ISSN 2071-081X
ISSN 2411-0035

REPLICACIÓN ASÍNCRONA DE BASE DE DATOS

Mijael Colquehuanca Javieri[1]

Instituto de Investigaciones en Ciencia y Tecnología,
Universidad La Salle
mijacolque@gmail.com

Resumen

La presente investigación pretende coadyuvar en procesos de optimización, en el manejo y administración de bases de datos distribuidas, con el fin de mejorar la lógica de transferencia de datos en procesos de replicación de bases de datos tradicional, a través de la elaboración de una estructura algorítmica del tipo evolutivo asíncrono, basado en heurísticas y modelos formales de programación.

Palabras Claves

Replicación, asíncrona, bases de datos, algoritmos, estructura algorítmica, evolutivos, banco de datos, información, procesos.

Abstract

This research aims to assist in optimizing processes in the management and administration of distributed databases, in order to improve data transfer logic replication processes traditional database, through the development of a structure asynchronous evolutionary algorithmic type, based on heuristics and formal programming models.

Key Words

Replication, asynchronous, databases, algorithms, algorithmic structure, evolutionary, information processes.

1 Ingeniero en Sistemas, entrenador en competencias de Programación ACM ICPC Bolivia. Desarrollador de sistemas informáticos multiplataforma.

1. Introducción

Hoy en día, en nuestro diario vivir, vamos generando volúmenes grandes de información, la misma, que necesita ser almacenada en forma ordenada y segura , en repositorios de datos, banco de datos o más conocidas como bases de datos. “Las bases de datos, juegan un papel muy importante en el mundo de los negocios, a través de ellas, las empresas obtienen información que les permite tomar decisiones, sobre el lanzamiento, distribución y elaboración de su nuevo producto o servicio”, afirmó la Dra. Malú Castellanos, investigadora de HP Laboratorios, durante su participación en el 1er. Taller de Investigación y Escuela Temática: De los datos al conocimiento, que se realizó en la Universidad de las Américas Puebla (Castellanos, 2011).

La cantidad de información que se genera a nivel mundial cada año, se duplica, y a finales de 2011, se alcanzó almacenar, la cantidad de 1.8 zettabytes , es decir, 1.8 trillones de gigabytes (Beneito, 2013).

Antes esta creciente avalancha de datos, la industria de almacenamiento digital enfrenta el reto de ofrecer soluciones que permitan administrar y almacenar grandes cantidades de información en menos espacio, de una forma más rápida, segura y sustentable.

Entonces, ¿Cómo es posible mejorar la lógica de transferencia de datos, en procesos de replicación de bases de datos transaccionales?

Por lo tanto, la presente investigación determina la siguiente hipótesis principal: la aplicación de algoritmos evolutivos asíncronos, ayuda a mejorar la lógica de transferencia de datos en los procesos de replicación de bases de datos transaccionales grandes.

2. Objetivos

2.1 Objetivo General

Mejorar la lógica de transferencia de datos, en procesos de replicación

de bases de datos tradicionales, a través del estudio de algoritmos del tipo evolutivo asíncrono, basado en heurísticas y modelos formales de programación.

2.2 Objetivos específicos

- Analizar los métodos y técnicas de elaboración de algoritmos evolutivos, del tipo asíncrono, basados en heurísticas y modelos formales de programación, representados en pseudocódigo para la implementación del mismo.
- Analizar técnicas de replicación de bases de datos tradicionales, para determinar los puntos críticos en el desarrollo del proceso.
- Definir una estructura algorítmica basada en heurísticas matemáticas, que nos permitan reducir los tiempos de transferencia de datos, en el proceso de la replicación.

3. Replicación de bases de datos

La replicación de bases de datos es la creación y el mantenimiento de múltiples copias de la misma base de datos. En la mayoría de las implementaciones de replicación, un servidor mantiene la copia maestra de la base de datos y los servidores adicionales mantienen copias esclavo (Bertone, Ruscuni, Milatón, & Mauriello, 2011).

Hay muchas técnicas para gestionar la replicación. Éstas se pueden clasificar de diferentes maneras, entorno a las necesidades que se tiene en el ambiente de trabajo. En este apartado explicamos dos parámetros para presentar dos posibles clasificaciones:

- qué réplica se cambia y
- cuándo se propagan las modificaciones al resto de réplicas.

Según el primer parámetro, los protocolos de replicación se pueden clasificar en single-master y multi-master. Según el segundo parámetro, en síncronas (Eager) o asíncronas (Lazy).

3.1 Single frente a Multi-masters

En el caso del single-master hay una copia principal de cada objeto, que la llamaremos primaria. Cuando hay una modificación, ésta se aplica primero a la copia primaria. Después se propaga al resto de copias (que son las secundarias). En este modelo se puede leer cualquier réplica de un objeto, pero sólo se puede modificar la primaria.

En la aproximación multi-master hay varios almacenes que contienen una copia primaria de un mismo objeto. Todas estas copias se pueden actualizar de forma concurrente. En este caso se puede acceder por lectura a cualquiera de las copias y por modificación a cualquiera de las primarias.

La aproximación multi-master reduce los cuellos de botella y los puntos de fallo, así también existen protocolos que aprovechan las ventajas de los sistemas de comunicación en grupo para evitar así algunos problemas de rendimiento.

Con este tipo de técnicas se consigue que, aunque haya varias copias de un mismo objeto, el usuario perciba que el comportamiento es como si sólo hubiera una. Este criterio de consistencia se conoce como one-copy seriability.

Finalmente, la operación de actualización acaba. En este momento todas las réplicas del objeto tienen el mismo valor. Es importante destacar que la operación no retorna hasta que todas las réplicas han aplicado la actualización.

La gran ventaja de los protocolos síncronos es que evitan la divergencia entre réplicas de un mismo dato.

El gran inconveniente es que cualquier escritura tiene que actualizar muchas o todas las réplicas antes de finalizar. Eso es un inconveniente para sistemas cuyos nodos sean dinámicos (sistemas de igual a igual o sistemas que permiten el trabajo en desconectado) o en entornos de gran alcance donde a causa de la latencia, es costoso en tiempo actualizar todas las réplicas (Gavilaud, 2002).

Además, tiene grandes limitaciones de escalabilidad debidas al tiempo necesario para actualizar todas las réplicas.

3.2 Replicación síncrona y asíncrona de bases de datos

La réplica de datos asíncrona trata de actualizar las bases de datos que residen en un sitio replicado después de que la base de datos primaria haya confirmado un cambio.

Con la réplica asíncrona, el retardo para actualizar las bases de datos de sitio replicado puede variar en función de la aplicación de empresa y los requisitos de usuario. Sin embargo, los datos se sincronizan finalmente con el mismo valor en todos los sitios. La principal ventaja de este tipo de réplica de datos consiste en que si falla un servidor de bases de datos determinado, el proceso de réplica puede continuar y se confirmarán todas las transacciones del sistema de réplica (Bertone, Ruscuni, Milatón, & Mauriello, 2011).

En cambio, la réplica de datos síncrona replica los datos inmediatamente cuando se actualizan los datos fuente. La réplica de datos síncrona utiliza la tecnología de confirmación en dos fases para proteger la integridad de los datos.

En una confirmación en dos fases, sólo se aplica una transacción si todos los sitios distribuidos interconectados acuerdan aceptar la transacción. La réplica de datos síncrona es apropiada para aplicaciones que requieren la sincronización de datos inmediata. Sin embargo, la réplica de datos síncrona necesita que todos los componentes de hardware y las redes del sistema de réplica estén disponibles en todo momento (Milán Franco, 2008).

Normalmente se prefiere la réplica asíncrona porque permite anomalías de sistema y de red.

La réplica asíncrona permite los modelos de réplica siguientes:

- Primario a destino (Sistema de réplica de primario a destino)

- Todos los cambios de base de datos se originan en la base de datos primaria y se replican en las bases de datos de destino. Los cambios efectuados en las bases de datos de destino no se replican en la base de datos primaria.
- Actualización en cualquier lugar (Sistema de réplica de actualización en cualquier lugar)
- Todas las bases de datos tienen posibilidades de lectura y grabación. Las actualizaciones se aplican en todas las bases de datos.

El modelo de actualización en cualquier lugar proporciona un mayor reto en la réplica asíncrona. Por ejemplo, si un sistema de réplica contiene tres sitios de réplica, todos los cuales tienen posibilidades de lectura y grabación, se producen conflictos cuando los sitios intentan actualizar los mismos datos al mismo tiempo. Se deberán detectar y resolver los conflictos para que los elementos de datos tengan finalmente el mismo valor en cada sitio.

4. Algoritmos evolutivos de replicación de bases de datos

Un algoritmo genético o evolutivo aplica los principios de la evolución que se encuentran en la naturaleza para encontrar la mejor solución a un problema. En un "algoritmo genético," el problema está codificado en una serie de cadenas de bits que son manipuladas por el algoritmo, denominadas nodos; en un "algoritmo evolutivo," las variables de decisión y funciones de problemas se utilizan directamente (Darwin & Wallace, 2009).

Entre algunos algoritmos evolutivos de replicación de bases de datos tenemos:

4.1 Algoritmo de reserva en dos fases

La reserva en dos fases es un mecanismo sencillo para garantizar seriabilidad y la primera copia seria (en inglés, one-copy seriability).

Seriabilidad: propiedad que provoca que el resultado de ejecutar una operación sea el mismo que si el resultado se hubiera ejecutado de manera secuencial (sin superposiciones debidas a la concurrencia).

One-copy seriability: propiedad que genera que un usuario perciba el comportamiento de un conjunto de copias de un dato replicado como si sólo hubiera uno.

El protocolo de reserva en dos fases (Castillo Valdivieso, 2000) gestiona las reservas (locks, en inglés) durante la ejecución de la transacción en las dos fases siguientes:

- Se adquieren todas las reservas y no se libera ninguna.
- Se liberan las reservas y no se adquiere ninguna.

Se garantiza la seriabilidad para ejecuciones que sigan este orden en la gestión de las reservas (Juarez Rodriguez, 2011).

4.2 Algoritmo de confirmación distribuida

Los algoritmos de confirmación distribuida son útiles para situaciones en las que interesa garantizar que todos los procesos de un grupo ejecutan una operación o que ninguno de ellos la ejecuta. En el caso de multicast fiable, la operación que se ejecuta sería la entrega del mensaje. En el caso de transacciones distribuidas, la operación sería la realización de la transacción (Tozo, 2009).

Generalmente, las operaciones de confirmación distribuida se basan en un coordinador que notifica al resto de procesos que ya pueden realizar (o no) la operación en cuestión (en local). Claramente ya se ve que si uno de los procesos no puede hacer la operación, no hay manera de notificarlo al coordinador.

Por este motivo son necesarios unos mecanismos más sofisticados para poder hacer estas confirmaciones distribuidas.

A continuación presentamos la confirmación en dos fases y la confirmación en tres fases.

4.3 Algoritmo de confirmación en dos fases

Es un algoritmo muy popular para hacer la confirmación distribuida (Tozo, 2009). Se basa en tener un coordinador y el resto de procesos. Si suponemos que no hay fallos, el funcionamiento del algoritmo sería el siguiente:

- El coordinador envía una petición de confirmación al resto de participantes.
- El coordinador espera hasta que recibe un mensaje de cada uno del resto de participantes.
- Cuando un participante recibe un mensaje de petición de confirmación contesta al coordinador un mensaje indicando si está de acuerdo en hacer la confirmación local o de aborto si no la puede hacer.

4.4 Algoritmo de confirmación en tres fases

Un problema del protocolo de confirmación en dos fases es que cuando el coordinador falla, los participantes pueden no ser capaces de llegar a una decisión final. Eso puede provocar que los participantes se queden bloqueados hasta que el coordinador se recupere. Aunque el protocolo de confirmación en tres fases sea no bloqueante, no se utiliza demasiado en la práctica, ya que las condiciones en las que el protocolo de confirmación en dos fases se bloquea ocurren raramente (Tozo, 2009).

La principal desventaja de este algoritmo es que no se puede recuperar de un fallo de partición de la red. Es decir, si los nodos se separan en dos mitades iguales, cada mitad continuará por su cuenta.

5. Desarrollo

Debido a las limitaciones, que nos proporcionan la utilización de métodos y algoritmos del tipo síncrono, ya mencionados anteriormente, este artículo de investigación propone una alternativa de solución en base a la organización y reestructuración de los datos a replicar, con el fin de poder mantener actualizados los viejos y nuevos datos en las base de datos, proponiendo básicamente tres etapas: recolección de información de la

transacción, mapeamiento y la ejecución.

5.1 Recolección de información de la transacción

La etapa de recolección de información de las transacciones es ejecutada a partir de un trigger bastante simple y genérico, al cual se lo denomino trigger recolector, todo esto, con el objetivo de recolectar toda la información de la transacción, por ejemplo referente a si ha sido insertada, actualizada, o borrada o saber cuáles fueron los valores anteriores o cuales son los valores que serán actualizados.

En esta etapa básicamente, se realiza lo que comúnmente conocemos como ¨Normalización de la base de datos¨, a partir de un trigger recolector.

5.2 Etapa de mapeamiento

El objetivo de la etapa de mapeo es definir un mapa del origen al destino de las transacciones.
Por ejemplo en la figura 1,podemos ver que la tabla principal llamada detalle, tiene como origen los campos entidad, fondo, clase, recurso; y como destino las tablas entidad, fondo, clase y recurso; obviamente cada una con sus respectiva clave primaria, antecesora a su clave foránea.

Figura 1
Mapeamiento de la tabla detalle
Fuente: Elaboración propia.

mapeo_detalle
detalle_entidad =>entidad_codigo=>if(codigo=true)(entidad=>nombre)
detalle_fondo =>fondo_codigo=>if(codigo=true)(entidad=>nombre)
detalle_clase =>clase_codigo=>if(codigo=true)(entidad=>nombre)
detalle_recurso =>recurso_codigo=>if(código=true)(entidad=>nombre)
..... y así sucesivamente para cada relación.

Aplicamos el mismo procedimiento, para cada campo relacional de la tabla a replicarse.

5.3 Etapa de ejecución

Con cierta frecuencia, que se define por el administrador de base de datos, se realiza el proceso de replicación de los datos, todo esto con el fin de que los esquemas viejos y nuevos se mantengan actualizados. El objetivo es procesar la información recogida en la primera etapa, siguiendo las asignaciones de la segunda.
A continuación, se puede ver las principales acciones llevadas a cabo, en un proceso de replicación:

- Mapeo de lectura: El proceso lee el mapa para deducir el origen y destino de la información.
- Uso de transacciones recogidas: La información de las transacciones se leen y se procesan de manera que es posible reproducir las operaciones en la tabla de destino.
- Modificadores: Las tablas definidas geográficamente conforme al proceso de mapeamiento se actualizan.
- Grabación de registros de ejecución: El resultado se registra en cada tabla y campo afectado.

Cuando los tres pasos se encuentran dentro de los disparadores en el formato propuesto por Ambler (Ambler & Sadalage, 2006) la actualización de los esquemas es sincrónica.

A diferencia, que cuando sólo la etapa de agrupamiento o recolección se realiza junto con la transacción de aplicación y la fase de ejecución (que es el paso que en realidad realiza el trabajo de actualización) se lleva a cabo en un momento posterior, se puede llamar a la actualización de forma asíncrona.

Como veremos, la presente investigación, propone una nueva forma de estructuración del algoritmo, que básicamente permite la resolución de los problemas señalados en el enfoque propuesto por (Ambler & Sadalage, 2006).

6. Hipótesis principal

"La aplicación de algoritmos evolutivos asíncronos, ayuda a mejorar la lógica de transferencia de datos en los procesos de replicación de bases de datos" (Colquehuanca Javieri, 2015).

Para una mejor demostración de la hipótesis principal planteada en la presente investigación, podemos deducir las siguientes variables de entorno, donde cada una de ellas evaluará el desempeño adquirido a cada determinada característica necesaria en un proceso de replicación, estas variables son:

- Variable 1: Tiempo.- El tiempo pendiente necesario para procesar un método de operación asincrónica utilizando algoritmos evolutivos, es pequeño y está en el orden de decenas de milisegundos.
- Variable 2: Bloqueos.- El método sincrónico de replicación de bases de datos sin la utilización de algoritmos evolutivos, genera muchos bloqueos, la cantidad es mayor que el método asincrónico, utilizando algoritmos evolutivos.
- Variable 3: Rendimiento.- La actualización usando un método de replicación asíncrona, tiene un rendimiento, medido por el número de operaciones que se actualiza por segundo, mayor que cuando se utiliza el método sincrónico.

Las variables 1 y 2 tienen como foco principal, la realización de la comparación entre los métodos sincrónicos y asincrónicos en términos de rendimiento.

El primero mirando la cantidad de bloqueos y el segundo mirando para el número de operaciones por segundo.

La variable 3 se refiere al período de que la tabla en estudio es inconsistente, porque sólo después de la ejecución del proceso de replicación es que no habrá más inconsistencia.

El tiempo promedio de procesamiento de una operación pendiente proporcionará una estimación del período de falta de coherencia en la tabla.

7. La estructura algorítmica

La estructura algorítmica propuesta en la presente investigación, consta de tres etapas principales: la recolección de información, el mapeamiento y la ejecución. Adicionalmente a estos tres pasos principales, explicaremos inicialmente la: "Preparación del ambiente".

7.1 Preparación del ambiente

Para una ejecución correcta de la estructura algorítmica, se ruega cumplir con los siguientes requisitos mínimos:

7.1.1 Ambiente Físico

La siguiente descripción de requisitos está basada en la experimentación realizada en el cuarto capítulo del trabajo de tesis de grado denominado "Replicación asíncrona de base de datos utilizando algoritmos evolutivos" ubicado en (Colquehuanca Javieri, 2015).

Mínimamente 2 Servidores que cumplan las siguientes características físicas:

Procesador	:	Intel Xeon Family t2.micro o superior - AMD Opteron 3300 o superior
Memoria ram	:	1GB
Velocidad	:	2.5 GHz
Disco duro	:	8GB

7.1.2 Ambiente Lógico

Sistema operativo	:	Ubuntu Server 10.04 LTS (HVM), SSD Volume Type – ami-29ebb519 o superior
Puertos de comunicación	:	SSH,HTTP,HTTPS,5432

Sistema gestor de base de datos	:	PostgreSQL 8.4 o superior
Herramienta de replicación	:	Bucardo
Lenguajes de programación	:	PERL ,PHP y PLSQL
Lenguajes de consultas	:	SQL

7.2 Recolección de Información

Para la primera etapa de la estructura algorítmica, vamos a realizar una copia de los nuevos y viejos datos de las tablas a replicarse; para ello ejecutaremos la siguiente línea de instrucción PL-SQL en nuestro SGBD-PG(Sistema Gestor de Base de Datos PostgreSQL) :

*select * into nombre-de-la-tabla-a-replicar_nuevo from nombre-de-la-tabla-a-replicar_viejo;*

Una vez que realizamos la copia de seguridad para cada una de las tablas a replicarse, procedemos a ejecutar nuestro script denominado "trigger recolector" en nuestro SGBD-PGSQL , bajo la siguiente instrucción PL-SQL:

```
CREATE OR REPLACE FUNCTION recolector_nombre_de_la_tabla()
RETURNS TRIGGER AS $trigger$
DECLARE BEGIN
        INSERT INTO nombre_de_la_tabla_nuevo("columna1",
        "coluemna2", "columna3", "columnaN",) select *
        from nombre_de_la_tabla;
        RETURN NULL;
END;
$trigger$LANGUAGE plpgsql;
```

7.3 Mapeamiento

En la segunda etapa de la estructura algorítmica, el objetivo es verificar si todas las tablas a replicarse contienen una clave primaria; para ello ejecutaremos la siguiente línea de instrucción PL-SQL en nuestro SGBD-PGSQL, para cada una de las tablas a replicarse:

ALTER TABLE nombre_de_la_tabla ADD COLUMN pk_nombre_de_la_tabla serial
ALTER TABLE "public"."nombre_de_la_tabla" ADD PRIMARY KEY ("pknombre_de_la_tabla");

En caso de que no exista una clave primaria, para una determinada tabla a replicarse, la instrucción anterior se encargará de la asignación de una nueva clave primaria a la tabla.

7.4 Ejecución

Para la ejecución de la tercera etapa de la estructura algorítmica, se supone que ya han sido ejecutados los pasos 1, 2, y 3 (preparación del ambiente, recolección de información y mapeamiento) de la presente investigación.

Para la siguiente experimentación, manejaremos una estructura de replicación maestro maestro; por lo que todos los nodos incluidos en el proceso se replicarán del nodo origen al nodo destino y viceversa, con el fin de mantener ambos nodos actualizados.

Para ello procedemos a la carga de la estructura algorítmica,a nuestra herramienta de procesos de replicación basados en PostgreSQL, denominado "bucardo", bajo los siguientes pasos :

- Descargar la herramienta de replicación en https://bucardo.org/downloads/Bucardo-5.4.0.tar.gz
- Descargar la estructura algorítmica en https://github.com/mijacolue/tesismijael.git
- Instalar la herramienta de replicación
- Ejecutar el shell de instrucciones tesismijael.git en una terminal linux.

Una vez ejecutados ambos archivos de instalación procedemos a la configuración de nuestros nodos de replicación :

bucardo add database nombre_servidor_origen dbname=$nombre_base_de_datos host=$host_servidor_origen user=$usuario password=$password port=5432

bucardo add database nombre_servidor_destino dbname=$nombre_base_de_datos host=$host_servidor_destino user=$usuario password=$password port=5432

bucardo add table $tabla_a_replicarse db=$nombre_base_de_datos_origen
bucardo add table $tabla_a_replicarse db=$nombre_base_de_datos_destino

bucardo add herd grupo_replicacion_ida $tabla_a_replicarse db=$nombre_base_de_datos_origen
bucardo add herd grupo_replicacion_vuelta $tabla_a_replicarse db=$nombre_base_de_datos_destino

bucardo add sync $nombre_base_de_datos_origen-sync-ida relgroup=grupo_replicacion_ida dbs=$nombre_base_de_datos_origen,$nombre_base_de_datos_destino

bucardo start

Listo, ambos nodos de replicación están conectados, y las bases de datos están listas para realizar la réplica asíncrona de datos.

8. Conclusiones

Partiendo de la hipótesis principal, y haciendo referencia a las variables de apoyo, expuestas anteriormente, tenemos las siguientes conclusiones

La variable 1 fue demostrada, mediante el análisis por el método estadístico coeficiente kappa, entorno a la cantidad de procesos y los tiempos de procesamiento para cada nodo de replicación, donde se concluyó, que el tiempo promedio en procesamientos de replicación varió de 4 milisegundos a 30 milisegundos para todos los niveles de concurrencia, es decir casi un 50% menor al método síncrono.

La variable 2 también fue demostrada, mediante el análisis por el método estadístico coeficiente kappa, entorno a la cantidad de procesos y la cantidad de bloqueos expresados en porcentaje, para cada nodo de replicación, donde se demuestra que la variable es verdadera.

En todos los niveles de concurrencia, el método sincrónico genera por lo menos tres veces más obstrucciones que un escenario sin trigger y por lo menos el doble que el método asincrónico.

Por último ,la variable 3 también fue demostrada, , mediante el análisis por el método estadístico coeficiente kappa, entorno a la cantidad de operaciones y los tiempos de procesamiento para cada nodo de replicación, respecto a un rendimiento alto, medio y bajo, donde se demuestra que la variable es verdadera.

Ya que se demostró que para los medios y altos niveles de concurrencia, el método asíncrono es el mejor. La excepción es cuando el nivel de concurrencia es bajo, en el que el método sincrónico tiene un mejor rendimiento debido a la cantidad de bloqueos no interfieren con su rendimiento.
En esta situación, no hay ninguna ventaja en el trabajo de forma asincrónica. Es importante señalar que muchos sistemas no críticos trabajan en una baja concurrencia y por lo tanto la aplicabilidad de la solución síncrona no es alta.

Por otra parte, para los sistemas críticos típicos, que tienen un nivel medio o alto de la concurrencia, la mejor solución es el método asíncrono para la replicación de datos.

El rendimiento es un factor clave en esta situación y las ganancias de 100% con respecto al método síncrono son notorias, ya que si trabajamos con un método síncrono bajo una alta concurrencia de datos, existirán problemas de tiempo insuficiente o tiempo de espera excedido o limitado.

Entorno a los objetivos específicos, podemos concluir:
Los objetivos específicos 1,2 y 3 , fueron demostrados bajo el análisis de las variables 1, 2 y 3, ya que la utilización de métodos asíncronos en los

procesos de replicación ,entorno a la estructura algorítmica propuesta, no genera bloqueos, ofrece un mejor rendimiento y los tiempos promedios de proceso son menores al síncrono.

Por todo lo mencionado anteriormente, la hipótesis principal de esta investigación queda entonces pues demostrada.

9. Recomendaciones

Mediante el presente tópico deseo motivar a otros estudiantes de pre y porque no, de postgrado, a realizar investigaciones similares, acerca del tema, ya que a mi parecer solo se investigó, solo uno de los muchos factores que podrían mejorar la lógica de replicación de bases de datos.

A continuación, se describe algunos temas de investigación, que a mi parecer son importantes, para investigaciones futuras:

El esquema algorítmico se puede mejorar, tomando en cuenta nuevos coeficientes de variabilidad, los cuales pueden ser generados y almacenados conforme se hagan las mediciones periódicas de las fuentes emisoras más significativas en nuestro entorno de trabajo.

Tras la aplicación de la herramienta, es importante poner en su uso en un entorno real con una base de datos utilizada por un gran número de aplicaciones distintas y heterogéneas. En este entorno, es necesario volver a hacer el experimento para verificar el comportamiento de la replicación asincrónica. Aunque el rendimiento de la solución es un factor relevante para ser medido y verificado, es posible que, en un entorno real, definimos métricas adicionales relacionados con la usabilidad que nos informan si realmente, con la herramienta, las refactorizaciones base de datos se pueden implementar de una manera fácil e intuitiva.

Referencias

Ambler, S., & Sadalage, P. (2006). Refactoring Databases: Evolutionary Database Design. Chicago, p.360-385

Beneito, R. (10 de enero de 2013). ¿Cuánta Información se Genera y Almacena en el mundo? Obtenido de http://documania20.wordpress.com/2013/09/16/cuanta-informacion-segenera-y-almacena-en-el-mundo/, p.1.

Bertone, R., Ruscuni, S., Milatón, I., & Mauriello, A. (2011). Análisis de rendimiento y replicación en Bases de datos distribuidas. Revista de Investigación y Desarrollo en Informática - Facultad de Informática. UNLP, p.1-3.

Castellanos, D. M. (24 de junio de 2011). Las bases de datos, indispensables para la toma de decisiones. Obtenido de http://blog.udlap.mx/blog/2011/06/laimportanciadelasbasesdedatosradicaenlaayudaparatomardecisiones/, p.1

Castillo Valdivieso, P. (2000). Tesis de doctorado: Optimización de perceptrones multicapa mediante algoritmos evolutivos. Alicante, España: Asociación Española para la Inteligencia Artificial, p. 2-5

Colquehuanca Javieri, M. (2015). Tesis de grado: Replicación asíncrona de bases de datos utilizando algoritmos evolutivos. La Paz, Murillo, Bolivia: Universidad La Salle, p. 16-88

Darwin, & Wallace. (2009). Selección natural: tres fragmentos para la historia. Mexico D-F: CSIC, p. 73-90

Gavilaud, G. (2002). SQL y Algebra relacional. Barcelona: ENI, p. 95-205.

Juarez Rodriguez, J. R. (2011). Tesis doctoral: Protocolos informáticos de replicación de bases de datos. Madrid, España: Universidad Pública de Navarra, p. 15-60

Milán Franco, J. M. (2008). Tesis doctoral:Replicación Autonómica de Bases de datos. Madrid, Madrid, España, p. 11-51

Tozo, G. (2009). Tesis de Maestría: Aplicación de prácticas de agentes en la construcción de Data Warehouse Evolutivo. Sao Paulo, Brasil: IME, p. 23-101

Recibido: 02-07-2015
Aceptado: 25-08-2015