión.

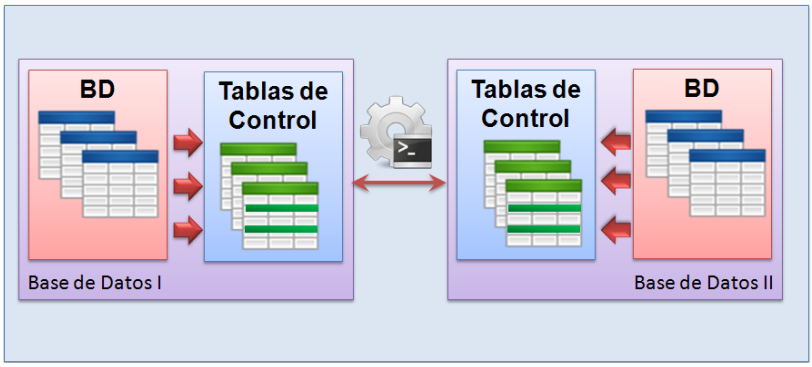

Figura 6: Réplica de datos basada en control de cambios.

Basada en lectura de log

La herramienta de réplica basada en lectura de log se encarga de analizar los logs generados por los sistemas gestores de bases de datos y es capaz de detectar los cambios realizados en la base de datos.

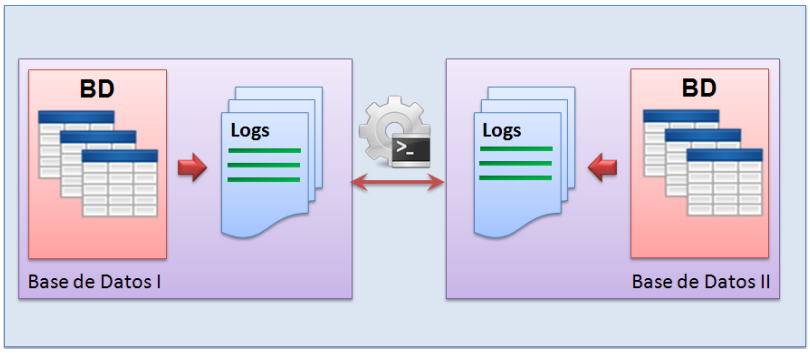

Figura 7: Réplica de datos basada en lectura de logs.

¿En qué momento realizar la replicación asincrónica?

El sistema de replicación podría proveer varias alternativas para elegir el momento de replicar que pudieran ser las siguientes:

- Inmediatamente como sea posible.
- Programado.
- Cuando se lancen los disparadores por operaciones de inserción (*insert*), actualización (*update*) y eliminación (*delete*).
- Bajo un control manual.

1.3.5 Conflictos de la Replicación

Conflicto de actualización: Ocurre cuando una fila está siendo actualizada por dos transacciones diferentes simultáneamente.

Conflicto de unicidad: Ocurre cuando la replicación de un registro intenta violar una restricción de integridad, ya sea por llave primaria o única (Primary Key o Unique). Por ejemplo cuando dos transacciones originadas de dos sitios diferentes, insertan cada una un registro, en su respectiva tabla, con el mismo valor de clave primaria.

Conflicto de Eliminación o Supresión: Ocurre cuando una transacción intenta eliminar un registro mientras que otra transacción lo está actualizando o eliminando.

Conflicto de orden: Ocurre cuando el orden de la propagación a los sitios se ve alterado. Suele producirse en ambientes de replicación con más de tres sitios maestros. Si por alguna razón la propagación a un sitio determinado se ve afectada o bloqueada, entonces la replicación de las modificaciones en los datos puede continuar su propagación a través de los sitios maestros, y al finalizar, esas modificaciones debieron ser propagadas al determinado servidor en un orden diferente de como ocurrieron en los sitios maestros. (13)

Para evitar y resolver estos conflictos los gestores de bases de datos y las herramientas de replicación aplican diferentes métodos de resolución de conflictos.

1.3.6 Fragmentación

Las bases de datos están constituidas principalmente por tablas, relaciones y funciones o procedimientos almacenados. Las tablas almacenan los datos en cada una de sus tuplas o filas, mientras que sus columnas representan los atributos, viéndolas como listados de objetos.

El diseño de la fragmentación se determina por la forma en que las relaciones globales se subdividen en fragmentos horizontales, verticales o mixtos. El problema de la fragmentación se refiere al particionamiento de la información para distribuir cada parte a los diferentes sitios de la red. Existen tres tipos de fragmentación:

1. Fragmentación horizontal
2. Fragmentación vertical
3. Fragmentación híbrida

Las técnicas de réplica y fragmentación se pueden aplicar sucesivamente a la misma relación de partida. Un fragmento se puede replicar y a su vez esa réplica ser fragmentada, para luego replicar alguno de esos fragmentos.

Fragmentación horizontal

Es donde se distribuyen los registros de las tablas de la base de datos entre diferentes nodos de la red. Consiste en el particionamiento en tuplas de una relación global en subconjuntos, donde cada subconjunto puede contener datos que tienen propiedades comunes y se puede definir expresando cada fragmento como una operación de selección sobre la relación global. Para fragmentar horizontalmente los datos, una

Tabla T se divide mediante operaciones de selección en subconjuntos T1, T2,…Tn. Cada fragmento se ubica en un nodo o bases de datos, y se reconstruyen con operaciones de unión.

Fragmentación vertical

Es donde se distribuyen los atributos de acuerdo con la frecuencia de su uso. Consiste en la subdivisión de atributos en grupos. Los fragmentos se obtienen proyectando la relación global sobre cada grupo. La fragmentación es correcta si cada atributo se mapea en al menos un atributo del fragmento. Para fragmentar verticalmente, una tabla se divide en subconjuntos. Los fragmentos se definen a través de una operación de proyección. Cada fragmento debe incluir la llave primaria de la tabla. Su reconstrucción se realizará con una operación de unión de los fragmentos componentes. En este contexto, una fragmentación "óptima" es aquella que produce un esquema de fragmentación que minimiza el tiempo de ejecución de las consultas de usuario.

Fragmentación híbrida

Consiste en aplicar la fragmentación vertical seguida de la fragmentación horizontal o viceversa, o sea, incluye una o varias secuencias de aplicación de las fragmentaciones anteriormente mencionadas. Ya que en muchos casos una fragmentación horizontal o vertical de un esquema de una base de datos, no será suficiente para satisfacer los requerimientos de aplicaciones de usuario.

1.4 Herramientas de réplica multi-maestra asincrónica existentes

Pyreplica

Es una herramienta de replicación maestro-esclavo y multi-maestro limitado, donde se replica en ambas direcciones, es decir, que cada base de datos es maestro y esclavo al mismo tiempo. Permite la replicación asincrónica de datos para PostgreSQL, utilizando disparadores implementados en pl/python[5], señales, secuencias y un script cliente influenciado por Slony-I[6] pero más simple y fácil. Está programado en Python[7] por lo que es simple y flexible, permite una fácil instalación, simplemente ejecutando un script SQL en el servidor, y copiando un script demonio[8] en el cliente, para lo que no se requiere compilación.

[5] *pl/python: Es un lenguaje procedural que permite escribir funciones python para la base de datos relacional PostgreSQL*

[6] *Slony-I: Es un sistema de replicación maestro/esclavo, asincrónico*

[7] *Python: Es un lenguaje de programación interpretado.*

[8] *Demonio o Daemon: Es un proceso informático que se ejecuta en segundo plano.*

No se necesita aprender nuevos comandos para su administración y es muy fácil de adaptar manualmente. Tiene un bajo impacto en el uso de memoria y la red al no utilizar transmisión secuencial (polling)[9] y es multiplataforma.

Ofrece al usuario la detección de conflictos y notificaciones vía e-mail. Permite el monitoreo de las conexiones (KeepAlive) y conexiones directas a las etapas finales (backends). Está protegido con transacciones en dos fases. Advierte al detectar conflictos de actualización/eliminación y falla en conflictos de inserción o errores de integridad de datos, pero no los corrige. Para la replicación de cambios de esquema, los comandos CREATE, ALTER, entre otros**,** deben ejecutarse manualmente en todos los servidores. El control de fallo no es automático y no hay soporte para objetos grandes**.** (14)

SymmetricDS

Es un software de replicación de datos asincrónico, puede ser maestro-esclavo y multi-maestro. Permite subscriptores múltiples y replicación bidireccional. Utiliza tecnologías web y de bases de datos para replicar tablas entre bases de datos relacionales casi en tiempo real. Fue diseñado para escalar a un gran número de bases de datos, trabajar con conexiones de bajo ancho de banda, y resistir a períodos de inoperatividad de la red. El software se instala de modo autónomo, como una aplicación web dentro del servidor de aplicaciones Java, o puede ser incorporado a otra aplicación Java. Un nodo es inicializado mediante un fichero *.properties*, y es configurado insertando datos de configuración en una serie de tablas de base de datos. A continuación, el nodo crea disparadores de base de datos en las tablas de aplicación especificadas, de modo que los eventos son capturados para ser entregados a otros nodos SymmetricDS. (15)

Soporta la replicación entre diferentes plataformas de base de datos, mediante el concepto de dialectos de base de datos. Un dialecto de base de datos es una capa de abstracción con la cual interactúa SymmetricDS para aislar la lógica de replicación de los detalles de implementación específicos de cada base de datos. Requiere tener instalada la máquina virtual de Java. (15)

Bucardo

Es un sistema de replicación asincrónico para PostgreSQL que permite tanto replicación multi-maestro como maestro-esclavo de fácil configuración. No requiere modificación de la instalación de PostgreSQL, y

[9]*Polling: Es una operación de consulta constante.*

se ejecuta como un demonio en Perl[10] que se conecta a la base de datos de control y todas las bases de datos que se repita, las actualizaciones pueden suceder más rápido ya que los cambios de datos no son rastreados. Para instalar Bucardo, se necesita, hacer e instalar los módulos Perl, crear la base de datos y crear el esquema de importación.

Permite que una base de datos PostgreSQL sea replicada en otra base de datos PostgreSQL, agrupando tablas a la manera de transacción segura. Las versiones más recientes emplean nuevas características, incluyendo un modelo robusto de demonio, configuración y logueo más flexibles, rutinas personalizadas de manejo de excepciones y conflictos, tiempos de replicación mucho más rápidos, y un alto nivel de auto-mantenimiento. Requiere Postgres 8.1 o superior, con Pl/Perl[11] y Pl/pgsql[12]. Sólo replica tablas, no la base de datos entera, no replica DDL y requiere una llave primaria sobre cada tabla para ser replicada. (16)

Magic@ Data ReplicationeXtensibleSolution

Es una solución de réplica multi-maestra asincrónica que se basa en el control de cambios en las bases de datos relacionales y la replicación de las mismas. Brinda solución a la problemática de la réplica entre bases de datos implementadas en cualquier gestor y es multiplataforma. Permite la implementación de sistemas de bases de datos distribuidas, aislando la complejidad del sistema del mecanismo de replicación de los datos. Simplifica y optimiza el proceso de replicación de la información entre servidores de bases de datos. Permite una réplica bidireccional y heterogénea. Utiliza disparadores para el mantenimiento de las tablas de control, sobre las tablas de la base de datos. Las tablas y disparadores son generados automáticamente por una herramienta que utiliza el esquema de datos de las tablas que participan en la réplica. El proceso de captura de modificaciones se divide en dos subprocesos, que se realizan de forma independiente: el subproceso de control de modificaciones y el subproceso de descubrimiento de estas modificaciones. (17)

Una de las posibilidades que brinda este mecanismo de réplica es que permite definir reglas para el filtrado de las tuplas a replicar, se pueden definir expresiones booleanas SQL para filtrar las tuplas que

[10]*Perl: Es un lenguaje imperativo, con variables, expresiones, asignaciones, bloques de código delimitados por llaves, estructuras de control y subrutinas.*

[11]*Pl/Perl: Es un lenguaje procedural soportado por el gestor de bases de datos PostgreSQL que permite crear funciones y utiliza la mayoría de las características del lenguaje Perl.*

[12]*Pl/pgsql: Es un lenguaje imperativo provisto por el gestor de bases de datos PostgreSQL que permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones.*

pueden ser replicadas. La solución permite implementar diferentes arquitecturas de base de datos distribuidas, tanto arquitectura con servidores que centralizan toda la información, o redes de replicación con servidores que según reglas se mantienen actualizados entre sí. (17)

La implementación de esta solución de réplica, disminuye la complejidad de un sistema distribuido, abstrayendo a los desarrolladores de la arquitectura de base de datos utilizada. Posibilita la configuración de distintas arquitecturas de base de datos distribuidas sin tener que realizar cambio en el modelo de negocio. El producto DBSynchronize implementa este mecanismo de replicación y permite diferentes configuraciones según el objetivo que se quiera cumplir. (17)

1.5 Tecnologías actuales

1.5.1 Tendencia al software libre

Se denomina software libre al que otorga a los usuarios las libertades de usarlo con cualquier propósito, estudiar su funcionamiento y adaptarlo a las necesidades, distribuir copias, y publicar sus mejoras en beneficio de la comunidad. La Fundación del Software Libre (FSF), se dedica a eliminar las restricciones sobre la copia, redistribución, entendimiento y modificación de los programas de computadoras promocionando el uso y desarrollo del software libre. Software Libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. (18)

1.5.2 Sistema de control de versiones Subversion

Subversion (svn) es un software de sistema de control de versiones. Es software libre bajo una licencia de tipo Apache/BSD. Con Subversion se puede acceder al repositorio a través de redes, lo que le permite ser usado desde distintas computadoras. La posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se envían sólo las diferencias en ambas direcciones. Subversion es reconocido como el único líder en la categoría de sistema de control de versiones.

1.5.3 Lenguaje de programación Java

Java, es un lenguaje orientado a objetos, con plataforma independiente. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir muchos errores, como la manipulación directa de punteros o memoria. La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor, como

de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras, ejecutando tareas simultáneamente, y de esa forma logra distribuir el trabajo a realizar. (19)

1.6 Solución Propuesta

Como resultado de la investigación realizada se decidió seleccionar para la implantación de la solución de réplica multi-maestra asincrónica las siguientes herramientas y tecnologías:

PostgreSQL 8.4 (PostgreSQL) como gestor de bases de datos ya que proporciona estabilidad, potencia, robustez, facilidad de administración e implementación de estándares lo que lo convierte en el sistema de gestión de bases de datos de código abierto más potente del mercado. Como se explicó anteriormente, funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema, lo cual es una de las características que más se explotan en sistemas de replicación y alta disponibilidad. Con PostgreSQL no se violan los acuerdos de licencia, puesto que no hay costo asociado a la licencia del software. Tiene una importante comunidad de profesionales alrededor del mundo con la que se puede obtener beneficios. El código fuente está disponible para todos. Si se necesita extender o personalizar PostgreSQL de alguna manera, se puede hacer sin costos adicionales. Además, es multiplataforma y está diseñado para ambientes de alto volumen.

PGAdmin III para la administración de PostgreSQL por ser la aplicación gráfica para la gestión de PostgreSQL más completa y popular. Tiene licencia Open Source. La interfaz gráfica soporta todas las características de PostgreSQL y facilita en gran medida la administración. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La aplicación incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, entre muchas otras funcionalidades. La conexión al servidor puede hacerse mediante TCP/IP y puede encriptarse mediante SSL para mayor seguridad. Se encuentra disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Windows, Linux, FreeBSD, Mac OSX y Solaris. (20)

Las herramientas estudiadas para llevar a cabo la propuesta y los criterios fundamentales que se tuvieron en cuenta se muestran en la Tabla 1: Herramientas de réplica.

Soluciones	Tiempo de Latencia	Sentido de la Réplica	Licencia	Principales Plataformas

Slony-I	Sincrónico	Maestro-Esclavo	BSD	Linux
PgCluster	Sincrónico	Multi-Maestro	DEB	Linux
CyberCluster	Sincrónico	Multi-Maestro	BSD	Windows, Linux
Pyreplica	Asincrónico	Multi-Maestro, Maestro-Esclavo	GPL	Windows, Linux
SymmetricDS	Asincrónico	Multi-Maestro	LGPL	Windows, Linux
Bucardo	Asincrónico	Multi-Maestro, Maestro-Esclavo	BSD	Linux
Magic@ Data Replication eXtensible Solution	Asincrónico	Multi-Maestro	Desarrollado en la UCI	Windows, Linux

Tabla 1: Herramientas de réplica.

Las herramientas Slony-I, PgCluster y CyberCluster no satisfacen los requerimientos para el desarrollo de la solución de réplica multi-maestra asincrónica. Las herramientas candidatas son: PyReplica, SymmetricDS, Bucardo y Magic@ Data Replication eXtensible Solution.

PyReplica no se selecciona porque su condición de multi-maestro es limitada, ya que tiene restricciones de solamente dos servidores maestros. Aunque existe una variante de configuración como se muestra en la Figura 8: Variante de configuración multi-maestra de PyReplica. Así sería el modelo de trabajo en la configuración para 3 servidores, donde cada servidor es maestro para otro servidor:

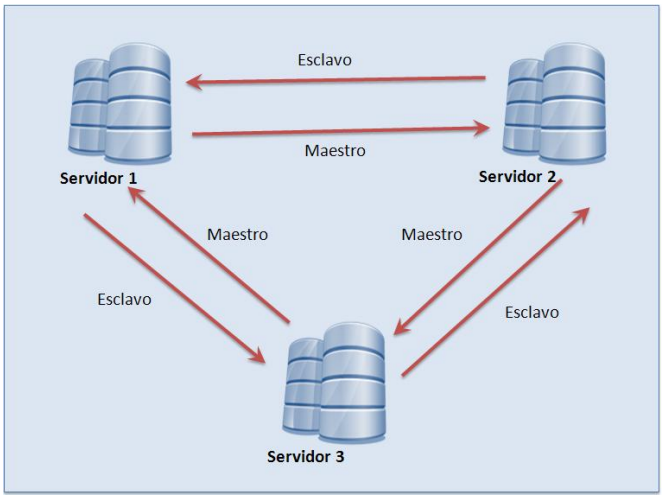

Figura 8: Variante de configuración multi-maestra de PyReplica.

- El servidor 1 es maestro para el servidor 2 y esclavo para el servidor 3.
- El servidor 2 es maestro del servidor 3 y esclavo del servidor 1.
- El servidor 3 es maestro para el servidor 1 y esclavo para el servidor 2.

Pero esta configuración presenta algunos problemas de integridad, ya que fallan a veces las transacciones y posee problemas de rendimiento, puesto que las conexiones se reinician a menudo. Con esta configuración si alguno de los servidores pierde la conexión, la replicación de los datos se realizará entre los restantes en un sólo un sentido.

Magic@ Data Replication eXtensible Solution no se selecciona porque es una herramienta que no cuenta con una comunidad que le brinde soporte. Esto trae consigo que si se encuentra algún error en el uso de la herramienta no existe un grupo de desarrollo que pueda ayudar a corregirlo o lo subsane en versiones posteriores. Posee muy poca o casi ninguna documentación y no existe un sitio oficial de donde descargar la última versión estable. La única versión que consta no ha sido oficialmente probada en suficientes proyectos. El proceso de instalación es complejo debido a que no cuenta con un instalador y es necesario modificar manualmente un gran grupo de archivos y ficheros.

Bucardo no se selecciona porque requiere una llave primaria sobre cada tabla para ser replicada, es la misma característica que posee Pyreplica, esto puede provocar que en algunas ocasiones ocurra conflicto de llave primaria y no se realice satisfactoriamente la replicación. No replica DDL, es decir, que no reconoce las sentencias para definir los objetos de la base de datos que brinda el Lenguaje de Definición de Datos.

SymmetricDS, en su versión 2.0, es la herramienta que se propone por ser la que mejor responde a las características requeridas por las soluciones de réplica multi-maestra asincrónica. Es una herramienta bastante completa que cumple con una serie de funcionalidades deseables a la hora de efectuar la replicación. Tiene capacidad para escalar un gran número de bases de datos. Trabaja con conexiones de bajo ancho de banda y soporta largos períodos de interrupción de la red. Es una herramienta de código abierto, ampliable a través de puntos de extensión. Los puntos de extensión son personalizados y el código de Java reutilizable se configura a través de XML. Esto permite un comportamiento personalizado, como por ejemplo definir la publicación de datos de otras fuentes, la transformación de datos, y la adopción de medidas diferentes en función del contenido o el estado de una replicación. La frecuencia de la notificación de cambio es configurable, por defecto es a una vez por minuto, y se puede ajustar al tiempo que se desee, lo cual es muy conveniente para entornos donde la red sufre prolongados períodos de inactividad. SymmetricDS apoya el concepto de los canales de datos, la replicación de datos es definida en el nivel de la tabla (o un subconjunto de la tabla), y cada tabla administrada puede ser asignada a un canal que ayuda a controlar el flujo de datos. Una serie de filtros se prestan para hacer cumplir la autenticación y para restringir el número de flujos de replicación simultánea. Su uso está basado en HTTP o HTTPS, por lo que resulta ligero y fácil de manejar.

1.7 Conclusiones

Todos los conceptos y definiciones brindados anteriormente posibilitan una mejor comprensión del proceso de replicación de datos y de las soluciones que se han desarrollado para mejorarlo. Para lograr este objetivo se llevó a cabo un estudio de las bases de datos, los sistemas gestores de bases de datos y de todo el proceso de replicación, así como las ventajas y desventajas del mismo. Se realizó un análisis profundo de las herramientas de réplica multi-maestra asincrónica existentes. En cada una de las cuales fueron bien examinadas sus características, ventajas y desventajas, para llegar a una correcta selección y tener una propuesta de solución bien justificada.

CAPÍTULO 2 ANÁLISIS Y DISEÑO

Introducción

En el presente capítulo se describe detalladamente la propuesta de solución de réplica multi-maestra asincrónica para bases de datos. Se realiza el análisis y diseño de la misma, identificando los requerimientos necesarios, dejando claros los principales conceptos y detallando la serie de pasos a seguir. Se enfocan los términos de configuración de la herramienta seleccionada y se procede a la instalación de la misma, mostrando la estructura de los principales directorios dentro del paquete de instalación y las opciones de instalación que se utilizarán. Para realizar la configuración inicial, se implementa una herramienta de apoyo. Además, se abordará el tema de la seguridad en el transporte de los datos y el filtrado de direcciones IP.

2.1 Análisis de la propuesta de solución de réplica multi-maestra asincrónica para bases de datos

Para el análisis de la propuesta de solución de réplica se ha organizado el trabajo en los siguientes pasos:

- Identificación de los requerimientos.
- Identificación de los nodos.
- Organización de los nodos.
- Definición de los grupos de nodos.
- Vinculación entre los nodos.
- Elección de los canales de datos.
- Definición de los cambios en los datos a ser capturados y enrutados.
    - Definición de los disparadores.
    - Definición de los enrutadores.
- Planificación de la carga de los datos iniciales.

2.1.1 Identificación de los requerimientos necesarios

Los requerimientos son las condiciones o capacidades que deben ser alcanzadas o poseídas por un sistema o componente de un sistema para satisfacer un estándar o documento impuesto formalmente.

(21) A continuación se dan a conocer los requerimientos que debe cumplir la solución de réplica multi-maestra asincrónica.

RF 1 Soportar la implantación de la réplica de datos dividida por niveles.

La solución debe soportar la réplica de datos para organizaciones que tienen varios niveles ubicados en distintos lugares. Garantizando así que la información se encuentre actualizada y disponible en cada uno de los servidores ubicados en los distintos niveles.

RF 2 Garantizar la disponibilidad de los datos.

Los datos de la fuente de datos deben estar actualizados en los demás servidores para lograr que los procesos sean atendidos en una razón de tiempo razonable, siendo el 100% el mejor de los casos. Los usuarios autorizados tendrán garantizado el acceso a la información.

RF 3 Garantizar la tolerancia a fallos.

Se debe lograr que pueda existir un cierto número de fallos manteniendo el correcto comportamiento del sistema.

RF 4 Garantizar la coordinación entre cada una de las partes que intervienen en el proceso de replicación.

Todas las partes que componen la base de datos, llegan a un consenso para la solicitud de los servicios a los objetos, para que se realice tal y como fue solicitado al final de la transacción.

RF 5 Garantizar que el sistema funcione en entornos que pueden estar durante cierto período de tiempo desconectados.

Cuando la red es inestable o se puede pasar períodos de tiempo sin conexión, el sistema debe ser capaz de realizar la replicación cuando se restablezcan las condiciones.

RF 6 Brindar la opción de configurar cada cuánto tiempo se desea realizar la replicación.

El administrador debe tener la posibilidad de determinar cada cuánto tiempo se desea realizar la replicación, si desea que se active el proceso de forma manual, en un intervalo de tiempo determinado o en un horario específico.

RF 7 Permitir seleccionar qué parte de la base de datos se desea replicar.

El administrador debe tener la posibilidad de escoger qué parte de la base de datos desea replicar y en qué sentido, seleccionando las tablas, filas o columnas.

RF 8 Mantener la confidencialidad de la información.

La información manejada por el sistema debe estar protegida de accesos no autorizados.

RF 9 Permitir el registro de logs y estadísticas.

Permitir el registro de logs y estadísticas para proporcionar a los administradores de la base de datos información de los errores ocurridos en el sistema, alteraciones en la información, acceso de los nodos, entre otros registros, para cuando puedan ser de utilidad.

RF 10 Incluir soporte para tipos de datos de PostgreSQL.

La herramienta utilizada debe incluir soporte para los tipos de datos de PostgreSQL, incluidos los tipos de datos para almacenar archivos e imágenes. PostgreSQL implementa los tipos de datos definidos para el estándar SQL3, como se muestra en Anexo 1: Tipos de datos estándar en PostgreSQL.

2.1.2 Identificación de los nodos

Un nodo es una instancia de SymmetricDS, el cual se encarga de gestionar la replicación de los datos que se envían y reciben a la base de datos. Cada nodo puede estar embebido en otra aplicación, puede ejecutarse de manera independiente o incluso ejecutarse en segundo plano como un servicio. Si se desea, los nodos pueden ser agrupados para ayudar a balancear la carga cuando se envía o recibe grandes volúmenes de datos. Para cada base de datos, debe haber una instancia correspondiente, o sea, un nodo responsable de gestionar la replicación.

2.1.3 Organización de los nodos

Los nodos en la solución propuesta están organizados dentro de una red de nodos conectados entre sí, que necesitan replicar información. La organización exacta de los nodos es muy específica para los objetivos y características de cada sistema. Como punto de partida, se deben desplegar los nodos en un diagrama y establecer conexiones entre ellos para representar los casos en que existe flujo de datos. La información puede fluir desde el nivel principal a niveles inferiores y viceversa.

2.1.4 Definición de los grupos de nodos

Una vez que se han identificado cada uno de los nodos, es necesario agruparlos basándose en qué nodos comparten funcionalidades comunes. Esta actividad se establece en el sistema de réplica a través de grupos de nodos. Muy a menudo, los grupos de nodos coinciden con cada uno de los niveles, individuales del sistema. Los grupos de nodos son una pieza clave del sistema ya que muchas de las funcionalidades de la herramienta a utilizar son especificadas a través de los grupos de nodos y no a un nodo individual, por esa razón la importancia de una correcta selección de los mismos. Por ejemplo, cuando es hora de decidir hacia dónde se replicarán los datos, esta acción es configurada a través de los grupos de nodos, y se denomina enrutamiento. Para seleccionar los grupos de nodos se debe tener en cuenta que cada conjunto de nodos en un grupo tiene en común las mismas tablas para ser replicadas.

2.1.5 Vinculación entre los grupos de nodos

Una vez establecidos los grupos de nodos, el próximo paso en el diseño del sistema de réplica es identificar el vínculo entre ellos. Los vínculos están compuestos esencialmente por un grupo origen, un grupo destino y el tipo de vínculo, que puede ser para recibir o enviar. El vínculo que se establece para recibir, hace que el grupo origen se conecte al grupo destino, mientras el vínculo que se establece para enviar, espera a que el grupo destino se conecte al grupo origen para enviarle los datos. En la Figura 9: Vinculación entre los grupos de nodos., se muestra un ejemplo de cómo establecer los vínculos entre grupos de nodos en un escenario con varios niveles.

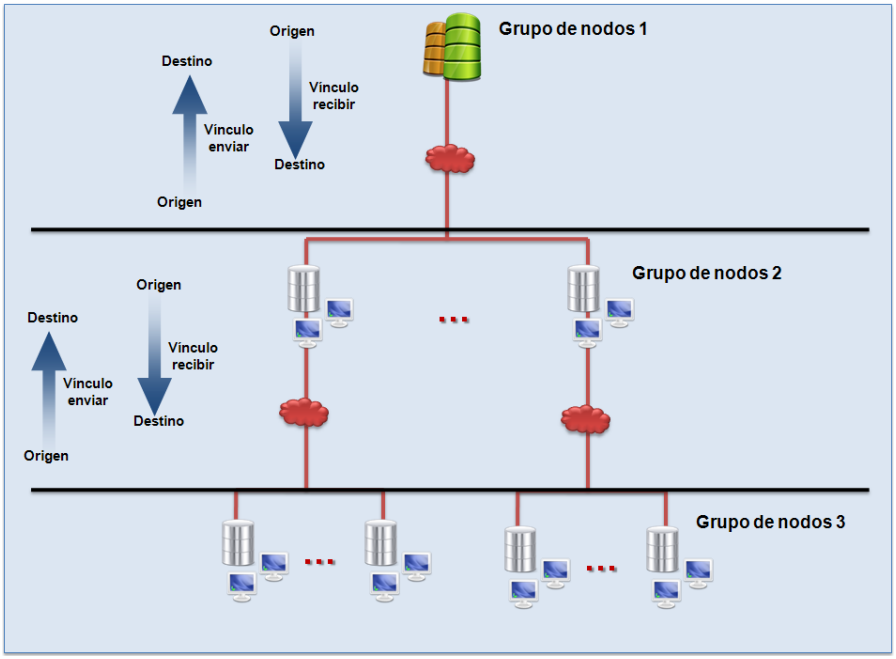

Figura 9: Vinculación entre los grupos de nodos.

2.1.6 Elección de los canales de datos

En el sistema de réplica, los cambios en la base de datos son capturados en el mismo orden en que ocurren, y ese orden se mantiene al realizar la réplica con otros nodos. Sin embargo, frecuentemente se presentan tipos de datos con diferentes prioridades. Por ejemplo los datos que no pueden ser retrasados debido a una gran carga de volumen u otro tipo de necesidad, tienen prioridad alta y precisan ser replicados por encima del orden normal. La herramienta seleccionada para la solución de réplica ofrece agrupar las tablas en canales de datos, lo que permite:

- Proveer la prioridad que tendrán los datos a la hora de realizar la replicación.
- Establecer la cantidad máxima de datos a agrupar, para ser enviados juntos por ese canal en el momento de la replicación.
- El aislamiento de los errores en los canales, esto ayuda a una mejor comprensión y corrección de los mismos.

Al clasificar los datos en canales y asignarlos a disparadores se obtiene un mayor control y visibilidad del flujo de información. Los canales pueden ser habilitados o deshabilitados. La frecuencia de replicación puede ser controlada a nivel de canal. Uno de los aspectos más importantes a la hora de definir los

canales de datos es asegurarse de que todas las tablas relacionadas por llave foránea sean incluidas en el mismo canal.

2.1.7 Definición de los cambios en los datos a ser capturados y enrutados

En este punto, se debe definir cuáles son los cambios que serán capturados, especificar los nodos que van a enrutar estos cambios y en qué condiciones. Esto se logra mediante la definición de los disparadores y enrutadores, que se explica a continuación:

Definición de los disparadores (triggers)

En el sistema de réplica propuesto se utilizan los disparadores para capturar y guardar los cambios que serán replicados con otros nodos basados en la configuración que se ha establecido. A cada disparador definido se le asocia una tabla y se puede especificar:

- Si se desea instalar el disparador para la inserción, modificación y/o eliminación.
- Las condiciones con las que la inserción, modificación y/o eliminación tienen lugar.
- Un listado de las columnas pertenecientes a la tabla asociada al disparador que no deben ser replicadas.

A la hora de definir los disparadores se debe considerar cuáles cambios en los datos son relevantes para el sistema de réplica y cuáles no. Es importante establecer bajo qué condiciones se desea enrutar estos cambios.

Definición de los enrutadores (routers)

Los disparadores solamente definen cuándo los cambios en los datos van a ser capturados para la replicación, pero no definen hacia dónde se enviarán. Los enrutadores, más una correlación entre disparadores y enrutadores son los que definen el proceso de determinar cuáles grupos de nodos recibirán los cambios en los datos. Para cada enrutador se debe especificar:

- La tabla destino que se encuentra dentro del nodo de destino hacia donde se enrutarán los datos.
- El grupo de nodos origen y destino que intervienen en el enrutamiento.
- El tipo de enrutamiento y la expresión de enrutamiento.
- Si debe enrutar actualización, inserción o eliminación.

Para cada uno de los disparadores se debe decidir qué enrutador le corresponde, en dependencia del comportamiento del disparador. Estas combinaciones se usan para definir la correlación entre disparadores y enrutadores al implementar el diseño.

2.1.8 Planificación de la carga inicial

La correlación entre disparadores y enrutadores implica más que una simple relación de muchos a muchos. Implica además, cómo y cuáles serán las operaciones de carga inicial que se llevarán a cabo desde el nodo raíz a un cliente. El concepto de carga inicial es usado para comenzar el proceso de replicación en la mayoría de los escenarios de réplica.

El proceso de carga inicial se lleva cabo cuando un nodo se conecta, luego se registra y si existe una petición de carga inicial, cada tabla que fue configurada para replicarse hacia un grupo de nodos destino determinado, recibirá un evento de recarga en el orden definido por el usuario.

2.2 Diseño de la propuesta de solución de réplica multi-maestra asincrónica para bases de datos

En el diseño de la propuesta de solución de réplica se hace alusión a cada uno de los puntos desarrollados en el análisis, pero enfocándose en términos de configuración e implementación. Se documenta cómo configurar la herramienta seleccionada para darle respuesta a las especificaciones detectadas en el análisis y se abordan los principales conceptos para la implantación de la solución.

En la Figura 10: Modelo de datos, aparecen las tablas más importantes que serán usadas, con todos los campos que poseen y las relaciones existentes entre ellas. El modelo de datos facilitará en gran medida la comprensión del diseño, proporcionando una vista de las tablas y los campos que se requieren para la correcta configuración.

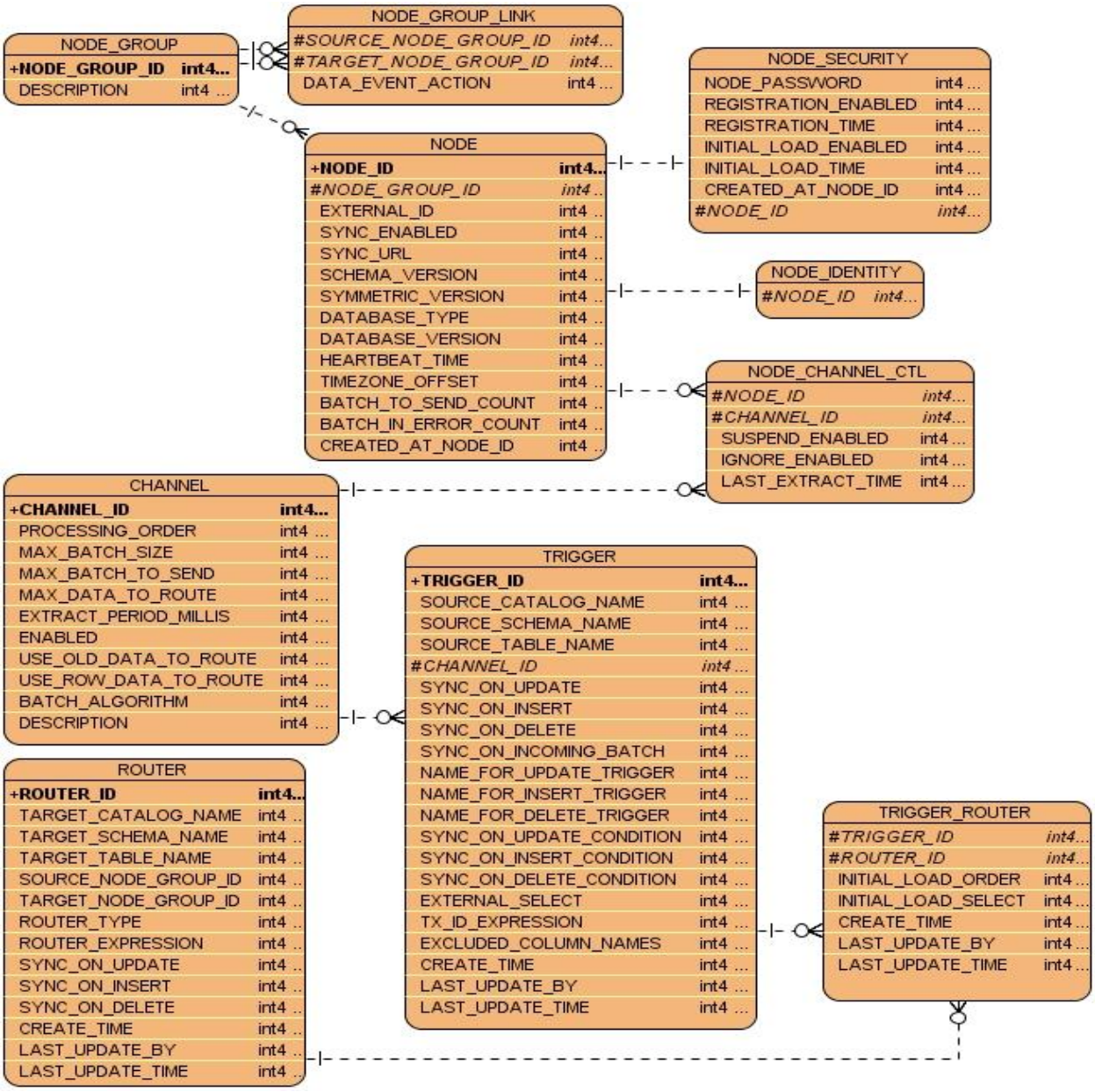

Figura 10: Modelo de datos.

2.2.1 Propiedades de un nodo

Recordando que un nodo es una instancia de SymmetricDS, que se encarga de gestionar la replicación de los datos que se envían y reciben a la base de datos. Para que un nodo se ejecute, necesita tener un identificador único, y conocer cómo conectarse a la base de datos para realizar la replicación. La forma de especificarlo es configurando las propiedades en el fichero de configuración con extensión *.properties*.

Cuando la herramienta seleccionada para la solución de réplica se inicia, toma la configuración y el estado de la base de datos. Si las tablas de configuración no se encuentran, la herramienta tiene la opción de crear automáticamente una configuración por defecto, esta opción puede ser habilitada o deshabilitada. La configuración básica se describe mediante las siguientes tablas:

NODE_GROUP: Especifica la información sobre los grupos de nodos existentes en el sistema de réplica, que generalmente coinciden con los niveles.

NODE_GROUP_LINK: Vincula dos grupos de nodos para la replicación.

CHANNEL: Define el agrupamiento y la prioridad para la replicación.

TRIGGER: Especifica las tablas, los canales y las condiciones con que los cambios en la base de datos deben ser capturados.

ROUTER: Especifica los enrutadores definidos para la replicación y otros detalles de enrutamiento.

TRIGGER_ROUTER: Provee la correlación entre disparadores y enrutadores.

Durante el inicio, los disparadores se verifican con la base de datos, y se instalan en las tablas que requieren la captura de los cambios en los datos. Los enrutadores envían y reciben los cambios en los datos al comenzar el funcionamiento, para replicar los cambios con otros nodos. Cada nodo requiere propiedades que le permitan conectarse con la base de datos y registrarse con su nodo padre. Para asignarle a un nodo su identificador es necesario establecer las siguientes propiedades:

group.id: Define el id del grupo de nodos al cual pertenece el nodo. La replicación es establecida por medio de grupos de nodos, por lo que sólo se necesita especificar una vez para múltiples nodos ubicados en un mismo grupo.

external.id: Especifica el id externo del nodo, que tiene significado para el usuario y provee la integración en el sistema donde será desplegado.

sync.url: Define la dirección URL a la que otros nodos pueden conectarse para la replicación. Al inicio y durante la replicación los nodos se conectan y actualizan los datos mediante la URL especificada.

Cuando un nuevo nodo es inicializado por primera vez, no tiene información acerca de la replicación. El nodo contacta con el servidor de registro para unirse al sistema de réplica y recibir su configuración. La configuración para todos los nodos es almacenada en el servidor de registro, y la URL debe ser especificada en la siguiente propiedad:

registration.url: especifica la dirección URL donde el nodo puede conectarse para registrarse y recibir su configuración. El servidor de registro forma parte de la herramienta seleccionada y está habilitado como parte de la implementación.

La herramienta crea la conexión de la base de datos usando el driver JDBC[13] de Java. Los parámetros necesarios para establecer la conexión son los siguientes:

db.driver: El nombre de la clase del driver JDBC.

db.url: La cadena de conexión que será utilizada por el driver JDBC para conectarse a la base de datos. Ejemplo: http://10.36.19.33:5432/mibasedatos

db.user: El nombre de usuario de la base de datos, usado para acceder, crear y actualizar las tablas de la herramienta.

db.password: La contraseña para el usuario de la base de datos.

2.2.2 Definición de un Nodo

En la herramienta seleccionada para la solución de réplica, los nodos son definidos en la tabla **NODE**, donde se especifican, entre otras, las siguientes propiedades:

NODE_ID: El id del nodo.

NODE_GROUP_ID: El id del grupo de nodos al que pertenece.

EXTERNAL_ID: El id externo.

SYNC_URL: La dirección URL donde puede ser contactado para la replicación.

SYNC_ENABLED: Indica si la replicación para el nodo está habilitada o deshabilitada.

Las tablas NODE_IDENTITY y NODE_SECURITY también desempeñan un papel importante en la definición de un nodo.

La tabla **NODE_IDENTITY** tiene un único campo, NODE_ID, que es insertado cuando el nodo se registra con un nodo padre. En el caso de que sea un nodo raíz el campo debe ser insertado por el propio usuario. El campo es usado por una instancia del nodo para determinar el identificador del nodo.

La tabla **NODE_SECURITY** tiene filas creadas para cada nodo hijo que se registra con el nodo. Se recogen datos acerca de la contraseña usada para que los nodos hijos prueben su identidad con el nodo

[13] *JDBC (Java Database Connectivity): Es un driver que permite efectuar conexiones con un gran número de bases de datos.*

padre, en el campo NODE_PASSWORD. Se almacena si está registrado o no y el momento en que lo hizo, en los campos REGISTRATION_ENABLED y REGISTRATION_TIME Se registra si una carga inicial será enviada al nodo y el momento en que comienza, en los campos INITIAL_LOAD_ENABLED e INITIAL_LOAD_TIME. Además, se guarda el id del nodo padre.

2.2.3 Grupos de nodos y vínculo entre ellos

Los grupos de nodos se definen y se configuran en la tabla **NODE_GROUP**, que contiene los campos NODE_GROUP_ID y DESCRIPTION, para el identificador y la descripción del grupo de nodos. El vínculo entre los grupos de nodos se establece en la tabla **NODE_GROUP_LINK**, especificando en los campos:

SOURCE_NODE_GROUP_ID: el grupo de nodos origen, donde los cambios de datos deben ser capturados.

TARGET_NODE_GROUP_ID: el grupo de nodos destino, donde los cambios deben ser enviados.

DATA_EVENT_ACTION: el tipo de vínculo, que puede ser de enviar o recibir. Si es para enviar, el campo toma el valor “P” (Push), y si es para recibir, toma el valor “W” (Wait for Pull).

2.2.4 Canales de datos

Los datos son agrupados por canales, para ser transferidos juntos a los nodos clientes. Para realizar el agrupamiento de los datos existen tres formas diferentes, siguiendo un algoritmo que es especificado en el campo batch_algoritm de la tabla **CHANNEL**. Los valores posibles que puede tomar el campo batch_algoritm, o sea, las tres maneras de realizar el agrupamiento, son las siguientes:

Default: Todos los cambios que ocurren en una transacción son agrupados juntos. Las transacciones múltiples son agrupadas hasta que se terminen o se alcance el límite especificado en la configuración del canal, en el campo MAX_BATCH_SIZE, de la tabla CHANNEL.

Transactional: Cada transacción en la base de datos se convierte en un grupo, es decir, habrá tantos grupos como transacciones existan. En este caso el campo MAX_BATCH_SIZE no se toma en cuenta.

Nontransactional: Las transacciones múltiples son agrupadas hasta que se terminen o se alcance el límite especificado en MAX_BATCH_SIZE. Este caso es muy parecido al Default, pero tiene la especificación que si se llega al límite establecido en MAX_BATCH_SIZE, el grupo será cortado independientemente de que se encuentre en medio de una transacción.

La tabla **CHANNEL** tiene otros campos, algunos de ellos son:

CHANNEL_ID: Indica un identificador único para el canal.

PROCESSING_ORDER: Indica el orden de secuencia para procesar los datos en el canal.

MAX_BATCH_TO_SEND: Indica el número máximo de grupos a enviar en la replicación entre dos nodos. Por ejemplo, si hay 23 grupos listos para ser enviados a un canal, y MAX_BATCH_TO_SEND es igual a 20, sólo serán enviados los primeros 20 grupos.

MAX_DATA_TO_ROUTE: El número máximo de filas de datos que serán enviados a través de un canal al mismo tiempo.

ENABLED: Indica si el canal está habilitado o no.

2.2.5 Disparadores

Recordando que los disparadores son utilizados para capturar y guardar los cambios en la base de datos que serán replicados con otros nodos. En la herramienta seleccionada para la solución de réplica, los disparadores se definen en la tabla **TRIGGER**. Cada registro es usado por la herramienta cuando se generan los disparadores de la base de datos. Los disparadores son generados cuando se encuentran asociados a un enrutador, donde el campo SOURCE_NODE_GROUP_ID de la tabla ROUTER coincide con el nodo actual. En la tabla **TRIGGER**, se deben especificar, entre otros, los siguientes campos:

SOURCE_TABLE_NAME: El nombre de la tabla a la que el disparador estará asociado, para capturar los cambios en los datos.

CHANNEL_ID: El id del canal a través del cual fluyen los cambios en los datos.

SYNC_ON_UPDATE, SYNC_ON_INSERT, SYNC_ON_DELETE: Si el disparador va a capturar o no los cambios de actualización, inserción o eliminación de datos.

SYNC_ON_UPDATE_CONDITION, SYNC_ON_INSERT_CONDITION, SYNC_ON_DELETE_ CONDITION: Las condiciones en que el disparador será activado para la actualización, inserción o eliminación de datos, usando una expresión específica de la base de datos.

2.2.6 Enrutadores

Recordando que los enrutadores son utilizados para determinar cuáles nodos recibirán los cambios en los datos. En la herramienta seleccionada para la solución de réplica, los enrutadores se definen en la tabla **ROUTER**, que cuenta principalmente con los siguientes campos:

SOURCE_NODE_GROUP_ID: El id del grupo de nodos donde serán instalados los disparadores que se relacionan con el enrutador.

TARGET_NODE_GROUP_ID: El id del grupo de nodos hacia los que se realizará el enrutamiento.

ROUTER_TYPE: El tipo de implementación para el enrutador.

SYNC_ON_UPDATE, SYNC_ON_INSERT, SYNC_ON_DELETE: Indican que el enrutador debe enrutar o no: actualización, inserción y/o eliminación.

CREATE_TIME: Indica el momento en que se creó el enrutador.

LAST_UPDATE_BY: Indica el usuario que actualizó por última vez el enrutador.

LAST_UPDATE_TIME: Indica el momento en que se actualizó por última vez el enrutador.

La herramienta seleccionada para la solución de réplica provee diferentes tipos de implementaciones base para los enrutadores, algunas de ellas son las siguientes:

- **Default Router:** Envía todos los datos a todos los nodos que forman parte del grupo establecido en el campo TARGET_NODE_GROUP_ID de la tabla ROUTER.
- **Column Match Router:** Compara nuevos o antiguos valores de una columna con una constante o con los valores de los parámetros external_id o node_id.
- **Sub-select Router:** Ejecuta una expresión SQL, sobre la base de datos para seleccionar los nodos a enrutar. La expresión SQL se establece en el campo ROUTER_EXPRESSION, de la tabla ROUTER.

Entre el conjunto de disparadores y el conjunto de enrutadores existe una relación de muchos a muchos. Lo que significa que un disparador puede capturar los cambios y enrutarlos a varias localizaciones. Un mismo enrutador puede estar definido y asociado con múltiples disparadores.

2.2.7 Registro de nodos

Una vez que se ha definido un nodo y configurado sus propiedades, se han definido los grupos de nodos y el vínculo entre ellos, se han establecido los canales de datos, los disparadores y enrutadores, la replicación no será ejecutada hasta que se realice el registro de los nodos y la carga inicial de los datos. El proceso de registro de un nodo es el acto de creación de un nuevo nodo en las tablas **NODE** y **NODE_SECURITY**, de modo que cuando el nuevo nodo se pone en línea se le permite unirse al sistema. Los nodos sólo pueden registrarse si existen los campos para el nodo y el parámetro

REGISTRATION_ENABLED de la tabla NODE_SECURITY se encuentra activado en 1, esto indica que el registro está abierto para ese nodo. Si la propiedad *auto.registration* de la herramienta seleccionada, que se encuentra entre los archivos de propiedades de la base de datos, se activa en true, el registro será abierto automáticamente para los nodos que lo soliciten, verificando primero que no se encuentre registrado, pero por defecto, esta propiedad está establecida en false. Si se intenta registrar un nodo, y el registro ya existe para un nodo con el mismo id externo, el comando *open-registration* permitirá abrir un nuevo registro para ese nodo, con un nuevo id y el mismo id externo.

2.2.8 Carga inicial

La carga inicial es el proceso de inserción de tablas en un nodo destino con los datos de su nodo padre. La carga inicial no puede ocurrir hasta después que un nodo sea registrado. Una carga inicial se solicita mediante el establecimiento en 1 del campo INITIAL_LOAD_ENABLED, de la tabla **NODE_SECURITY**, que por defecto aparece en 0, en el campo para el nodo destino en la base de datos del nodo padre. Cuando se replique con el nodo destino, los grupos de recarga serán insertados, y al mismo tiempo todos los grupos pendientes para el nodo se marcan como enviados satisfactoriamente. La herramienta reconoce que una carga inicial se ha completado cuando el campo INITIAL_LOAD_TIME de la tabla NODE_SECURITY, en el nodo destino, se establece con un valor no nulo.

Los grupos de recarga se insertan en el orden de acuerdo con el campo INITIAL_LOAD_ORDER, de la tabla **TRIGGER_ROUTER**. La carga inicial de datos siempre es consultada desde la tabla de la base de datos origen y todos los datos pasan a través del enrutador, configurado para filtrar los datos que están destinados al nodo. Una manera más eficiente de cargar el subconjunto de datos es proveer una cláusula en el campo INITIAL_LOAD_ORDER, de la tabla **ROUTER**, que indica el orden de secuencia cuando una carga inicial es enviada a un nodo.

2.2.9 Replicación por niveles

Como se ha explicado anteriormente, existen organizaciones que tienen varios niveles de dirección, ubicados en distintos lugares geográficos. El continuo flujo de información que se genera debe ser replicado hacia todos los niveles, para mantener la misma información disponible y actualizada. Como se ilustra en la Figura 11: Niveles, una estructura de árbol, donde cada nivel requiere un subconjunto diferente de datos. Un nivel es representado por un grupo de nodos, y cada nodo en ese nivel, pertenece al grupo de nodos que lo representa.

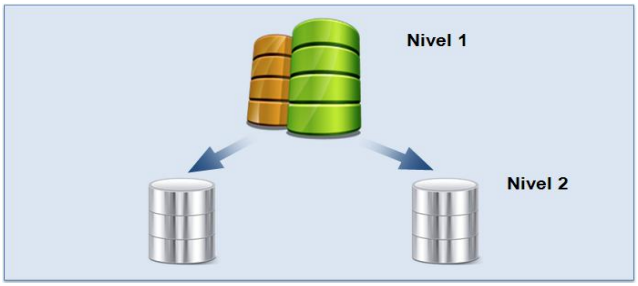

Figura 11: Niveles.

Un grupo de nodos, siempre va a enviar y recibir datos con otro grupo de nodos, de acuerdo con la configuración del enlace entre ellos. Por lo tanto, sólo se realizará la replicación siempre y cuando se encuentre especificado el otro nodo en la tabla **NODE** de la base de datos, y el campo **SYNC_ENABLED**, esté habilitado en **1**.

2.2.10 Redirección del registro

Al implementar un sistema de varios niveles, puede ser ventajoso tener un único servidor de registro, a pesar de que el padre de un nodo registrado puede ser cualquiera de los nodos del sistema. La tabla **REGISTRATION_REDIRECT**, permite un sólo nodo, por lo general el servidor raíz en la red, para redireccionar el registro de los nodos a sus verdaderos padres en el sistema de réplica. La tabla REGISTRATION_REDIRECT, cuenta con el campo REGISTRANT_EXTERNAL_ID, que es el id del grupo de nodos al que pertenece el nodo que desea registrarse, y el campo REGISTRATION_NODE_ID, que es el id del grupo a donde se va a redireccionar para que lo registre en el sistema.

2.2.11 Control de la replicación

La frecuencia de replicación de los datos se controla mediante la coordinación de una serie de eventos asincrónicos. Después que los datos son capturados, el primer evento que se produce es el enrutamiento. Para ello, los datos son extraídos por el canal, desde las tablas donde se produjeron los cambios hasta la tabla **DATA** de la herramienta. Esta operación es controlada por el campo EXTRACT_PERIOD_MILLIS, en la tabla **CHANNEL**. El canal siempre se encarga de guardar la última vez que se extrajeron datos en el campo LAST_EXTRACT_TIME, de la tabla **NODE_CHANNEL_CTL**. Toda la carga de datos se puede deshabilitar estableciendo la propiedad *dataloader.enabled* en false. Así, sólo se permitirá el envío, pero no el recibo de datos a ser replicados. Toda la extracción se puede deshabilitar estableciendo la propiedad

dataextractor.enabled en false. Estas propiedades se controlan mediante la tabla PARAMETER, del servidor raíz y afectan a todos los canales, exceptuando el canal “config”.

2.2.12 Instalación de la herramienta seleccionada

La Guía de instalación y configuración de PostgreSQL, para el uso de la herramienta SymmetricDS, se muestra en el Anexo 2, y la Guía de instalación y configuración de la herramienta SymmetricDS, se muestra en el Anexo 3.

Estructura de los principales directorios

En la Figura 12: Estructura de los principales directorios, se muestra la organización de los archivos y carpetas más significativos.

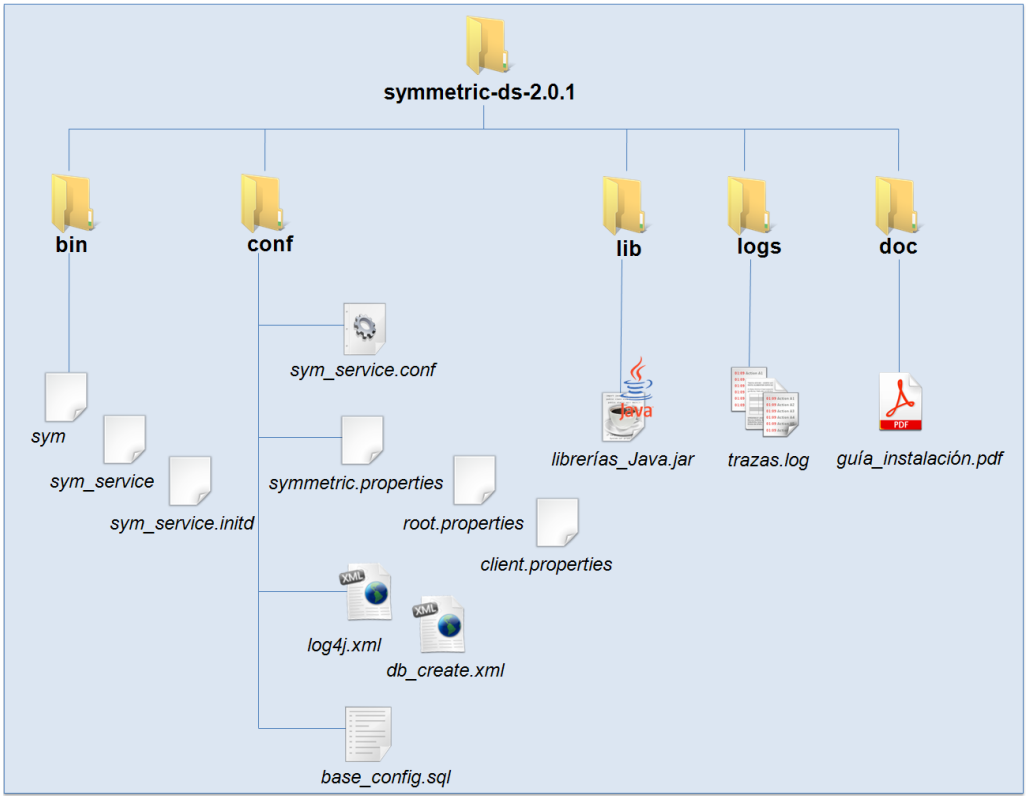

Figura 12: Estructura de los principales directorios.

La carpeta *symmetric-ds* está compuesta por las siguientes carpetas:

- *bin*: contiene los archivos ejecutables, el servicio para automatizar el despliegue de la herramienta, y la configuración del mismo. Dichos archivos son los siguientes:
    - *sym*: es la herramienta de líneas de comando que se utiliza para ejecutar la mayoría de las funcionalidades del sistema de réplica.
    - *sym_service.initd*: es donde se establece la configuración del archivo *sym_service.*
    - *sym_service*: se utiliza como el demonio que inicia y detiene el servidor web Jetty automáticamente, además de permitir el modo interactivo con el mismo.
- *conf*: contiene los siguientes archivos que sirven como plantilla para hacer las configuraciones de la herramienta.
    - *log4j.xml*: establece la configuración de los logs del sistema.
    - *sym_service.conf*: es donde se albergan los parámetros del servicio.
    - *symmetric.properties* es donde se establecen los parámetros generales de configuración.
    - *root.properties y client.properties*: establecen la configuración específica para un nodo raíz y un nodo cliente, respectivamente.
    - *db_create.xml*: es el script de creación de la base de datos.
    - *base_config.sql*: contiene la configuración base de la herramienta.
- *doc*: contiene la guía de instalación de la herramienta en formato pdf.
- *lib*: contiene archivos de extensión .jar, que son las librerías de Java necesarias para que la herramienta funcione.
- *logs*: en ella se guardan los archivos de trazas, de la instancia de la herramienta que se está ejecutando, que proporcionan información sobre todo lo ocurrido en el sistema.

Es válido aclarar que los archivos a los que se hace alusión anteriormente pueden variar el nombre, manteniendo la misma extensión.

Opciones de instalación

Se realizará la instalación de la herramienta como un servicio independiente con un servidor web Jetty[14] embebido. Es una manera simple y eficiente de efectuar la instalación, ya que el servidor Jetty aporta

[14] *Jetty: Es un servidor HTTP basado en Java y un contenedor de Servlets escrito en Java. Está publicado como un proyecto de software libre bajo la licencia Apache 2.0.*

escalabilidad y tiene un gran rendimiento. Además, la herramienta se estará ejecutando bajo un servicio de Linux, o demonio donde reside el servidor web Jetty.

- **Servicio independiente con un servidor web Jetty embebido**

El servicio independiente usa las opciones de la línea de comando *sym* para ejecutar el servidor web Jetty. Una instancia embebida del servidor web Jetty se utiliza para atender las solicitudes web de todos los servlets[15].

En el siguiente ejemplo, se inicia el servidor en el puerto 8080, con las propiedades de inicio que se encuentran en el archivo *root.properties*:

```
/symmetric/bin/sym --properties root.properties --port 8080 --server
```

- **Ejecutándose como un servicio de Linux**

La herramienta utiliza el software Java Service Wrapper[16], para ejecutarse en segundo plano como un servicio de Linux. El ejecutable del Java Service Wrapper es el *sym_service,* que puede ser fácilmente identificado desde una lista de procesos en ejecución. La configuración del servicio se encuentra en el archivo *sym_service.conf,* el cual se puede editar si se desea cambiar el número de puerto, que por defecto es 8080, la capacidad inicial de la memoria y el tamaño del archivo de registro de logs, entre otras configuraciones.

El script *sym_service.initd* facilita la configuración para trabajar con los niveles de Linux. Para instalarlo, se debe copiar el script para el directorio de inicio del sistema, o sea, para la carpeta *bin*, lo cual se puede lograr ejecutando:

```
cp bin/sym_service.initd /etc/init.d/sym_service
```

El servicio se puede ejecutar automáticamente cuando se inicia el sistema, esta opción puede ser habilitada o deshabilitada de la siguiente manera, respectivamente:

[15] *Servlet: Es un programa que se ejecuta en un servidor. Su uso más común es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.*

[16] *Java Service Wrapper: Es una herramienta que permite instalar los programas Java como servicios nativos de Windows o de Linux, permitiendo controlarlos con las herramientas del SO.*

```
/sbin/chkconfig --add sym_service
```

```
/sbin/chkconfig --del sym_service
```

Herramienta para la configuración inicial: "SymmetricBC"

Con esta herramienta se pretende facilitar el proceso de configuración a la hora de instalar la herramienta SymmetricDS.

Desde la **interfaz principal** (Interfaz Principal), se puede acceder a las siguientes funcionalidades:

- **Interfaz para la configuración de los grupos de nodos:**

Se especifica el id del grupo de nodos en el campo de texto Node Group Id y la descripción del mismo en el campo Description. En el botón New se crea un grupo nuevo, se actualiza en el botón Refresh y en el botón Save se guardan los cambios. (Interfaz para la configuración de los grupos de nodos)

- **Interfaz para la configuración del vínculo entre los grupos de nodos:**

Se especifica el id del nodo destino y el id del nodo origen en los campos Source Node Group Id y Target Node Group Id respectivamente. El tipo de vínculo en el campo Data Event Action, que puede tomar los valores: "P", para enviar, y "W" para recibir. En el botón New se crea un vínculo nuevo, se actualiza en el botón Refresh y en el botón Save se guardan los cambios. (Interfaz para la configuración del vínculo entre los grupos de nodos)

- **Interfaz para la configuración de un nodo:**

Se especifica el id del nodo en el campo Node Id. El id del grupo de nodos al que pertenece en el campo Node Group Id. El id externo en el campo External Id, y además, se puede habilitar o deshabilitar la replicación para el nodo en el campo Sync Enabled. En el botón New se crea un nodo nuevo, se actualiza en el botón Refresh y en el botón Save se guardan los cambios. (Interfaz para la configuración de un nodo)

- **Interfaz para la configuración de los canales de datos:**

Se especifica el id del canal en el campo Channel Id. El orden de secuencia para procesar los datos en el canal en el campo Processing Order. El número máximo de datos a enviar a través de un canal en el campo Max BatchSize y la descripción del canal en el campo Description. Se puede habilitar o deshabilitar

el canal en el campo Enabled. En el botón New se crea un canal nuevo, se actualiza en el botón Refresh y en el botón Save se guardan los cambios. (Interfaz para la configuración de los canales de datos)

- **Interfaz para la configuración de los disparadores**:

Se especifica el id del disparador en el campo Trigger Id. El nombre de la tabla a la que el disparador estará asociado para capturar los cambios de los datos, en el campo Source Table Name. El id del canal a través del cual fluyen los cambios de los datos, en el campo Channel Id. En los campos Sync On Insert, Sync On Update y Sync On Delete se indica si el disparador va a capturar o no los cambios de inserción, actualización, o eliminación de datos, respectivamente. En el campo Create Time se guarda el momento de creación del disparador y en el campo Last Update Time, el último momento en que se modificó. En el botón New se crea un disparador nuevo, se actualiza en el botón Refresh y en el botón Save se guardan los cambios. (Interfaz para la configuración de los disparadores)

- **Interfaz para la configuración de los enrutadores:**

Se especifica el id del enrutador en el campo Router Id. El id del nodo origen en el campo Source Node Group Id y el id del nodo destino en el campo Target Node Rout Id. En el campo Create Time se guarda el momento de creación del enrutador y en el campo Last Update Time, el último momento en que se modificó. En el botón New se crea un enrutador nuevo, se actualiza en el botón Refresh y en el botón Save se guardan los cambios. (Interfaz para la configuración de los enrutadores)

- **Interfaz para la configuración de la correlación entre disparadores y enrutadores:**

Se especifica en los campos Trigger Id y en Router Id, el id del disparador y del enrutador que se están asociando. En el campo Initial Load Order, se define el orden de secuencia de las tablas, cuando una carga inicial es enviada a un nodo. En el campo Create Time se guarda el momento de creación y en el campo Last Update Time, el último momento en que se modificó. En el botón New se crea una correlación nueva, se actualiza en el botón Refresh y en el botón Save se guardan los cambios. (Interfaz para la configuración de la correlación entre disparadores y enrutadores)

Seguridad del transporte

Al especificar el protocolo https para la dirección URL, la herramienta se comunicará a través del Protocolo de Capa de Conexión Segura SSL[17], para el transporte cifrado. Con https en la dirección URL, se deben ajustar las siguientes propiedades:

sync.url: es la dirección URL del nodo actual. Se debe especificar https en esta dirección, si se desea que otros nodos se comuniquen a través de SSL con él.

registration.url: es la dirección URL donde el nodo se va a conectar para registrarse cuando se inicie por primera vez. Se debe especificar https en esta dirección, para proteger el registro con SSL.

Para las conexiones https entrantes, la herramienta depende del servidor web en que esté desplegada, por lo que el servidor debe estar configurado para https. Mediante el comando sym se puede habilitar el soporte https.

- **Comando *sym***

Como se ha venido explicando, el comando sym, utiliza el Jetty como servidor web embebido. Usando las siguientes opciones de líneas de comando, el servidor web puede contar con los protocolos http, https o con ambos, respectivamente:

```
sym --port 8080
```

```
sym --secure-port 8443 --secure-server
```

```
sym --port 8080 --secure-port 8443 --mixed-server
```

Filtrado de direcciones IP

La herramienta puede filtrar las direcciones IP de los clientes que pueden conectarse a los servidores mediante diferentes formas, para ello utiliza los siguientes tipos de filtros:

- **Filtro CIDR (Classless Inter-Domain Routing)[18]**

[17] *SSL (Secure Sockets Layer): Es un protocolo criptográfico que proporciona comunicaciones seguras, autenticación y privacidad de la información.*

Es la definición el bloque de direcciones IP y bit significativos de la dirección que se va a comprobar. El filtro debe estar en un formato de direcciones IP, seguido por "/" y un valor numérico entre 0 y 32. Si se usa 0, indica que serán permitidas todas las direcciones IP y si se usa 32, indica que sólo será permitida una única dirección IP, para lo cual se puede utilizar también un Filtro Literal.

- **Filtro Literal**

Es la definición de una única dirección IP que será autorizada a conectarse al servidor. La cadena de la dirección IP debe estar completa y en un formato correcto.

- **Filtro de comodín**

Permite que todos los valores (de 0 a 255), para un elemento específico de una dirección IP sean válidos. Esto se denota con un " * " dentro de la parte específica de una dirección IP. El filtro de comodín se puede combinar con los demás, excepto con el Filtro CIDR.

- **Filtro de rangos**

Permite especificar un rango numérico dentro de un filtro de direcciones. El rango debe ser válido para una parte específica de la dirección IP.

2.3 Conclusiones

En el capítulo, se ha explicado detalladamente en términos de análisis y diseño cada uno de los aspectos necesarios para la concesión de la solución de réplica. En el análisis, fundamentalmente quedan definidos la serie de pasos lógicos, mientras que en el diseño, se explica cómo lograr la materialización de los mismos. La herramienta implementada, facilita en gran medida el proceso de realizar la configuración inicial en la instalación de la herramienta seleccionada. Se obtienen así las entradas para el próximo capítulo: Validación de la solución.

[18] *CIDR (Classless Inter-Domain Routing): Es la notación para restringir las conexiones de un nodo cliente a un nodo servidor. Es muy utilizado para el filtrado de direcciones IP.*

CAPÍTULO 3 VALIDACIÓN

Introducción

En el presente capítulo se valida la solución de réplica multi-maestra asincrónica, realizándole pruebas, enfocadas a los requerimientos identificados. Se disponen los entornos en los cuales van a ser ejecutadas las pruebas, para luego observar, comparar y evaluar los resultados obtenidos en cada entorno.

3.1 Realización de las pruebas

Las pruebas son el proceso de analizar un elemento de software para detectar diferencias entre las condiciones existentes y las requeridas. (21) O sea, el paso en el cual un sistema se ejecuta en circunstancias predefinidas, y los resultados que se obtienen son observados y registrados para realizar una evaluación. Los objetivos de las pruebas son demostrar que el sistema satisface sus requerimientos o descubrir defectos existentes en el mismo.

3.1.1 Entornos de Prueba

Los recursos de software y de hardware a utilizar para realizar las pruebas son los siguientes:

Nodo	Procesador (CPU)	Memoria (RAM)	Sistema Operativo
Raíz	Intel (R) Pentium (R) 4. Velocidad de CPU 2.4 GHz.	512 MB	Ubuntu (Linux)
Clientes	Intel (R) Pentium (R) 4. Velocidad de CPU 2.4 GHz.	512 MB	Ubuntu (Linux)

Tabla 2 Recursos de hardware y de software

Las pruebas serán ejecutadas usando los siguientes entornos:

Entorno Negocio de Tiendas Online

Se cuenta con una base de datos con una parte representativa de los tipos de datos de PostgresSQL 8.4. Los datos serán replicados desde cada uno de los dos niveles. Los niveles estarán asilados sin la introducción de errores ni sobrecargas de rendimiento.

Se trata de una tienda que tiene una sucursal ubicada en otro municipio. La base de datos está compuesta por 5 tablas:

item	sale_return_line_item	sale_transaction
ítem_selling_price	sale_tender_line_item	

Las dos primeras tablas se encuentran configuradas para replicarse desde el nodo raíz al nodo cliente y las demás, desde el nodo cliente al nodo raíz.

Entorno Empresa de Software

La solución de réplica manipula una base de datos con algunos tipos de datos de PostgreSQL 8.4. Los procesos de replicación operan con grandes volúmenes de datos y en cada nivel se introducen fallos.

Se trata de una empresa de software, que tiene una oficina en Ciudad de la Habana y una oficina en Granma. El equipo de desarrollo está compuesto por trabajadores y a su vez tiene asignados varios o ningún proyecto. Los trabajadores pueden ser jefe de proyecto o programador, estos últimos tienen una categoría, y dominan uno o varios lenguajes de programación. Los proyectos pueden ser de gestión o de multimedia. La base de datos está compuesta por 10 tablas:

dequipodesarrollo	ncategoria	dproyecto
dtrabajador	rprogramadorlenguaje	dproyectogestion
djefeproyecto	nlenguajeprogramacion	dprotyectomultimedia
dprogramador		

Los procesos de negocio en este entorno funcionan de la siguiente manera:

El centro que se encuentra en La Habana es el encargado de contactar con los clientes y crear los proyectos. El centro ubicado en Granma es el encargado de llevar a cabo la gestión de los recursos humanos, así como la asignación de los equipos de desarrollo a los proyectos. Por lo tanto, desde el grupo de nodos “habana”, se va a replicar la información de las tablas dproyecto, dproyectogestion y dproyectomultimedia. Mientras que las demás se van a replicar desde el grupo de nodos “granma”.

3.1.2 Tipos de pruebas

Las pruebas serán enfocadas atendiendo a los requerimientos funcionales. A continuación se especifica cada uno de los tipos de pruebas a efectuar.

Pruebas Funcionales

Las pruebas funcionales están orientadas a detectar si el sistema cumple adecuadamente con las funcionalidades descritas en los requerimientos. Se realizan pruebas funcionales para:

RF 1 Soportar la implantación de la réplica de datos dividida por niveles.

Pruebas de Rendimiento

Las pruebas de rendimiento determinan si los tiempos de respuesta del sistema, tanto en condiciones normales como en condiciones especiales, se encuentran dentro de los límites predefinidos. Se realizan pruebas de rendimiento para:

RF 2 Garantizar la disponibilidad de los datos.

RF 5 Garantizar que el sistema funcione en entornos que pueden estar durante cierto período de tiempo desconectados.

Pruebas de Disponibilidad

Las pruebas de disponibilidad consisten en probar la disponibilidad del sistema ante fallos de diferentes componentes de la arquitectura, físicos o lógicos. Se realizan pruebas de disponibilidad para:

RF 3 Garantizar la tolerancia a fallos.

Pruebas de Seguridad

Las pruebas de seguridad intentan verificar que los mecanismos de protección del sistema lo protegerán de accesos impropios. Se trata de violar toda protección del sistema, por ejemplo, probando aspectos relativos a la confidencialidad o integridad de la información. Se realizan pruebas de seguridad para:

RF 8 Mantener la confidencialidad de la información.

3.1.3 Pruebas

Las pruebas fueron ejecutadas en cada uno de los entornos anteriormente descritos. Para ambos entornos las mismas arrojaron resultados satisfactorios. A continuación se representa cada una de las pruebas efectuadas, describiendo las condiciones preparadas para su ejecución, la entrada, o sea, la acción que se realizó sobre el sistema de réplica y el resultado obtenido.

Soporte de varios niveles.

Objetivo: Insertar datos en cada uno de los nodos ubicados en lugares separados para comprobar que la replicación se realiza correctamente a través de varios niveles.

Prueba 1	Soporte de varios niveles.
Condiciones:	Se disponen los dos niveles, con una instancia de la solución de réplica. Cada instancia ejecutándose en nodos distintos y en subredes separadas.
Entrada:	Se insertaron datos en varias tablas donde se espera que se repliquen hacia el otro nivel. De igual manera se hizo el proceso para tablas pertenecientes al otro nodo. Luego de haber insertado todos estos datos se realizó la replicación.
Resultado:	Los datos insertados fueron replicados con éxito en ambos nodos.

Tabla 3 Prueba de soporte de niveles.

Se comprobó que el sistema de réplica funciona correctamente al ser instalado en varios niveles, ubicados en distintos nodos y con subredes separadas. Se evidenció la réplica multi-maestra, ya que ambos servidores se comportan como maestros, al poder realizar cambios en la base de datos.

Disponibilidad de los datos

Objetivo: Simular un gran número de usuarios conectados concurrentemente. Detener el servicio de replicación e insertar un gran volumen de datos en uno de los nodos. Luego se inicia el servicio de replicación con vista a probar los tiempos de respuesta de las peticiones hechas por los usuarios al mismo tiempo que el sistema de replicación se encuentra en funcionamiento.

Prueba 2	Disponibilidad de los datos.

Condiciones:	Con el servicio de replicación detenido, se tienen 30 conexiones simultáneas al servidor de base de datos ejecutando consultas, se realizó una consulta que arrojó 500 tuplas en un tiempo de 78 Ms.
Entrada:	Se insertan 1000 tuplas, después se restablece el servicio y se realiza la consulta, con la misma cantidad de usuarios conectados.
Resultado:	El tiempo de ejecución de la consulta realizada en el momento de replicación inicializado fue de 82 Ms.

Tabla 4 Prueba de disponibilidad de los datos

La misma operación se realizó nuevamente, insertando 3000 y 5000 tuplas. El siguiente gráfico muestra el comportamiento del tiempo de ejecución de la consulta antes y después de ser insertadas las tuplas, en el entorno “Negocio de Tiendas Online”.

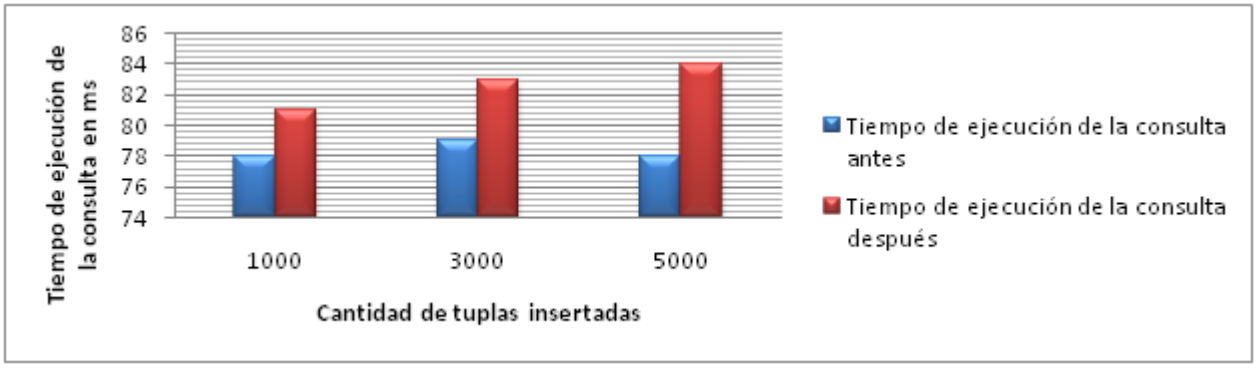

Gráfico 1 Prueba de disponibilidad

Los tiempos de ejecución de la consulta no tuvieron una diferencia significativamente grande al aumentar la cantidad de tuplas insertadas. Se comprobó que el sistema de réplica no afecta en gran medida los tiempos de respuesta del servidor donde se encuentra funcionando.

Tolerancia a Fallos

Objetivo: Provocar intencionalmente fallos en la conexión entre los nodos.

Prueba 3	Tolerancia a Fallos.
Condiciones:	Se cuenta con el sistema de réplica en medio del proceso de replicación enviando una carga de 3000 tuplas desde el nodo raíz al nodo cliente.
Entrada:	En medio del proceso de replicación se detuvo el servidor cliente y luego de

	10 minutos se inició.
Resultado:	Al encender el servidor cliente los datos fueron recibidos en sus tablas correspondientes en el mismo orden en que fueron enviados.

Tabla 5 Prueba de tolerancia a fallos

El sistema soporta que puedan existir fallos, como por ejemplo la pérdida de conexión en el momento de la replicación por parte de alguno de los nodos, manteniendo el correcto comportamiento.

Entornos desconectados

Objetivo: Quitar la conexión en los nodos que forman parte del sistema de réplica durante un tiempo prudencial, realizar modificaciones a los datos en uno de ellos y luego restablecer la conexión.

Prueba 4	**Entornos desconectados.**
Condiciones:	Se tiene el sistema de réplica, con los nodos raíz y cliente en correcto funcionamiento pero sin conexión uno con el otro.
Entrada:	Se insertaron 1000 filas en el nodo cliente, de estas se eliminaron 1000 y se modificaron 20, al cabo de 15 minutos se restableció la conexión.
Resultado:	Al restablecer la conexión, los datos que se introdujeron en el nodo cliente fueron replicados con éxito en el nodo raíz.

Tabla 6 Prueba de entornos desconectados

Se comprobó que el sistema de réplica funciona en entornos donde la red es inestable, sin alterar su funcionamiento, ya que al pasar un período de tiempo sin conexión, el mismo es capaz de restablecer la replicación de los datos.

Confidencialidad

Objetivo: Tratar de realizar conexiones y recibir datos desde un nodo cliente sin tener abierto el registro.

Prueba 5	**Confidencialidad.**
Condiciones:	Se tiene un nuevo nodo cliente, con una instancia de la solución de réplica instalada, el cual realizó una petición de registro. El administrador en el nodo raíz no ha abierto el registro para ese nodo cliente.
Entrada:	Se intentó conectar desde el nodo cliente al nodo raíz para recibir la carga

	inicial de los datos, enviar y recibir datos y manipular la base de datos.
Resultado:	No se logró acceder a la base de datos y al realizar la carga inicial de los datos se produjo un error.

Tabla 7 Prueba de confidencialidad

El sistema de réplica garantiza la seguridad y confidencialidad de los datos. Sólo tendrán acceso a la información los nodos que tengan abierto el registro por el administrador del nodo raíz.

Soporte de tipos de datos

Objetivo: Manipular información con tipos de datos de PostgreSQL para comprobar que el sistema los reconoce.

Prueba 6	**Soporte de tipos de datos.**
Condiciones:	La base de datos cuenta con tuplas que poseen los tipos de datos: integer, character, boolean, date, double y bytea, que representan los tipos de datos de PostgreSQL.
Entrada:	Se insertaron en el nodo raíz juegos de datos de pequeños y medianos volúmenes, con los tipos de datos integer, character, boolean, date, double y bytea. Posteriormente se inició la replicación.
Resultado:	El sistema admitió los datos insertados y los replicó hacia el nodo cliente.

Tabla 8 Prueba de soporte de tipos de datos.

Se comprobó que el sistema de réplica tiene soporte para los tipos de datos del gestor de bases de datos utilizado, PostgreSQL.

3.2 Conclusiones

Las pruebas efectuadas arrojaron resultados satisfactorios, por lo que se concluye que la solución de réplica multi-maestra asincrónica cumple con los requerimientos necesarios. Garantizando así la integridad, disponibilidad y seguridad de los datos, la efectividad y la tolerancia a fallos. La efectividad, es directamente proporcional a la disponibilidad de datos a la hora de ejecutar procesos paralelos. Con una alta disponibilidad se logra que los procesos sean atendidos en una razón de tiempo razonable, a pesar de fallos que puedan ocurrir en el servidor. Esta tolerancia a fallos permitirá que pueda existir un cierto número de fallos manteniendo el correcto comportamiento del sistema.

CONCLUSIONES GENERALES

En el trabajo se ha dado cumplimiento al objetivo general y a cada uno de los objetivos específicos planteados al inicio, para ello:

Se realizó un estudio detallado sobre los aspectos relacionados con las bases de datos y la replicación de datos, que aportó una serie de definiciones y conceptos importantes a tener en cuenta a la hora de llevar a cabo este proceso. Se efectuó una búsqueda exhaustiva de las herramientas de réplica multi-maestra asincrónica existentes a nivel mundial, seleccionando la más adecuada para la solución y fundamentando su elección.

Como parte del análisis:

- Se identificaron los requerimientos necesarios, dejando claras todas las funcionalidades que debe cumplir la solución.
- Se determinaron los pasos a seguir para la implantación de la solución y se aclararon los principales conceptos que deben ser dominados.

Posteriormente en el diseño:

- Se definieron los términos de configuración de la herramienta seleccionada, adaptado a los pasos seguidos en el análisis.
- Se mostró el modelo de datos de la herramienta seleccionada, con las tablas más importantes que serán usadas.
- Se presentaron las opciones de instalación de la herramienta.
- Se implementó una herramienta de apoyo para realizar la configuración inicial, que facilita en gran medida este proceso a la hora de la instalación.

Se validó la solución realizando pruebas en varios entornos, enfocadas a los requerimientos identificados en el análisis. Las pruebas arrojaron resultados satisfactorios, por lo que se concluye que la solución de réplica multi-maestra asincrónica propuesta garantiza la efectividad, la alta disponibilidad y la tolerancia a fallos.

RECOMENDACIONES

Se recomienda:

- Profundizar en los temas de configuración de la herramienta SymmetricDS para la fragmentación de datos horizontal y vertical, así lograr perfeccionar la propuesta.
- Ampliar la herramienta implementada para permitir guardar en forma de script SQL la configuración inicial que se establece en el archivo *base_config.sql.*
- Aplicar la solución propuesta en futuros proyectos que asuma el Centro de Tecnologías de Gestión Datos.

REFERENCIAS BIBLIOGRÁFICAS

1. Sitio oficial del Ministerio de la Informática y las Comunicaciones de Cuba (MIC). [Online] [Cited: enero 29, 2010.] http://www.mic.gov.cu/.

2. **Valenzuela Ruz, Victor.** Diseño de Bases de Datos. [Online] [Cited: enero 31, 2010.] http://www.scribd.com/doc/20710988/bases-de-datos.

3. *Réplica entre servidores Oracle, alternativas de solución.* **Piñeiro, DrC. Pedro Yobanis.** Ciudad de la Habana : s.n., 2006.

4. **Meyer, Beltran.** *Construcción de Software OO.*

5. **Date, C. J.** *Introduction to Database Systems.* s.l. : Addison Wesley, 2003.

6. **Carballal, F.M.** *Bases de Datos Distribuídas. .* 2007.

7. **Mato Garcia, Rosa Maria.** *Diseño de Bases de Datos.* La Habana : s.n., 1999.

8. Sitio Oficial de SQLServer. [Online] http://www.microsoft.com/spain/sql/2008/default.aspx.

9. **Oracle.** *Oracle Documentation 11g.*

10. Sitio oficial de PostgreSQL. [Online] http://www.postgresql.org/docs/faq/.

11. **Buretta, M.** *Data Replication Tools and Techniques for managing distributed information.* s.l. : Wiley, 1997.

12. **Fraire Huacuja, Héctor Joaquín.** *Automatización del Diseño de la Fragmentación Horizontal en Bases de Datos Distribuidas.*

13. **Steffen, Ing. Hermann.** Técnicas Avanzadas para Gestión de Sistemas de Información.

14. **Reingart, Mariano.** PyReplica Sistema de replicación simple para PostgreSQL programado en Python.

15. **Long, Eric and Henson, Chris.** SymmetricDS User Guide Version 1.0.

16. Bucardo.org. [Online] http://bucardo.org/.

17. *Réplica bidireccional basada en control de cambios.* **Landrian García, Jorge.** La Habana : s.n.

18. Portal Software Libre Universidad de las Ciencias Informáticas. [Online] http://softwarelibre.uci.cu/.

19. Lenguajes de programación. [Online] http://www.lenguajes-de-programacion.com/programacion-java.shtml.

20. Guia Documentada Para UBUNTU. [Online] http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

21. **IEE.** IEEE Standard Glossary of Software Engineering Terminology.

BIBLIOGRAFÍA

1. Sitio oficial del Ministerio de la Informática y las Comunicaciones de Cuba (MIC). [Online] [Cited: enero 29, 2010.] http://www.mic.gov.cu/.

2. **Valenzuela Ruz, Victor.** Diseño de Bases de Datos. [Online] [Cited: enero 31, 2010.] http://www.scribd.com/doc/20710988/bases-de-datos.

3. *Réplica entre servidores Oracle, alternativas de solución.* **Piñeiro, DrC. Pedro Yobanis.** Ciudad de la Habana : s.n., 2006.

4. **Meyer, Beltran.** *Construcción de Software OO.*

5. **Date, C. J.** *Introduction to Database Systems.* s.l. : Addison Wesley, 2003.

6. **Carballal, F.M.** *Bases de Datos Distribuídas. .* 2007.

7. **Mato Garcia, Rosa Maria.** *Diseño de Bases de Datos.* La Habana : s.n., 1999.

8. Sitio Oficial de SQLServer. [Online] http://www.microsoft.com/spain/sql/2008/default.aspx.

9. **Oracle.** *Oracle Documentation 11g.*

10. Sitio oficial de PostgreSQL. [Online] http://www.postgresql.org/docs/faq/.

11. **Buretta, M.** *Data Replication Tools and Techniques for managing distributed information.* s.l. : Wiley, 1997.

12. **Fraire Huacuja, Héctor Joaquín.** *Automatización del Diseño de la Fragmentación Horizontal en Bases de Datos Distribuidas.*

13. **Steffen, Ing. Hermann.** Técnicas Avanzadas para Gestión de Sistemas de Información.

14. **Reingart, Mariano.** PyReplica Sistema de replicación simple para PostgreSQL programado en Python.

15. **Long, Eric and Henson, Chris.** SymmetricDS User Guide Version 1.0.

16. Bucardo.org. [Online] http://bucardo.org/.

17. *Réplica bidireccional basada en control de cambios.* **Landrian García, Jorge.** La Habana : s.n.

18. Portal Software Libre Universidad de las Ciencias Informáticas. [Online] http://softwarelibre.uci.cu/.

19. Lenguajes de programación. [Online] http://www.lenguajes-de-programacion.com/programacion-java.shtml.

20. Guia Documentada Para UBUNTU. [Online] http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

21. **IEE.** IEEE Standard Glossary of Software Engineering Terminology.

22. **Elmasri, R. and Navathe, S.B.** *Fundamentals of database systems.* s.l. : Addison-Wesley, 2000.

23. Microsoft/Technet. [Online] http://technet.microsoft.com/es-es/library/ms152567.aspx.

24. **Jacobson, I, Booch, G and Rumbaugh, J.** *El proceso unificado de desarrollo de software.* s.l. : Addison Wesley, 2000.

25. **Española, Real Academia.** *Diccionario de la Real Academia Española (DRAE) .*

26. *Replicación de Datos en SQL Server.* **Castro Morell, Daniel Eduardo.** Santa Clara : s.n.

27. **Long, Eric, Henson, Chris and Hanes, Mark.** SymmetricDS 2 User Guide. 2007-2010.

28. *Tipos de datos relevantes en PostgreSQL.* [Online] http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/.

29. **Cristian.** Java, Linux y Programación. [Online] http://casidiablo.net/correr-programa-java-como-demonio/.

30. www.javahispano.org: Tu lenguaje, tu comunidad. [Online] www.javahispano.org.

31. The Source for Java Developers. [Online] http://java.sun.com.

32. RecorteX. [Online] http://www.recortex.com/.

33. **Castillo Contreras, Gabriel Fernando.** *SISTEMAS DE BASES DE DATOS DE ALTA DISPONIBILIDAD.* Guatemala : s.n., 2006.

34. **Dieguez Sánchez, Roberto Carlos and Fernández Banguela, Sergio.** Sistema de sincronización y gestión de nodos aislados para la Replicación de bases de datos en PostgresSQL. La Habana : s.n., 2008.

35. **Fajardo Vega, Norge.** Sistema de réplica para bases de datos distribuidas en PostgreSQL. La Habana : s.n., 2007.

ANEXOS

Anexo 1. Tipos de datos estándar en PostgreSQL

Tipo en PostgreSQL	Correspondiente en SQL3	Descripción
bool	boolean	valor lógico o booleano (true/false)
char(n)	character(n)	cadena de caracteres de tamaño fijo
date	date	fecha (sin hora)
float4/8	float	número de punto flotante con precisión
float8	real, double precision	número de punto flotante de doble precisión
int2	smallint	entero de dos bytes con signo
int4	int, integer	entero de cuatro bytes con signo
int4	decimal	número exacto
int4	numeric	número exacto
money	decimal	cantidad monetaria
time	time	hora en horas, minutos, segundos y centésimas
timespan	interval	intervalo de tiempo
timestamp	timestamp with time zone	fecha y hora con zonificación
varchar(n)	character varying(n)	cadena de caracteres de tamaño variable

Tabla 9 Tipos de datos de estándar SQL3 en PostgreSQL

Anexo 2. Guía de instalación y configuración de PostgreSQL, para el uso de la herramienta SymmetricDS

- Instalar PostgreSQL 8.4.
- Cambiar la contraseña del usuario postgres.
    - Loguearse con el usuario postgres.

```
symmetric@symmetric-desktop:~$ sudo su
[sudo] password for symmetric:
root@symmetric-desktop:/home/symmetric# su postgres
postgres@symmetric-desktop:/home/symmetric$
```

- Entrar en la consola de postgres y ejecutar la consulta: alter user postgres with password '***'.

```
postgres@symmetric-desktop:/home/symmetric$ psql
psql (8.4.2)
Digite «help» para obtener ayuda.

postgres=# alter user postgres with password 'asd';
ALTER ROLE
postgres=#
```

- Salir de la consola de postgres (ctrl+z) y salir del usuario postgres (exit).

- Editar el fichero pg_hba.conf:
  - Agregar el IP y el puerto para que todas las direcciones IP se puedan conectar.

```
# IPv4 local connections:
host    all         all         127.0.0.1/32          md5
host    all         all         10.0.0.0/8            md5
```

  - Reiniciar el servicio de postgres.

```
root@symmetric-desktop:/home/symmetric# /etc/init.d/postgresql-8.4 stop
 * Stopping PostgreSQL 8.4 database server
root@symmetric-desktop:/home/symmetric# /etc/init.d/postgresql-8.4 start
 * Starting PostgreSQL 8.4 database server
root@symmetric-desktop:/home/symmetric#
```

- Editar el fichero postgresql.conf.
  - Establecer en listen_addresses = ' * ', para que escuche todas las direcciones.

```
#---------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#---------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
```

  - Establecer la lista de nombres de clase de la variable personalizada, en custom_variable_classes = 'symmetric'.

```
#--------------------------------------------------------------------------
# CUSTOMIZED OPTIONS
#--------------------------------------------------------------------------

custom_variable_classes = 'symmetric'     # list of custom variable class names
```

- Reiniciar el servicio de postgres.

Anexo 3. Guía de instalación y configuración de la herramienta SymmetricDS

- Editar con el usuario con que se va a instalar la herramienta, el archivo de extensión .properties para la configuración de los nodos raíz o cliente, los mismos se encuentran dentro de la carpeta conf. Editar las siguientes propiedades en cada uno de ellos:
  - El nombre de la clase para el driver JDBC, que es el driver que utiliza Java para conectarse a la base de datos, seleccionando db.driver=org.postgresql.Driver.

```
# The class name for the JDBC Driver
#db.driver=com.mysql.jdbc.Driver
#db.driver=oracle.jdbc.driver.OracleDriver
db.driver=org.postgresql.Driver
#db.driver=org.apache.derby.jdbc.EmbeddedDriver
#db.driver=org.hsqldb.jdbcDriver
#db.driver=net.sourceforge.jtds.jdbc.Driver
#db.driver=com.ibm.db2.jcc.DB2Driver
#db.driver=org.h2.Driver
```

  - La dirección URL que usará para conectarse a la base de datos, de la siguiente forma: db.url= jdbc:postgresql://nombre o IP del servidor de base de datos: puerto/nombre de la base de datos.

```
# The JDBC URL used to connect to the database
#db.url=jdbc:mysql://localhost/sampleroot
#db.url=jdbc:oracle:thin:@127.0.0.1:1521:sampleroot
db.url=jdbc:postgresql://localhost/bd
#db.url=jdbc:derby:sampleroot;create=true
#db.url=jdbc:hsqldb:file:sampleroot;shutdown=true
#db.url=jdbc:h2:file:sampleroot
#db.url=jdbc:jtds:sqlserver://localhost:2299/sampleroot
#db.url=jdbc:db2://localhost/samproot
#db.url=jdbc:h2:sampleroot;AUTO_SERVER=TRUE;LOCK_TIMEOUT=60000
```

  - El usuario y la contraseña con que se va a acceder a la base de datos.

```
# The user to login as who can create and update tables
db.user=postgres
```

```
# The password for the user to login as
db.password=postgres
```

- Establecer en el nodo raíz la siguiente propiedad.

```
sync.url=http://localhost:8080/sync
```

- Establecer en el nodo cliente la siguiente propiedad.

```
# The HTTP URL of the root node to contact for registration
registration.url=http://localhost:8080/sync
```

- **Crear la base de datos, lenguaje y funciones**

Previamente, deben estar creadas las bases de datos, usando el cliente para la administración del sistema gestor de base de datos PostgreSQL 8.4, PgAdmin III.

- En cada BD, se debe ejecutar la siguiente consulta para agregar el lenguaje plpgsql y las funciones plpgsql_call_handler y plpgsql_validator, que utilizará SymmetricDS.

```
CREATE FUNCTION plpgsql_call_handler() RETURNS language_handler AS
'$libdir/plpgsql' LANGUAGE C;
CREATE FUNCTION plpgsql_validator(oid) RETURNS void AS
'$libdir/plpgsql' LANGUAGE C;
CREATE TRUSTED PROCEDURAL LANGUAGE plpgsql
HANDLER plpgsql_call_handler
VALIDATOR plpgsql_validator;
```

- Crear las tablas de la base de datos, en el nodo raíz:

```
../bin/sym -p root.properties --run-ddl nombre del script de la BD.xml
```

El script puede ser con extensión xml o sql, en caso de ser sql se pude utilizar PgAdmin.

- Crear las tablas propias de la herramienta en la base de datos del nodo raíz:

```
../bin/sym -p root.properties --auto-create
```

- Cargar los datos y la configuración de las tablas en la base de datos del nodo raíz:

```
../bin/sym -p root.properties --run-sql nombre del script de configuración de la
                                base de datos.sql
```

En caso de estar en presencia de un modelo de base de datos de gran tamaño utilizar la herramienta **SymmetricBC**.

- Crear las tablas de la base de datos en el nodo cliente:

```
../bin/sym -p client.properties --run-ddl nombre del script de la BD.xml
```

Al igual que en el nodo raíz pueden utilizase cualquiera de las dos variantes para crear las tablas.

- **Iniciar la herramienta:**
  - Iniciar el nodo raíz

```
../bin/sym -p root.properties --port 8080 –server
```

  - Iniciar el nodo cliente:

```
../bin/sym -p client.properties --port 9090 –server
```

  - Abrir el registro para el nodo cliente:

```
../bin/sym -p root.properties --open-registration “grupo de nodos al que
                pertenece el nodo,id externo del nodo"
```

- Enviar la carga inicial de datos:

```
../bin/sym -p root.properties --reload-node id externo del nodo
```

Anexo 4. Herramienta para la configuración inicial: "SymmetricBC"

Anexo 4.1 Interfaz Principal

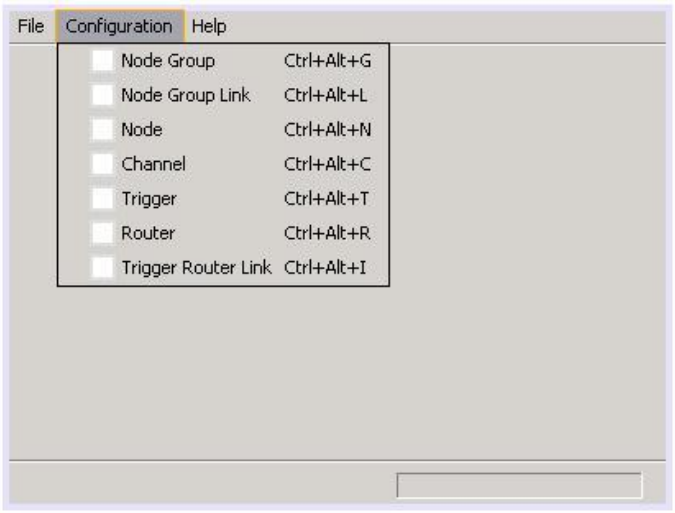

Figura 13: Interfaz principal, opciones de configuración.

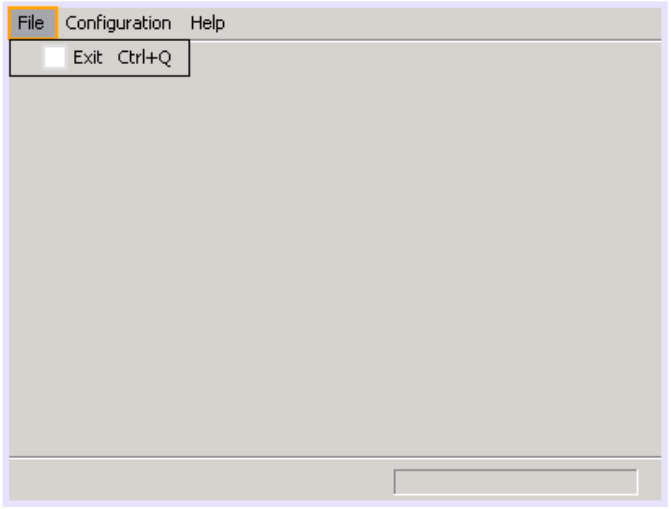

Figura 14: Interfaz principal, opción salir de la herramienta.

Anexo 4.2 Interfaz para la configuración de los grupos de nodos

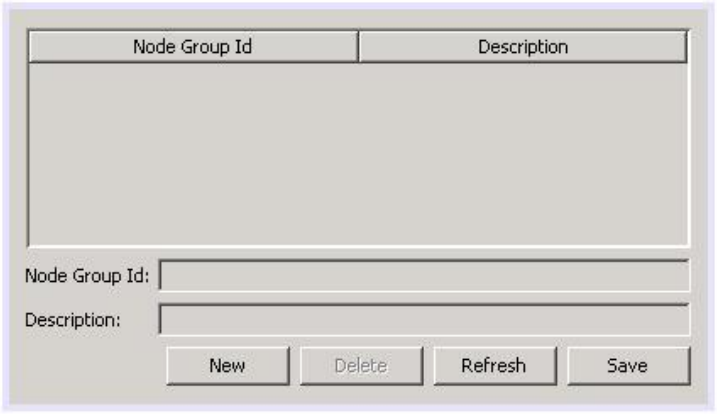

Figura 15: Interfaz para la configuración de los grupos de nodos.

Anexo 4.3 Interfaz para la configuración del vínculo entre los grupos de nodos

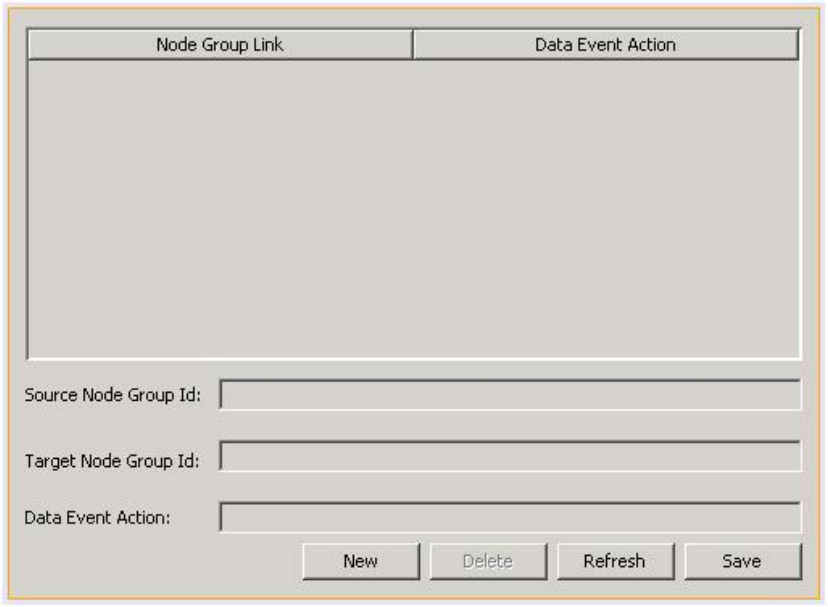

Figura 16: Interfaz para la configuración del vínculo entre los grupos de nodos.

Anexo 4.4 Interfaz para la configuración de un nodo

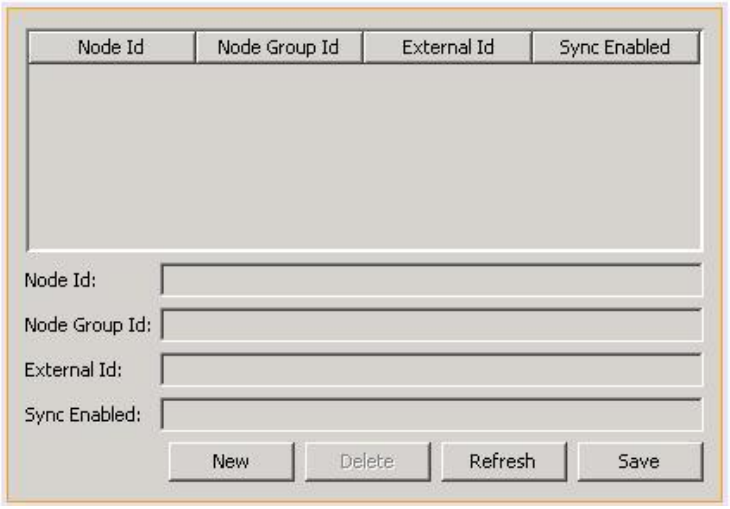

Figura 17: Interfaz para la configuración de un nodo.

Anexo 4.5 Interfaz para la configuración de los canales de datos

Figura 18: Interfaz para la configuración de los canales de datos.

Anexo 4.6 Interfaz para la configuración de los disparadores

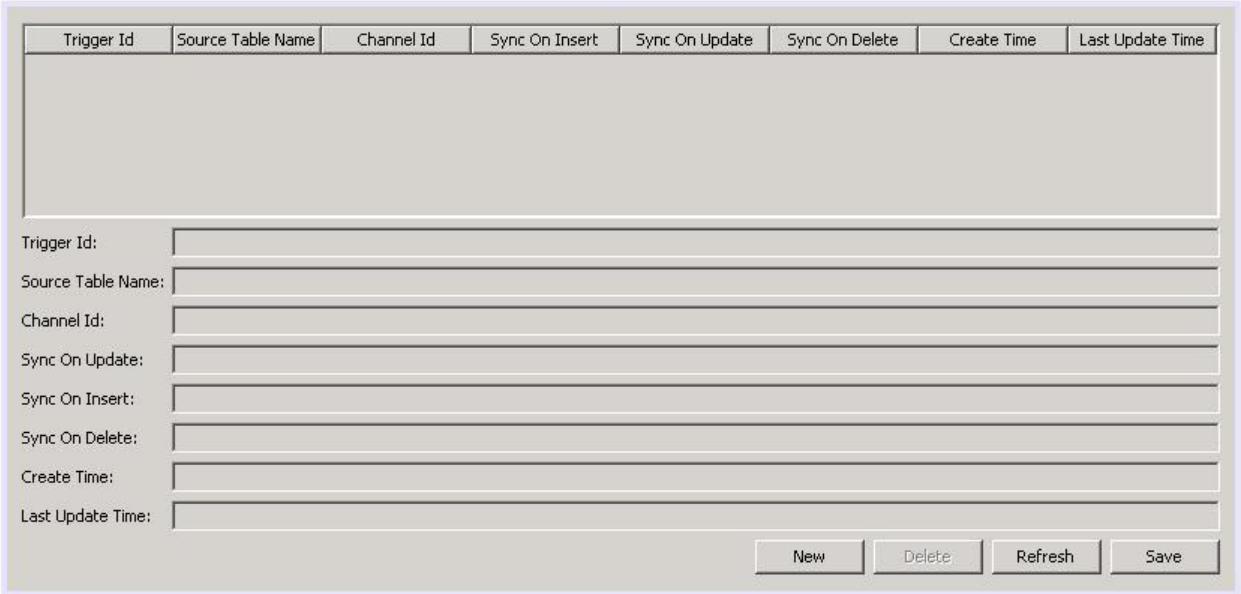

Figura 19: Interfaz para la configuración de los disparadores.

Anexo 4.7 Interfaz para la configuración de los enrutadores

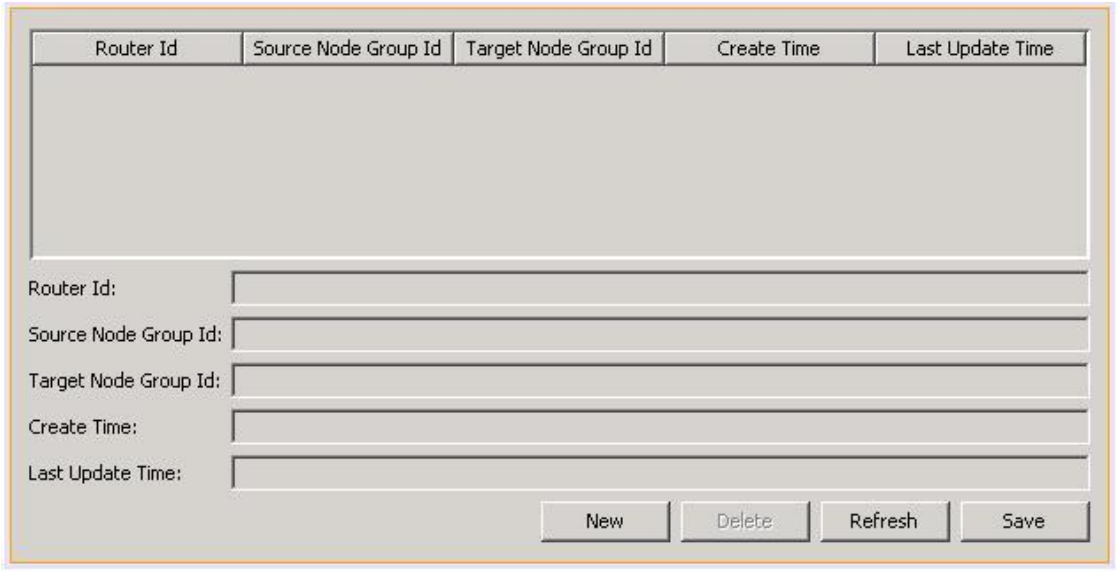

Figura 20: Interfaz para la configuración de los enrutadores.

Anexo 4.8 Interfaz para la configuración de la correlación entre disparadores y enrutadores

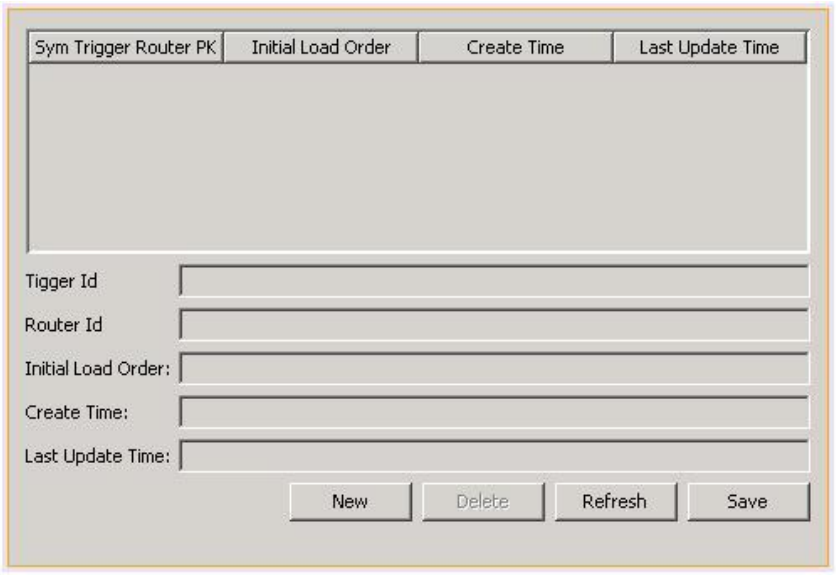

Figura 21: Interfaz para la configuración de la correlación entre disparadores y enrutadores.

GLOSARIO DE TÉRMINOS

Ancho de banda: Cantidad de información o de datos que se puede enviar a través de una conexión de red en un período de tiempo dado. Se puede indicar en bites por segundo (BPS), kilobites por segundo (kbps), o megabites por segundo (mps).

Balanceo de carga: Técnica usada para compartir el trabajo a realizar entre varios procesos, ordenadores, discos u otros recursos. Se mantiene gracias a un algoritmo que divide de la manera más equitativa posible el trabajo, para evitar los así denominados cuellos de botella.

Bases de datos: Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Confidencialidad: Hacer que la información sea ininteligible para aquellos individuos que no estén involucrados en la operación, o sea, asegurar que sólo los usuarios autorizados tengan acceso a los recursos que se intercambian.

Demonio (Daemon): Proceso informático que se ejecuta en segundo plano.

Disponibilidad: Garantizar el acceso a un servicio o a los recursos, o sea, avalar el correcto funcionamiento de los sistemas de información.

DCL: Lenguaje de Control de Datos (Data Control Languaje), usado para controlar los permisos de los objetos de la base de datos.

DDL: Lenguaje de Definición de Datos (Data Definition Language), usado para crear objetos en la base de datos.

DML: Lenguaje de Manipulación de datos (Data Manipulation Languaje), usado para consultar y modificar los datos.

Enrutamiento: Función de buscar un camino entre todos los posibles en una red de paquetes. Permite determinar la ruta que debe tomar el paquete de datos.

Escalabilidad: Propiedad deseable de un sistema, que indica su habilidad para extender el margen de operaciones sin perder calidad y manejar el crecimiento continuo de trabajo de manera fluida.

Hardware: Conjunto de los componentes que integran la parte material de una computadora.

Industria del software: Industria que involucra la investigación, desarrollo, comercialización y distribución de software.

Integridad: Determinar si se han alterado los datos durante la transmisión (accidental o intencionalmente), o sea, garantizar que los datos sean los que se supone que son.

Log: Registro oficial de eventos durante un rango de tiempo en particular.

Medida de latencia: Tiempo que una réplica puede estar inconsistente hasta llegar a estar consistente con la fuente primaria designada

Multiplataforma: Software que puede ejecutarse en varias plataformas.

Período de inoperatividad de la red: Tiempo en que el servicio de red no está disponible.

Polling: Operación de consulta constante.

Servidor web: programa que está diseñado para transferir hipertextos, páginas web o páginas HTML.

Sistema gestor de bases de datos: Sistema dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Software: Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

Software libre: Libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

SQL: Lenguaje estructurado de consulta, está compuesto por comandos, cláusulas, operadores y funciones, que se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

Técnicas de replicación: Transporte de datos entre dos o más instancias de servidores.

Tecnologías de la información y las comunicaciones (TIC): Tecnologías que agrupan los elementos y las técnicas utilizadas en el tratamiento y la transmisión de las informaciones, principalmente de informática, Internet y telecomunicaciones.

Tupla: Conjunto de elementos de distinto tipo que se guardan de forma consecutiva en memoria.