FRAGMENT: A Web Application for Database Fragmentation, Allocation and Replication Over a Cloud Environment

F. Castro, L. Rodríguez, A. López, M. Abud, and G. Alor

***Abstract*—Fragmentation, allocation and replication are techniques widely used in relational databases to improve the performance of operations and reduce their cost in distributed environments. This article shows an analysis of different methods for database fragmentation, allocation and replication and a Web application called FRAGMENT that adopts the work technique that was selected in the analysis stage, because it presents a fragmentation and replication method, it is applied to a cloud environment, it is easy to implement, it focuses on improving the performance of the operations executed on the database, it shows everything necessary for its implementation and is based on a cost model. FRAGMENT analyzes the operations performed in any table of a database, proposes fragmentation schemes based on the most expensive attributes and allocates and replicates a scheme chosen by the user in a distributed environment in the cloud. This work shows a common problem in fragmentation methods, overlapping fragments, and provides an algorithm with an approach to address it. This algorithm results in the predicates that will define each fragment in a distributed environment. To validate the implemented technique, a second web application is presented, dedicated to simulate operations on sites and focused on producing a log file for the main application. Experiments with the TPC-E benchmark demonstrated lower response time of the queries executed against the distributed database generated by FRAGMENT compared with a centralized database.**

***Index Terms*—Data fragmentation, Replication, Cloud environment.**

I. Introducción

Las técnicas de fragmentación, asignación y replicación se presentan en muchos trabajos a lo largo de la literatura, sin embargo, en la mayor parte de estos no incluyen los tres tópicos juntos [1].

Este trabajo fue patrocinado por el Consejo Nacional de Ciencia y Tecnología (CONACYT).

F. Castro, Tecnológico Nacional de México/I. T. Orizaba, Orizaba, Veracruz, México, felipecastromedina123@gmail.com.

L. Rodríguez, Tecnológico Nacional de México/I. T. Orizaba, Orizaba, Veracruz, México, lrodriguezm@ito-depi.edu.mx.

A. López, Universidad Autónoma del Estado de México, Centro Universitario UAEM Zumpango, Zumpango, Estado de México, México, alchau@uaemex.mx.

M. A. Abud, Tecnológico Nacional de México/I. T. Orizaba, Orizaba, Veracruz México, mabud@ito-depi.edu.mx.

G. Alor, Tecnológico Nacional de México/I. T. Orizaba, Orizaba, Veracruz, México, galor@itorizaba.edu.mx.

Un método que, además de incluir estas tres técnicas en conjunto, incluya un modelo de costos, sea fácil de implementar y esté enfocado en la nube es esencial para el desarrollo de este trabajo. Hay un gran número de aspectos a considerar en el problema de asignación de fragmentos a sitios, sin embargo, existen diferentes autores que intentan reducir la complejidad de DAP (*Database Allocation Problem*, Problema de Asignación de Base de Datos) [2]. Abdel et al. [3] presentaron una técnica de fragmentación que contempla la replicación, se aplica a la nube, es de fácil implementación, se enfoca en mejorar el desempeño de la base de datos, presenta todos los detalles para implementarse e incluye rasgos de gran importancia para resolver DAP mediante un modelo de costos simplificado.

Este artículo presenta una aplicación Web llamada FRAGMENT, la cual tiene por objetivo aplicar un esquema adecuado de fragmentación sobre sitios en la nube, realizando las operaciones de asignación y replicación automáticamente, resolviendo de esta manera la problemática antes mencionada. Los esquemas de fragmentación son producidos mediante la técnica propuesta en [3] y para obtener la información requerida por ésta se analiza el *log file* (archivo de carga) de la base de datos. Para la validación de la aplicación Web se presentan dos casos de estudio, uno de ellos basado en una aplicación Web alterna dedicada a la simulación de operaciones y sitios, y el otro basado en un ambiente real en la nube sobre AWS (*Amazon Web Services*, Servicios Web de Amazon) [4]. El resto de este trabajo está organizado de la siguiente manera: La sección II muestra el estado del arte de los trabajos relacionados con los tópicos de interés de esta investigación, en la sección III se presenta la arquitectura y el flujo de trabajo de FRAGMENT, los resultados se discuten en la sección IV y la sección V muestra las conclusiones y el trabajo a futuro.

II. Trabajos Relacionados

Para llevar a cabo este trabajo, los artículos más relacionados, publicados por las principales editoriales científicas, se buscaron usando una metodología para su posterior clasificación. La Fig. 1 muestra la metodología usada para la selección y el análisis de cada artículo.

La metodología se compone de tres etapas. La primera etapa fue buscar en las principales bases de datos de revistas electrónicas una bibliografía completa de investigaciones relevantes sobre fragmentación y replicación de datos en la nube. Las principales bibliotecas digitales consideradas fueron:

1) Biblioteca digital ACM, 2) Biblioteca digital IEEE Xplore, 3) ScienceDirect (Elsevier) y 4) SpringerLink. Los trabajos encontrados que no se publicaron por dichas bibliotecas digitales se categorizaron en "Otras". Se considera que los trabajos publicados por revistas académicas, talleres y actas de conferencias internacionales son confiables y valiosos. Como segunda etapa, se empleó una búsqueda basada en palabras clave para seleccionar los artículos más relevantes. Las palabras clave principales fueron: 1) Fragmentation o partitioning, 2) Replication, 3) Cloud, 4) Cost model y 5) Performance. Se seleccionaron los artículos que cumplían estas características y se descartaron los trabajos que no eran adecuados para el estudio. Los siguientes puntos describen los criterios considerados para la omisión de un trabajo de investigación:

- Documentos de trabajos no publicados, documentos no revisados por pares, documentos que no están en inglés, libros de texto, disertaciones de maestría y doctorado.
- Debido a que la investigación de la fragmentación y replicación de datos en la nube es relativamente actual, solo se incluyeron artículos recientes publicados entre 2010 y 2018.

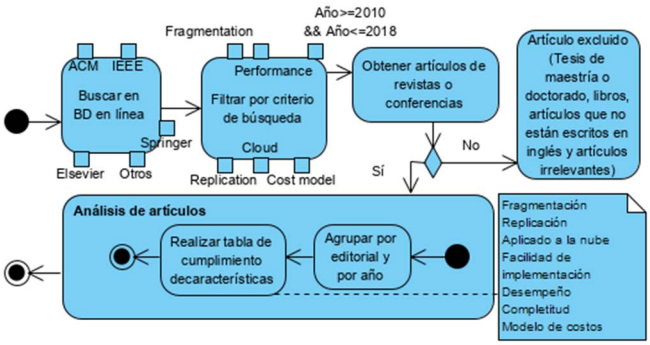

Fig. 1. Diagrama de flujo del criterio de selección.

Para realizar la búsqueda en cada una de las bibliotecas digitales se emplearon los predicados que se muestran en la Tabla I.

TABLA I
PREDICADOS DE BÚSQUEDA EN CADA BIBLIOTECA DIGITAL

Biblioteca digital	Predicado	N° de artículos
ACM	([All: fragmentation] AND [All: replication]) OR [All: fragmentation] AND [All: cloud] AND [Publication Date: (01/01/2010 TO 12/31/2018)]	5,878
IEEE Xplore	((((("All Metadata":fragmentation) AND "All Metadata":replication) OR ("All Metadata":fragmentation) AND "All Metadata":performance) Filters Applied: 2010 - 2018	505
SpringerLink	((fragmentation AND replication) OR fragmentation) AND "Cost Model" AND fragmentation, within 2010 - 2018	326
ScienceDirect	((fragmentation AND replication) OR fragmentation) AND cloud AND performance, Year: 2010-2018	2,854
Otras	"fragmentation or partitioning and database and cloud" and last eight years	15,300

El proceso de selección resultó en 83 artículos más relevantes, que fueron analizados por siete características para registrar si los cumplieron y de qué manera lo hicieron. El análisis completo puede consultarse en [1].

A continuación, se muestran algunos artículos que se obtuvieron durante la metodología de búsqueda, concluyendo con la selección del artículo que cumplió con las características deseadas.

Moral y Kumar [5] propusieron un sistema que usa FASR (*Fragmentation and Sole Replication*, Fragmentación y Replicación Única) para mejorar la seguridad y los tiempos de respuesta en lugar de usar las técnicas tradicionales de encriptación. La selección de los nodos y el almacenamiento están dados por la técnica *t-coloring*. La contribución principal de FASR fue evitar la pérdida de información en caso de un ataque. La fragmentación se realizó de tal forma que cada uno de los fragmentos de un archivo se almacenan en nodos distintos a través de la red.

En [6] se desarrolló un enfoque de heurística optimizada para la fragmentación horizontal y la asignación de datos, el cual tuvo por objetivo proponer una fragmentación de datos adecuada, una asignación de datos precisa y diseñar un algoritmo práctico para el agrupamiento de sitios. La fragmentación de datos que se propuso se lleva a cabo por medio de un modelo matemático de asignación y replicación que optimizó notablemente la fragmentación horizontal.

El estudio presentado en [7] analizó la mejora de la disponibilidad de servicios de almacenamiento federados para proveer mejor calidad en el servicio a los clientes con pocos recursos computacionales. López et al. propusieron una replicación inteligente y un método para mejorar la disponibilidad del almacenamiento a través del análisis de sentimientos del archivo de carga. La contribución principal de [7] es reducir el costo, optimizar recursos y garantizar la disponibilidad requerida por el cliente usando el número mínimo de servicios de replicación.

Kohler et al. [8] presentaron un enfoque que usa los modelos tradicionales de datos relacionales y distribuye los datos por medio de una fragmentación vertical a través de diferentes proveedores en la nube, de esta manera cada proveedor tendrá solo un pequeño pedazo independiente que es inservible sin las otras partes. Este trabajo propuso reescribir las consultas para paralelizarlas y de esta manera mejore el desempeño del esquema. Se mencionó que el enfoque utilizado no es factible en escenarios prácticos del mundo real, ya que se generan grandes pérdidas de rendimiento.

Rodríguez-Mazahua et al. [9] propusieron un conjunto de reglas activas para realizar la fragmentación vertical dinámica sobre bases de datos multimedia. Las reglas activas se implementan en DYMOND (*DYnamic Multimedia ONline Distribution*, Distribución dinámica multimedia en línea), el cual es un sistema basado en reglas activas para la fragmentación vertical dinámica de bases de datos multimedia. La cantidad de accesos locales irrelevantes y la cantidad de accesos remotos relevantes se utilizan para determinar el desempeño límite de una base de datos multimedia y determinar cuándo desencadenar una nueva fragmentación.

En [10] se analizaron tres protocolos para la administración de datos distribuidos en la nube: *ReadOne-Write-All-Available* (ROWAA, Leer Una y Escribir en Todas las Disponibles), *Majority Quorum* (MQ, Mayor Audiencia) y *Data Partitioning* (DP, Partición de Datos). Presentaron BEOWULF, un meta-protocolo basado en un modelo de costos completo que integra los tres protocolos y que selecciona dinámicamente el protocolo con la latencia más baja para una carga de trabajo determinada. Adicionalmente se analizó la calidad de predicción del modelo de costos comparándolo con los costos de un despliegue real.

Marcus et al. [11] presentaron NashDB, un marco de trabajo de distribución de datos adaptativo que se basa en un modelo económico de balance automatizado que suministra y solicita fragmentos de datos, réplicas y nodos agrupados. NashDB adapta sus decisiones a las consultas prioritarias y cambia cargas de trabajo, mientras quita grupos de nodos no utilizados y réplicas redundantes. En este trabajo también se evaluó el modelo NashDB para la eficiente identificación de esquemas de distribución de datos, para que satisfaga la demanda de carga de trabajo, reduciendo la sobrecarga de transmisión de datos.

Brigit et al. [12] propusieron un algoritmo metaheurístico para el particionamiento de bases de datos gráficas a través de los nodos por medio de la colocación de toda la información relacionada en el mismo o en nodos adyacentes. Este algoritmo metaheurístico comprende el mejor ajuste de disminución con la optimización de colonia de hormigas.

En [13] se propuso una estrategia de re-replicación proactiva que utiliza el uso previsto del CPU, del disco y la popularidad de las réplicas para realizar la replicación de manera efectiva y garantizar que todas las cargas de trabajo del servidor estén equilibradas. Los resultados de la simulación demuestran que la utilización de todos los servidores es equilibrada y el enfoque propuesto mejora el rendimiento en términos de tiempo de re-replicación en comparación con el Sistema de archivos distribuidos Hadoop (HDFS).

Abdel et al. [3] desarrollaron un sistema dinámico mejorado de bases de datos distribuidas sobre un ambiente en la nube, el cual permite tomar dinámicamente decisiones de fragmentación, asignación y replicación en tiempo de ejecución. Además, se presentó una técnica mejorada de asignación y replicación que se aplica en una fase inicial del diseño de bases de datos distribuidas cuando no se tiene información de la ejecución de consultas. Mediante la información de las operaciones realizadas a una base de datos se crea una matriz llamada MCRUD (*Create, Read, Update and Delete Matrix*, Matriz de Creación, Lectura, Actualización y Eliminación) que relaciona los tipos de operaciones que se realizaron con sus predicados y los sitios en los que fueron ejecutadas. Por medio de la MCRUD se obtiene la tabla ALP (*Attribute Locality Precedence*, Precendencia de la Localidad del Atributo), la cual muestra los costos de uso de cada atributo. De esta manera se crea un esquema de fragmentación teniendo en cuenta el atributo más costoso.

En la Tabla II se muestra una comparación de los trabajos mencionados en esta sección. Los criterios de comparación son: que los métodos incluyan la fragmentación y la replicación, que estén aplicados a la nube, que sean fáciles de implementar (claridad en la explicación de cada paso), que se enfoquen en el desempeño de las operaciones realizadas en la base de datos, que mencionen todo lo necesario para ser implementados, que estén basados en un modelo de costos y que presenten una solución al problema de fragmentos traslapados.

La Tabla II muestra la importancia de FRAGMENT, ya que se observa que diferentes trabajos incluyen características similares como la fácil implementación o que atienden el problema del desempeño, sin embargo, FRAGMENT resalta por incluir todas estas características juntas.

TABLA II
COMPARACIÓN DE LOS TRABAJOS RELACIONADOS

Artículo	1	2	3	4	5	6	7	8
Moral y Kumar [5]	x	x	x	x	x			
Ali et al. [6]	x	x		x	x		x	
López et al. [7]		x	x		x	x	x	
Kohler et al. [8]	x		x		x	x		
Rodríguez-Mazahua et al. [9]	x			x	x	x	x	
Stiemer et al. [10]		x	x		x		x	
Marcus et al. [11]	x	x	x	x	x		x	
Brigit et al. [12]	x	x		x	x	x	x	
Shwe et al. [13]		x	x	x	x			
Abdel et al. [3]	x	x	x	x	x	x	x	
FRAGMENT	x	x	x	x	x	x	x	x

1)Fragmentación, 2)Replicación, 3)Aplicado a la nube, 4)Facilidad de implementación, 5)Desempeño, 6)Completitud, 7)Modelo de costos, 8)Traslape de fragmentos.

Como se observa en la Tabla II, el método de Abdel et al. [3] cumple con la mayoría de las características, presenta una metodología deseada y una facilidad de implementación de la técnica para incluir los costos en la asignación y la replicación. Por las razones antes mencionadas, la técnica presentada en [3] es utilizada para desarrollar la fragmentación, asignación y replicación en este trabajo de investigación. Adicionalmente, FRAGMENT considera el traslape de fragmentos.

III. FRAGMENT: UNA APLICACIÓN WEB PARA LA FRAGMENTACIÓN, ASIGNACIÓN Y REPLICACIÓN DE BASES DE DATOS EN LA NUBE

La Fig. 2 muestra el proceso que utiliza la aplicación Web para obtener y aplicar el esquema de fragmentación, asignación y replicación. La Fig. 3 representa la arquitectura elegida la cual es "Modelo Vista Controlador" (MVC): en la vista están las páginas que interactúan con el usuario para solicitar información sobre la base de datos y cómo acceder a ella. Los *beans* gestionados por JSF (*JavaServer Faces*) que gestionan el flujo de información se encuentran en el controlador. JSF [14] fue seleccionado debido a la facilidad de implementación de aplicaciones web con este marco de trabajo y porque es de uso gratuito. En el modelo se define la única regla de negocio, la cual es la aplicación de la técnica de fragmentación y replicación, contemplando el cálculo del modelo de costos para una base de datos proporcionada por el usuario. La aplicación se implementó en AWS porque proporciona servicios de computación en la nube confiables, escalables y económicos y porque tiene la mejor accesibilidad al idioma elegido. NetBeans

[15] fue elegido como IDE (*Integrated Development Environment,* Ambiente de Desarrollo Integrado) debido a su interfaz intuitiva y su gran facilidad para desarrollar aplicaciones con el marco de trabajo seleccionado. UWE (*UML-based web engineering*, Ingeniería Web basada en UML) [16] fue seleccionada como metodología para el desarrollo de software, ya que proporciona pasos específicos dirigidos a aplicaciones web y posee una excelente relación con los diagramas UML (*Unified Modeling Language*, Lenguaje de Modelado Unificado) que se considera estándar en el OMG (*Object Management Group*, Grupo de administración de objetos). MySQL [17] fue elegido como DBMS (DataBase Management System), debido a la velocidad en sus consultas y la facilidad de integración con AWS.

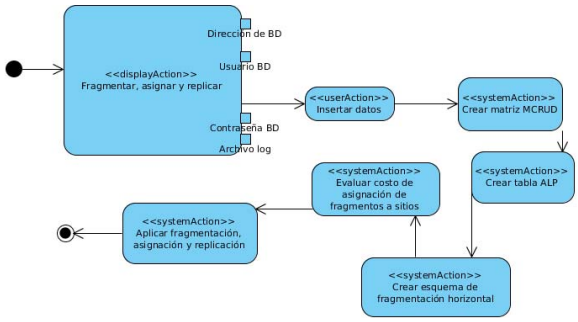

Fig 2. Flujo de trabajo de la aplicación Web.

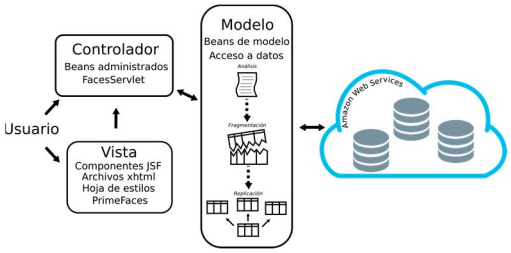

Fig.3. Arquitectura de la aplicación.

Como se observa en la Fig. 2, la primera etapa es obtener los datos de la conexión y el *log file* de la base de datos centralizada. Se muestra en la Fig. 4 la página para la obtención de estos datos y en la Fig. 5 la validación de los datos y la selección de la tabla a ser procesada.

Fig. 4. Formulario de los datos de conexión y el *log file.*

Fig. 5. Estatus de conexión y selección de tabla.

Después de analizar los datos, mediante el *log file* se crea la MCRUD mostrada en la Fig. 6. En la siguiente etapa, por medio de la MCRUD se determina la tabla ALP, la cual se presenta en la Fig. 7. Junto a la tabla ALP también se muestra el esquema propuesto de fragmentación, asignación y replicación, siendo éste la siguiente etapa del flujo de trabajo. Si se presentan dos máximos valores ALP se puede optar por cambiar el atributo principal y de esta manera modificar el esquema de fragmentación, asignación y replicación propuesto.

MCRUD

Predicados	localhost	sitio3	sitio2
id<20	R		
type='Ind'	U		
id=3	U		
id=19		C	D
name='Dhk'		C	D
type='Cor'		C	
customer_id=5		C	
open_date='2008/01/01'		C	
balance=10000		C	

Fig. 6. Matriz MCRUD.

TABLA ALP

Atributo	Valor ALP	
id	4	
type	5	Seleccionar
customer_id	2	
open_date	2	
balance	2	
name	0	

RESULTADO FINAL

Fragmento	Asignar	Replicar	
type='Ind'	localhost	sitio3	✓
type='Cor'	sitio3	localhost	✓

Fig. 7. Tabla ALP y Tabla de Resultados finales.

El Algoritmo 1 representa el pseudocódigo para obtener la matriz de predicados, donde las filas son el conjunto de predicados para obtener cada fragmento. De esta manera, colocando los predicados de cada fila dentro de una consulta, al ser ejecutada se obtendrá cada fragmento con sus correspondientes tuplas. Si la lista de predicados de entrada contiene id<30, id<20, id=40 de una tabla llamada “cuentas”, la cual tiene tuplas con “id” de 1 a 50, como resultado, se obtendrá la matriz mostrada en la Fig. 8.

(id<30)	!(id=40)
!(id<30)	(id=40)
!(id<30)	!(id=40)

Fig. 8. Resultado del ejemplo.

En la Fig. 7, dentro de la tabla del resultado, por cada fragmento se muestra un icono verde, el cual se coloca solo en los fragmentos que serán aplicados, ya que los que no lo presentan se traslapan con algún otro fragmento más grande. Si los predicados mostrados en la tabla de resultados no cubren completamente la tabla original, se presenta un fragmento adicional llamado “Sin fragmento”, y corresponde a todas las tuplas de la tabla original que no están contenidas en los otros fragmentos. Este fragmento solo se asigna, pero no se replica.

Algoritmo 1: Fragmentación con traslape

```
1  input: Lista de predicados[], Nombre de la tabla
2  output: Lista de fragmentos
3  for x from 0 to size of Lista de predicados[] do
4      eliminar todo en línea
5      for y from 0 to size of Lista de predicados[] do
6          if Lista de predicados[y]=Lista de predicados[x] then
7              línea[y]<-- "(" + Lista de predicados[y] + ")"
8          else
9              línea[y]<-- "!(" + Lista de predicados[y] + ")"
10     predicados de fragmentos[x][]<-- línea[]
11 bandera<--false
12 while bandera=false do
13     encontrado<--false;
14     línea de encontrado<--0;
15     for i from 0 to size of predicados de fragmentos[][] while encontrado=false do
16         consulta<--"select * from " + Nombre de la tabla + " where "
17         for j from 0 to size of predicados de fragmentos [i][] do
18             consulta<--consulta + predicados de fragmentos[i][j] + " && "
19         consulta<--consulta - los ultimos tres caracteres
20         consulta<--consulta+" limit 1;"
21         conjunto resultado<--execute consulta in database
22         if conjunto resultado is empty then
23             línea de encontrado<--i
24             encontrado<--true
25     if encontrado=true then
26         índice<--0
27         for q from 0 to size of predicados de fragmentos[línea de encontrado][] do
28             if el primer carácter de predicados de fragmentos[línea de encontrado][q] is "!" then
29                 índice<--q;
30         eliminar línea de encontrado en predicados de fragmentos[][]
31         for q from 0 to size of predicados de fragmentos[][] do
32             eliminar índice en predicados de fragmentos[q][]
33     else
34         bandera<--true
35 consulta<--"select * from " + Nombre de la tabla + " where "
36 eliminar todo en línea
37 foreach x<--predicado in Lista de predicados[], 0<=i<=size of Lista de predicados[] do
38     consulta<--consulta + "!("+x+ ") && "
39 consulta<--consulta - los ultimos tres caracteres
40 consulta<--consulta+" limit 1;"
41 conjunto resultado<--execute consulta in database
42 if conjunto resultado is not empty then
43     for x from 0 to size of Lista de predicados[] do
44         línea[x]<--"!("+Lista de predicados[x]+ ")"
45     predicados de fragmentos[size of predicados de fragmentos][]<-- línea[]
46 return predicados de fragmentos[][]
```

IV. Resultados

A. Caso de Estudio Mediante Simulación

Se desarrolló una aplicación Web alterna para integrar la base de datos del *benchmark* TPC-E [18] y el marco de trabajo CloudSim [19], la cual genera aleatoriamente el número de operaciones y sitios indicados en la página principal, enfocados en la base de datos del *benchmark*. El Algoritmo 2 presenta el pseudocódigo para obtener el archivo Log con el conjunto de sitios y consultas indicados. En la Fig. 9 se muestra la pantalla principal del simulador CloudSim-TPC-E.

Fig. 9. Pantalla principal de Simulación CloudSim – TPC-E.

Una vez generado el archivo de carga, el botón de descarga se habilita para la obtención de este archivo. El archivo que se descarga de este simulador contiene operaciones de lectura, escritura, actualización y eliminación, distribuidas entre sitios simulados. El archivo está en el formato de un *log* común en MySQL.

Algoritmo 2: Simulador de operaciones

```
1  input: Número de sitios, Número de consultas
2  output: Log
3  tablas[]<-- tablas de la base de datos
4  for j from 0 to Número de sitios do
5      Log <-- Log + generar encabezado de Log de MySQL
6                  tomando en cuenta j como pid
7                  y una ip aleatoria
8      for i from 0 to Número de consultas do
9          tabla <-- tablas[Aleatorio]
10         atributos[]<-- obtener atributos de tabla
11         nAtributo <-- Aleatorio menor a tamaño de atributos[]
12         atributo <-- atributos[nAtributo]
13         c_operador[]<-- ">", "<", "<=", ">=", "!=", "="
14         c_operacion0[]<-- "SELECT", "UPDATE", "DELETE", "INSERT"
15         tam0 <-- Número de tuplas de tabla en la base de datos
16         consulta <-- "SELECT " + Nombre de atributos[] + " FROM "
17                         + Nombre tabla + " LIMIT "
18                         + Aleatorio menor a tam0 + ",1;"
19         c_valor[]<-- ejecutar consulta en base de datos
20         if tipo de atributo = alfanumérico then
21             operador0 <-- c_operador0[Aleatorio 4-5]
22             operacion0 <-- c_operacion0[Aleatorio 0-2]
23         else if tipo de atributo = numérico
24                 or tipo de atributo = fecha/tiempo then
25             operador <-- c_operador[Aleatorio 0-5];
26             operacion <-- c_operacion[Aleatorio 0-2];
27         if operacion0 = "SELECT" then
28             Log <-- Log + j + " Query " + operacion0 + " "
29                     + Nombre atributo + " FROM " + Nombre tabla
30                     + " WHERE " + Nombre atributo
31                     + operador0 + c_valor[nAtributo]
32         else if operacion0 = "UPDATE" then
33             Log<-- Log + j + " Query " + operacion0 + " "
34                     + Nombre tabla + " SET " + Nombre atributo
35                     + "=" + Nombre atributo + " WHERE "
36                     + Nombre atributo + operador0 + c_valor[nAtributo]
37         else if operacion0 = "DELETE" then
38             Log <-- Log + j + " Query " + operacion0
39                     + " FROM " + Nombre tabla + " WHERE "
40                     + Nombre atributo + operador0 + c_valor[nAtributo]
41         else if  operacion0 = "INSERT" then
42             Log<-- Log + j + " Query " + operacion0 + " INTO "
43                     + Nombre tabla + " values("
44                     + items de c_valor[] + ");"
45     Log <-- Log + j + " Quit"
46 return Log
```

Para validar la técnica implementada se generó un *log file* con el simulador, configurado para tres sitios con cuatro operaciones cada uno, las cuales se observan en la Tabla III. Este archivo se introdujo en la aplicación Web principal, la tabla seleccionada para procesar fue "SETTLEMENT". La tabla ALP y el resultado final se observa en la Fig. 10.

TABLA ALP

Atributo	Valor ALP	
SE_T_ID	1	
SE_CASH_TYPE	0	
SE_CASH_DUE_DATE	1	
SE_AMT	2	Seleccionar

RESULTADO FINAL

Fragmento	Asignar	Replicar	
SE_AMT>=16198.01	8.5.216.212	192.66.234.124	✓
Sin predicado	220.57.249.11	Sin replica	✓

Fig. 10. Tabla ALP y resultados obtenidos del simulador.

Se observa en la Fig. 10 la tabla ALP, la cual determina que el atributo más costoso es "SE_AMT" y por medio de éste se realiza la tabla de resultados. Se observa que serán aplicados dos fragmentos, uno con el predicado que se encontró de este atributo y el otro con todas las tuplas que no se encuentren contenidas en ese predicado.

B. Caso de Estudio en Amazon Web Services

Para validar la aplicación del método y la generación real de los fragmentos se utilizó una cuenta en *Amazon Web Services*

[4] (Servicios Web de Amazon) y se configuraron tres instancias en EC2 (*Elastic Compute Cloud*, Computación Elástica en la Nube) de AMI (*Amazon Machine Image*, Imagen, de Máquina de Amazon) tipo t2.micro.

TABLA III
OPERACIONES OBTENIDAS MEDIANTE EL SIMULADOR

Operación	Sitio
DELETE FROM FINANCIAL WHERE FI_OUT_BASIC>=3824575572	192.66.234.124
SELECT SE_CASH_DUE_DATE FROM SETTLEMENT WHERE SE_CASH_DUE_DATE<"2005-01-26"	192.66.234.124
DELETE FROM BROKER WHERE B_ID=4300000006	192.66.234.124
INSERT INTO BROKER values(4.300000007E9, "ACTV", "Patrick G. Coder", 174202.0, 5131114.3)	192.66.234.124
SELECT WL_C_ID FROM WATCH_LIST WHERE WL_C_ID!=4300000747	220.57.249.11
SELECT SE_T_ID FROM SETTLEMENT WHERE SE_T_ID=200000000159239	220.57.249.11
DELETE FROM WATCH_LIST WHERE WL_C_ID>4300000281	220.57.249.11
INSERT INTO WATCH_LIST values(4.300000438E9, 4.300000141E9)	220.57.249.11
DELETE FROM SETTLEMENT WHERE SE_AMT>=16198.01	8.5.216.212
DELETE FROM COMPANY WHERE CO_CEO="Amada Lerma"	8.5.216.212
UPDATE HOLDING SET H_DTS=H_DTS WHERE H_DTS<"2005-02-10"	8.5.216.212
INSERT INTO HOLDING values(2.00000001367687E14, 4.3000008454E10, "AP","2005-02-03", 21.64, 200.0)	8.5.216.212

En una de las instancias se implementó la base de datos del *benchmark* TPC-E y se realizaron aleatoriamente las consultas que se muestran en la Tabla IV desde cada uno de los sitios sobre la tabla "CUSTOMER" de la base de datos TPC-E. Cada instancia posee Linux como sistema operativo, 1 GB en memoria RAM, procesador de la familia Intel Xeon con 1 CPU de 2.5 GHz.

En la Fig. 11 se muestra la tabla ALP, en la cual se observa que el atributo más costoso de la tabla "CUSTOMER" es "C_ID" con un valor de 38. En la Fig. 12 se observa el esquema propuesto de asignación y replicación con base en el atributo más costoso.

Atributo	Valor ALP	
C_ID	38	Seleccionar
C_TAX_ID	0	
C_ST_ID	0	
C_L_NAME	0	
C_F_NAME	8	
C_M_NAME	0	

Fig. 11. Tabla ALP de operaciones sobre "CUSTOMER".

El fragmento sin predicado se observa en la última tupla de la tabla, debido a que los fragmentos anteriores no cubren toda la tabla original. Este fragmento solo se asigna, ya que ninguna operación se ejecutó en esta parte de la tabla. Los fragmentos de las líneas 2, 3 y 5 no se aplicarán debido a que son predicados traslapados. Los predicados de las líneas 2 y 3 se traslapan con el predicado de la primera línea, ya que 123 y 4300000300 están contenidos en C_ID<4300000400, de igual forma el predicado de la línea 5 se traslapa con el predicado de la cuarta línea.

TABLA IV
OPERACIONES REALIZADAS EN AMAZON WEB SERVICES

Operación	Sitio
select * from CUSTOMER where C_F_NAME="Joshua"	localhost
select * from CUSTOMER where C_EMAIL_1="JFowle@msn.com"	localhost
update CUSTOMER set C_EMAIL_2="cmgelipe@gmail.com" where C_F_NAME="Joshua" and C_EMAIL_1="JFowle@msn.com"	localhost
select * from CUSTOMER where C_ID<4300000400	localhost
insert into CUSTOMER values(123, "123", "ACTV","CASTRO", "FELIPE", "M", "M", 2, "1993-11-20",4300000505, "123", "123", "123", "123", "123", "123","123", "123", "123", "123", "cmgelipe@gmail.com", "felipecastromedina123@gmail.com");	localhost
select * from CUSTOMER where C_ID<4300000300	localhost
select * from CUSTOMER where C_ID<4300000300	localhost
select * from CUSTOMER where C_ID<4300000300	localhost
select * from CUSTOMER where C_ID<4300000300	localhost
DELETE from CUSTOMER where C_ID=123	localhost
select * from CUSTOMER where C_F_NAME="Magan"	18.188.122.205
select * from CUSTOMER where C_EMAIL_1="MLemarr@rr.com"	18.188.122.205
update CUSTOMER set C_EMAIL_2="correoinventado@gmail.com" where C_F_NAME="Magan" and C_EMAIL_1="MLemarr@rr.com"	18.188.122.205
select * from CUSTOMER where C_ID>4300000600	18.188.122.205
insert into CUSTOMER values(4300001001, "123", "ACTV","CASTRO", "FELIPE", "M", "M", 2, "1993-11-20",4300000505, "123", "123", "123", "123", "123", "123","123", "123", "123", "123", "cmgelipe@gmail.com", "felipecastromedina123@gmail.com");	18.188.122.205
select * from CUSTOMER where C_ID>4300000600	18.188.122.205
select * from CUSTOMER where C_ID>4300000600	18.188.122.205
select * from CUSTOMER where C_ID>4300000600	18.188.122.205
select * from CUSTOMER where C_ID>4300000600	18.188.122.205
DELETE from CUSTOMER where C_ID=4300001001	18.188.122.205
select * from CUSTOMER where C_ID=4300000400	18.223.114.241
UPDATE CUSTOMER set C_EMAIL_1="cmgelipe@gmail.com" where C_ID=4300000401	18.223.114.241
select * from CUSTOMER where C_ID=4300000500	18.223.114.241
UPDATE CUSTOMER set C_EMAIL_1="cmgelipe@gmail.com" where C_ID=4300000500	18.223.114.241
select * from CUSTOMER where C_ID=4300000600	18.223.114.241
UPDATE CUSTOMER set C_EMAIL_1="cmgelipe@gmail.com" where C_ID=4300000600	18.223.114.241
select * from CUSTOMER where C_ID=4300000550	18.223.114.241
UPDATE CUSTOMER set C_EMAIL_1="cmgelipe@gmail.com" where C_ID=4300000550	18.223.114.241
select * from CUSTOMER where C_ID=4300000450	18.223.114.241
UPDATE CUSTOMER set C_EMAIL_1="cmgelipe@gmail.com" where C_ID=4300000450	18.223.114.241

Después de asignar y replicar los fragmentos se realizó una comparación de tiempos de ejecución de las mismas consultas con el esquema fragmentado, ésta se observa en la sección izquierda de la Fig. 13, la cual muestra las consultas más significativas. Cada operación se realizó tres veces, el tiempo mostrado en la gráfica es el promedio de estas ejecuciones.

RESULTADO FINAL

Fragmento	Asignar	Replicar	
C_ID<4300000400	localhost	18.188.122.205	✓
C_ID=123	localhost	18.188.122.205	
C_ID<4300000300	localhost	18.188.122.205	
C_ID>4300000600	18.188.122.205	localhost	✓
C_ID=4300001001	18.188.122.205	localhost	
C_ID=4300000400	18.223.114.241	localhost	✓
C_ID=4300000401	18.223.114.241	localhost	✓
C_ID=4300000500	18.223.114.241	localhost	✓
C_ID=4300000600	18.223.114.241	localhost	✓
C_ID=4300000550	18.223.114.241	localhost	✓
C_ID=4300000450	18.223.114.241	localhost	✓
Sin predicado	18.188.122.205	Sin replica	✓

Fig. 12. Esquema propuesto de asignación y replicación.

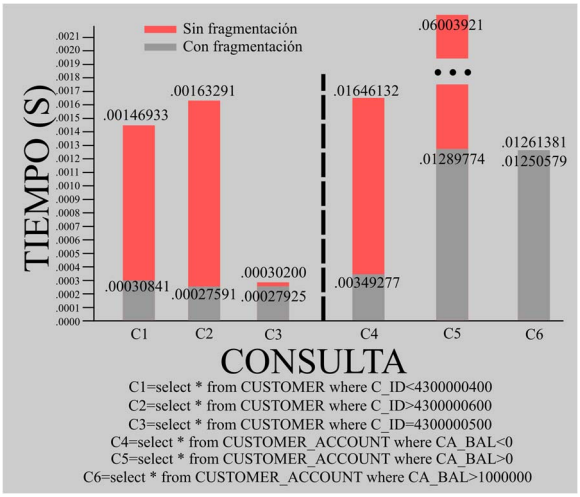

Fig. 13. Comparación de tiempo de ejecución de consultas sobre tabla CUSTOMER y CUSTOMER_ACCOUNT.

Se observa en la Fig. 13 que los tiempos de ejecución de consultas se reducen, debido a que se realizan de manera local, minimizando los costos de transporte de atributos remotos.

Se realizó un nuevo experimento bajo las mismas condiciones de *hardware*, el cual se desempeñó sobre la tabla CUSTOMER_ACCOUNT, la cual pertenece a la misma base de datos del *benchmark* utilizado por el simulador y el anterior experimento. Se muestran en la Tabla V las operaciones realizadas aleatoriamente sobre AWS y el sitio en el que se desempeñaron.

Se observa en la Fig. 14 que dos atributos obtuvieron el mayor valor ALP, debido a esto el DBA (*Database Administrator*, Administrador de Bases de Datos) deberá seleccionar qué atributo será el que determinará el esquema final. En este experimento se utilizó CA_BAL, el cual representa el balance económico de las cuentas de clientes de la base de datos del *benchmark*. En la sección derecha de la Fig. 13 se muestra una gráfica que compara los tiempos de respuesta del esquema fragmentado con tres operaciones representativas de lectura que utilizan predicados obtenidos de la Tabla V.

TABLA ALP

Atributo	Valor ALP	
CA_ID	5	
CA_B_ID	2	
CA_C_ID	5	
CA_NAME	5	
CA_TAX_ST	9	Seleccionar
CA_BAL	9	Seleccionar

Fig. 14. Tabla ALP de operaciones sobre "CUSTOMER_ACCOUNT".

TABLA V
OPERACIONES SOBRE CUSTOMER_ACCOUNT

Operación	Sitio
DELETE FROM CUSTOMER_ACCOUNT WHERE CA_ BAL>0	Localhost
SELECT CA_NAME FROM CUSTOMER_ACCOUNT WHERE CA_BAL<0	Localhost
UPDATE CUSTOMER_ACCOUNT SET CA_NAME="Felipe Castro Medina Fund" WHERE CA_C_ID=4300000001	Localhost
UPDATE CUSTOMER_ACCOUNT SET CA_TAX_ST=2 WHERE CA_ID=43000000002	Localhost
INSERT INTO CUSTOMER_ACCOUNT values(44000000002, 44000000002, 44000000002, "Sunny Martinez Account", 1, 0)	Localhost
DELETE FROM CUSTOMER_ACCOUNT WHERE CA_BAL>0	3.133.58.155
DELETE FROM CUSTOMER_ACCOUNT WHERE CA_BAL<100	3.133.58.155
SELECT CA_B_ID FROM CUSTOMER_ACCOUNT WHERE CA_BAL>=0	3.133.58.155
SELECT CA_NAME FROM CUSTOMER_ACCOUNT WHERE CA_BAL>1000000	3.133.58.155
UPDATE CUSTOMER_ACCOUNT SET CA_TAX_ST=5 WHERE CA_BAL>1000000	3.133.58.155
SELECT CA_C_ID FROM CUSTOMER_ACCOUNT WHERE CA_BAL<0	18.222.180.52
SELECT CA_TAX_ST FROM CUSTOMER_ACCOUNT WHERE CA_BAL<-1000000	18.222.180.52
SELECT CA_C_ID FROM CUSTOMER_ACCOUNT WHERE CA_BAL<-1000000	18.222.180.52
DELETE FROM CUSTOMER_ACCOUNT WHERE CA_BAL=0	18.222.180.52
SELECT CA_NAME FROM CUSTOMER_ACCOUNT WHERE CA_TAX_ST=0	18.222.180.52

Cada operación se realizó tres veces, cada tiempo mostrado en la gráfica es el promedio de estas tres ejecuciones. Se observa que el tiempo de respuesta de las consultas es menor ya que las consultas se realizan localmente y no de manera remota.

V. CONCLUSIONES

Después del análisis de trabajos relacionados, se implementó el método de fragmentación, asignación y replicación basado en el método que propone Abdel et al. [3] por medio de una aplicación Web que realiza todas las operaciones requeridas para asignar y replicar fragmentos sobre un conjunto de sitios mediante el esquema que produce el método y adicionalmente atendiendo el problema de los fragmentos traslapados. Se creó el simulador de operaciones con CloudSim para validar la implementación sobre el *benchmark* TPC-E y por último se utilizó un ambiente real sobre *Amazon Web Services*. Al solucionar el problema de los fragmentos traslapados se obtuvo un esquema adecuado con fragmentos disjuntos y completos, a partir de la relación original. La implementación fue exitosa y el desempeño de la base de datos del *benchmark* mejoró, ya que los tiempos de respuesta de las mismas consultas fueron menores.

FRAGMENT desempeña una fragmentación estática. Este tipo de fragmentación supone un problema de mantenimiento, ya que el DBA debe saber cuándo realizar nuevamente otra fragmentación y esto puede ser una tarea difícil o imposible de

realizar adecuadamente [20]. Como trabajo a futuro se propone agregar un modelo de fragmentación dinámica para que éste se base en umbrales de costo para realizar fragmentaciones posteriores a la inicial.

AGRADECIMIENTOS

Los autores agradecen al Tecnológico Nacional de México por respaldar este trabajo. Además, este proyecto de investigación fue patrocinado por el Consejo Nacional de Ciencia y Tecnología (CONACYT).

REFERENCIAS

[1] F. Castro-Medina, L. Rodríguez-Mazahua, M. A. Abud-Figueroa, C. Romero-Torres, L. Á. Reyes-Hernández and G. Alor-Hernández, "*Application of data fragmentation and replication methods in the cloud: a review*" 2019 International Conference on Electronics, Communications and Computers (CONIELECOMP), Cholula, Mexico, 2019, pp. 47-54.

[2] M. Özsu and P. Valduriez, *Principles of Distributed Database Systems*, Third Edition, 3rd ed. New York, NY: Springer New York, 2011, p. 121.

[3] A. E. Abdel, N. L. Badr and M. F. Tolba, "*Distributed Database System (DSS) Design Over a Cloud Environment*", Multimedia Forensics and Security, Vol. 115, pp. 97-116, 2016.

[4] "Amazon Web Services (AWS) - Cloud Computing Services", *Amazon Web Services, Inc.*, 2019. [Online]. Available: https://aws.amazon.com/. [Accessed: 08- Apr- 2019].

[5] W. D. Moral and B. M. Kumar, "*Improve the data retrieval time and security through fragmentation and replication in the Cloud*", International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), Ramanathapuram, 2016, pp. 539-545.

[6] A. Ali, A. Adel, M. A. Taha, "An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems (DDBSs)", en Heliyon, vol. 3 no. 12, pp. 1-42, 2017.

[7] C. López, R. Heinsen and E. Huh, "Improving Availability Applying Intelligent Replication in Federated Cloud Storage Based on Log Analysis", International Conference on Machine Learning and Soft Computing 2017, Ho Chi Minh City, VN, 2017, pp. 148-153.

[8] J. Kohler, K. Simov, A. Fiech, y T. Specht, "*On the Performance of Query Rewriting in Vertically Distributed Cloud Databases*", en Innovative Approaches and Solutions in Advanced Intelligent Systems, vol. 648, 2016, pp. 59-73.

[9] L. Rodríguez-Mazahua, G. Alor-Hernández, X. Li, J. Cervantes and A. López-Chau, "Active rule base development for dynamic vertical partitioning of multimedia databases", Journal of Intelligent Information Systems, vol. 48, no. 2, pp. 421-451, 2017.

[10] A. Stiemer, I. Fetai and H. Schuldt, "*Analyzing the performance of data replication and data partitioning in the cloud: The BEOWULF approach*", IEEE International Conference on Big Data (Big Data), Washington, DC, 2016, pp. 2837-2846.

[11] R. Marcus, O. Papaemmanouil, S. Semenova and S. Garber, "NashDB: An End-to-End Economic Method for Elastic Database Fragmentation, Replication, and Provisioning", Proceedings of the 2018 International Conference on Management of Data, Houston, TX, USA, 2018, pp. 1253-1267.

[12] A. Brigit, "Data allocation optimization for query processing in graph databases using Lucene", Computers & Electrical Engineering, Vol. 70, pp. 1019-1033, 2018.

[13] T. Shwe and M. Aritsugi, "Proactive Re-replication Strategy in HDFS based Cloud Data Center", Proceedings of the10th International Conference on Utility and Cloud Computing, Austin, TX, USA, 2017, pp. 121-130.

[14] Oracle Corporation (2019, Oct 16). JavaServer Faces Technology [Online]. Available:https://www.oracle.com/technetwork/java/javaee/javaserverfaces 139869.html.

[15] Oracle Corporation (2019, Oct 16). NetBeans IDE [Online]. Available:https://netbeans.org/.

[16] Institute for Informatics Research Unit of Programming and SoftwareEngineering (2019, Oct 16). UWE – UML-based Web Engineering [Online]. Available: http://uwe.pst.ifi.lmu.de/

[17] Oracle Corporation (2018, Oct 13). MySQL [Online]. Available:https://www.mysql.com

[18] "TPC-Homepage V5", *Tpc.org*, 2019. [Online]. Available: http://www.tpc.org/. [Accessed: 19- Apr- 2019].

[19] "The CLOUDS Lab: Flagship Projects - Gridbus and Cloudbus, *Cloudbus.org*, 2019. [Online]. Available: http://www.cloudbus.org/cloudsim/. [Accessed: 19- Apr- 2019].

[20] F. Castro Medina, "Aplicación de métodos de fragmentación y replicación de bases de datos en la nube", Master thesis, Instituto Tecnológico de Orizaba, 2019.

Felipe Castro Medina, Posee el grado de Maestro en Sistemas Computacionales (2019) y actualmente estudia el Doctorado en Ciencias de la Ingeniería en el Instituto Tecnológico de Orizaba. Posee experiencia en el desarrollo de software de sistemas y aplicaciones Web y sus intereses de investigación incluyen el diseño de bases de datos distribuidas en la nube y Big Data.

Lisbeth Rodríguez Mazahua, recibió su licenciatura en informática y su maestría en ciencias de la computación del Instituto Tecnológico de Orizaba, Veracruz, México en 2004 y 2007, respectivamente. En 2012 obtuvo su doctorado en Ciencias en Computación en el Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN). De 2012 a 2014, fue profesora de tiempo completo en la Universidad Autónoma del Estado de México, Centro Universitario UAEM Texcoco. Desde febrero de 2016 es profesora de tiempo completo en el instituto Tecnológico de Orizaba. Sus intereses de investigación actuales incluyen el diseño de bases de datos, sistemas de bases de datos autónomas, bases de datos multimedia, minería de datos y Big Data.

Asdrúbal López Chau, es Doctor en Ciencias en Computación por el CINVESTAV-IPN, México, Maestro en Ciencias en Ingeniería de Cómputo por el CIC-IPN, México, e Ingeniero en Comunicaciones y Electrónica por la ESIME-IPN, México. Ha sido miembro del Sistema Nacional de Investigadores de CONACYT en Nivel I desde enero de 2015. Actualmente es Profesor de Tiempo Completo del Programa Educativo de Ingeniero en Computación, en el Centro Universitario UAEM Zumpango. Sus líneas de investigación son el aprendizaje automático, minería de datos y sistemas empotrados.

María Antonieta Abud Figueroa, nació en la ciudad de Orizaba, Ver. Es ingeniero en electrónica por la UAM-Iztapalapa, México DF en el año 1984, y maestra en ciencias en sistemas de información por el ITESM-Morelos, en la ciudad de Cuernavaca, Mor. en el año 1991. Desde el año 1995 es profesora-investigadora en el área de posgrado del Instituto Tecnológico de Orizaba, en la ciudad de Orizaba, Ver. México. Su línea de investigación es la Ingeniería de Software. La M.C. Abud es miembro de ACM.

Giner Alor Hernández, es un investigador de tiempo completo de la División de Investigación y Estudios de Posgrado en el Instituto Tecnológico de Orizaba. Recibió su maestría y doctorado en Ciencias de la Computación del Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV), México. Es miembro del Sistema Nacional de Investigadores. Sus intereses de investigación incluyen servicios web, comercio electrónico, Web semántica, Web 2.0, arquitecturas orientadas a eventos y orientadas a servicios, e integración de aplicaciones empresariales.