EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Orizaba

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

OPCIÓN I.- TESIS

TRABAJO PROFESIONAL

"DESARROLLO DE UN MÉTODO DE FRAGMENTACIÓN HORIZONTAL DINÁMICA PARA BASES DE DATOS MULTIMEDIA"

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:

I.S.C. Abraham Castillo García

DIRECTOR DE TESIS:

Dra. Lisbeth Rodríguez Mazahua

CODIRECTOR DE TESIS:

M.S.C. Felipe Castro Medina

ORIZABA, VERACRUZ, MÉXICO. ABRIL 2022

Orizaba, Veracruz, 08/abril/2022
Dependencia: **División de Estudios de Posgrado e Investigación**
Asunto: **Autorización de Impresión**
OPCION: **I**

C. ABRAHAM CASTILLO GARCÍA
Candidato a Grado de Maestro en:
SISTEMAS COMPUTACIONALES
P R E S E N T E.-

De acuerdo con el Reglamento de Titulación vigente de los Centros de Enseñanza Técnica Superior, dependiente de la Dirección General de Institutos Tecnológicos de la Secretaría de Educación Pública y habiendo cumplido con todas las indicaciones que la Comisión Revisora le hizo respecto a su Trabajo Profesional titulado:

" Desarrollo de un método de fragmentación horizontal dinámica para bases de datos multimedia "

comunico a Usted que este Departamento concede su autorización para que proceda a la impresión del mismo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
CIENCIA – TÉCNICA - CULTURA®

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE ORIZABA
DIV. DE EST. DE POSGRADO E INVEST.

DR. MARIO LEONCIO ARRIOJA RODRÍGUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN

OG-13-F06

Orizaba, Veracruz, 01/marzo/2022
Asunto: **Revisión de trabajo escrito**

C. MARIO LEONCIO ARRIOJA RODRÍGUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
P R E S E N T E.-

Los que suscriben, miembros del Jurado, han realizado la revisión de la Tesis del (la) C.

ABRAHAM CASTILLO GARCÍA

la cual lleva el título de:

"Desarrollo de un método de fragmentación horizontal dinámica para bases de datos multimedia "

y concluyen que se acepta.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
CIENCIA – TÉCNICA - CULTURA®

PRESIDENTE:	DRA. LISBETH RODRÍGUEZ MAZAHUA	*FIRMA*
SECRETARIO:	M.C.E BEATRIZ ALEJANDRA OLIVARES ZEPAHUA	*FIRMA*
VOCAL:	M.C. MA. ANTONIETA ABUD FIGUEROA	*FIRMA*
VOCAL SUP.:	M.S.C. FELIPE CASTRO MEDINA	*FIRMA*

TA-09-21

Agradecimientos

Agradezco el apoyo de mi familia, por siempre creer en mí, por toda su paciencia, por todo su cariño y comprensión. Doy gracias a mi novia por su infinita paciencia, compresión y ese amor incondicional que me da fuerzas para no rendirme.

Agradezco a mis padres por toda la comprensión y confianza en cada decisión que he tomado en mi vida, de su apoyo, de sus sabios consejos que me han ayudado en los momentos difíciles, a mis hermanos por su apoyo, por su comprensión y cariño. A mi hermano por ser despistado y hacerme reír mucho. A mi hermana por confiar y creer en mí y siempre desear lo mejor para mí.

Agradezco a mi persona especial que siempre está en mi cabeza y corazón, gracias Ale, mi novia, mi amiga, mi confidente, muchas gracias por esa comprensión, por esa alegría que me da ánimos, por cada sonrisa que dibujas en mi rostro, por motivarme, por compartir cada momento conmigo.

Agradezco a la Dra. Lisbeth por ser tan comprensiva, por confiar en mis habilidades, por apoyarme en este proyecto, por brindarme sus conocimientos y experiencias, agradezco su infinita paciencia para explicarme y guiarme en todo el desarrollo de este proyecto se lo agradezco mucho de corazón.

Agradezco especialmente a la maestra Betty por ser una excelente profesora y brindarme todo los conocimientos y herramientas necesarias para ser un gran ingeniero en sistemas computacionales. Agradezco a la maestra Betty por brindarme la oportunidad de realizar el posgrado, por su entusiasmo y su apoyo.

Agradezco a mi amigo Andrés Urbano, por darme ese empujo y apoyarme a realizar mi licenciatura, gracias por creer en mí, amigo.

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo de la beca otorgada durante el periodo de estudios y al Tecnológico Nacional de México (TecNM) por dar soporte a esta investigación y a toda la academia de posgrado, sin sus esfuerzos nada de esto sería posible.

Índice

Índice de figuras

Índice de tablas

Resumen

Los métodos de fragmentación horizontal son muy utilizados en la industria, ya que logran mejorar el rendimiento de las bases de datos, al reducir los tiempos de repuesta de consultas y disminuir el costo de transmisión de datos.

El objetivo de este trabajo fue realizar un análisis comparativo de las diversas técnicas de fragmentación horizontal para posteriormente desarrollar una metodología de fragmentación horizontal dinámica para bases de datos multimedia que cumpla todos los criterios de comparación del análisis: que sea fácil de implementar, que tenga un modelo de costos, que sea capaz de adaptar el esquema, entre otros. Finalmente, la técnica propuesta se validó con una base de datos multimedia denominada HITO (Historia del instituto Tecnológico de Orizaba).

Las tecnologías utilizadas para el desarrollo de este proyecto son como marco de trabajo JavaServer Faces con la biblioteca PrimeFaces, como sistemas gestores de bases de datos MySQL y MongoDB, como lenguaje de programación Java y como IDE (*Integrated Development Environment*, entorno de desarrollo integrado) NetBeans que es el más adecuado para el desarrollo con estas tecnologías.

Abstract

Horizontal fragmentation methods are widely used in the industry since they improve database performance, enhance query response times and reduce the cost of data transmission.

The objective of this work was to perform a comparative analysis of the various horizontal fragmentation techniques to subsequently develop a dynamic horizontal fragmentation methodology for multimedia databases that meets all the analysis comparison criteria: it is easy to implement, has a cost model, is capable of adapting the schema, and others. Finally, the proposed technique was validated with a multimedia database called HITO (History of the Technological Institute Orizaba).

The technologies used for the development of this project are JavaServer Faces as framework with the PrimeFaces library, MySQL and MongoDB as database manager, Java programming language and NetBeans as IDE (Integrated Development Environment), which is the most suitable for the development with these technologies.

Introducción

Los métodos de fragmentación son ampliamente usados en las bases de datos, actualmente existe un gran número de aplicaciones de varios dominios, que generan y recopilan grandes volúmenes de datos, por lo tanto, mejorar el rendimiento de una base de datos por medio de un método de fragmentación es muy necesario.

El problema que se observa en esta investigación es que, hasta ahora existe una carencia de métodos de fragmentación horizontal dinámica para bases de datos multimedia en la literatura. Los investigadores o personas interesadas en técnicas de fragmentación horizontal se beneficiarán con este desarrollo.

Este trabajo resuelve el problema observado haciendo un análisis y comparando técnicas de fragmentación horizontal, para diseñar un método para posteriormente implementar y validar. De esta manera se desarrolla mejor una técnica tomando en cuenta las características obtenidas del análisis realizado.

Este trabajo se organiza en cinco capítulos principales: en el capítulo uno se abarca el marco teórico, planteamiento del problema, objetivo general y específicos y la justificación; el capítulo dos incluye el estado de la práctica y un análisis comparativo; el capítulo tres presenta un análisis profundo de los métodos de fragmentación horizontal y el diseño de la aplicación web y el flujo de trabajo del vigilante de fragmentación para implementar la técnica desarrollada; el capítulo cuatro describe los resultados obtenidos por la aplicación web y el vigilante de fragmentación y la descripción de un caso de estudio para validar su funcionamiento; por último, en el capítulo cinco se presenta las conclusiones y recomendaciones.

Capítulo 1. Antecedentes

1.1 Marco teórico

A continuación, se explicarán los conceptos más importantes relacionados con el trabajo presentado.

1.1.1 Bases de datos multimedia

Una base de datos multimedia se refiere a una colección de datos en la que existen múltiples modalidades de datos, como texto e imágenes. En este sistema de base de datos, los datos de las diferentes modalidades están relacionados entre sí. Por ejemplo, los datos de texto están relacionados con las imágenes como sus datos de anotación [1].

1.1.2 Fragmentación

La fragmentación es el proceso de dividir las tablas relacionales ya sea a través de un operador de selección o de proyección, dicho proceso puede anidar más procesos de fragmentación [2].

El proceso de fragmentación en las bases de datos tiene una gran importancia, ya que gracias a la fragmentación se asegura la colocación de datos lo más cerca posible de las aplicaciones que los utilizan, se concentra un mayor número de operaciones (consultas) sobre un mismo fragmento optimizando los tiempos de respuesta, y, por último, se reducen los costos de almacenamiento de datos en los diferentes sitios [3].

1.1.3 Reglas de fragmentación

Para garantizar que la base de datos no sufra cambios semánticos durante la fragmentación, como la pérdida de datos a consecuencia de esto [2]. Es necesario poder argumentar la integridad y la reconstrucción. Para ello se toman en cuenta las siguientes tres reglas:

1. **Completitud:** Si una instancia de relación R se descompone en fragmentos $F_R = \{R_1, R_2, \ldots, R_n\}$, cada dato que está en R puede encontrarse también en uno o más de los R_i. En el caso de una fragmentación horizontal, el "elemento" suele referirse a una tupla, mientras que, en el caso de la fragmentación vertical, se refiere a un atributo [2].

2. **Reconstrucción:** Si una relación R se descompone en fragmentos $F_R = \{R_1, R_2, \ldots, R_n\}$, debería ser posible definir un operador ∇ relacional tal que

$$R = \nabla R_i, \forall R_i \in F_R$$

El operador será diferente para las distintas formas de fragmentación; sin embargo, es importante que pueda identificarse. La posibilidad de reconstruir la relación a partir de sus fragmentos garantiza la conservación de las restricciones definidas en los datos en forma de dependencias [2].

3. **Disyunción:** Si una relación *R* se descompone en fragmentos $F_R = \{R_1, R_2, \ldots, R_n\}$ y el dato d_i está en R_j, no está ningún otro fragmento R_k ($k \neq j$). Este criterio garantiza que los fragmentos horizontales son disjuntos. Si la relación *R* se descompone verticalmente, sus atributos de clave primaria suelen repetirse en todos sus fragmentos (para la reconstrucción). Por lo tanto, en el caso de la fragmentación vertical, la disyunción se define solo en los atributos de clave no primaria de una relación [2].

1.1.4 Tipos de fragmentación

Los tipos de fragmentación dependerán del operador a utilizar; se considera fragmentación horizontal al utilizar un operador de selección y fragmentación vertical cuando se usa un operador de proyección [2].

1.1.4.1 Fragmentación horizontal (primaria y derivada)

Es el proceso de división de una relación utilizando un operador de selección en el que los predicados de selección determinan la fragmentación a lo largo de sus tuplas (filas de la tabla), existen dos versiones: primaria y derivada [2].

- Fragmentación horizontal primaria: es realizada utilizando predicados que se definen sobre la relación.
- Fragmentación horizontal derivada: se lleva a cabo a través de predicados definidos en otra relación.

1.1.4.2 Fragmentación vertical

Es el proceso de división de una relación utilizando un operador de proyección para obtener un conjunto de relaciones más pequeñas provocando que muchas operaciones se ejecuten en un solo fragmento [2].

1.1.4.3 Fragmentación híbrida

En algunos casos, una simple fragmentación vertical u horizontal de un esquema no es suficiente para satisfacer los requisitos de las aplicaciones. En estos casos el proceso de fragmentación se anida aplicando las dos estrategias de fragmentación una tras otra, a esta alternativa se le llama fragmentación híbrida [2].

1.1.5 Fragmentación dinámica

Enfoque adaptativo que consiste en monitorizar continuamente la base de datos y al detectar suficientes cambios en los patrones de acceso o la carga de trabajo, dar lugar a la re-fragmentación, la cual es considerada un proceso de rediseño [2].

1.2 Situación tecnológica, económica y operativa de la empresa.

El Instituto Tecnológico de Orizaba, ubicado en el estado de Veracruz, es uno de los centros educativos más importantes de la región. Actualmente alberga a más de 6000 estudiantes distribuidos entre las 8 carreras de ingeniería y 6 posgrados.

Esta casa de estudios pertenece al Tecnológico Nacional de México (TecNM) y es una institución sustentable comprometida con la formación de líderes competentes para responder a los retos y expectativas internacionales.

La misión del Instituto se centra en fortalecer los servicios educativos a través de la cobertura, equidad, promoción e inclusión, en la formación integral de los estudiantes impulsando la innovación, ciencia y tecnología; para consolidar la vinculación con pertinencia en los diferentes sectores estratégicos, modernizando la gestión institucional con transparencia y rendición de cuentas en un ámbito sustentable.

Este trabajo se realizó bajo la dirección de la División de Estudios de Posgrado e Investigación, adjunta al Instituto Tecnológico de Orizaba. Se desarrolló como proyecto de investigación para la Maestría en Sistemas Computacionales, programa ofertado en esta división.

1.3 Planteamiento del problema

Las aplicaciones con uso intensivo de datos en varios dominios generan y recopilan cantidades masivas de información en poco tiempo. Una gran parte de los datos utilizados es

contenido multimedia, el cual, debido a su gran tamaño, provoca un crecimiento acelerado en las bases de datos multimedia y, sin una estrategia adecuada, estas se ven afectadas al recuperar la información por el aumento de costos en la trasmisión de datos y la demora en los tiempos de respuesta. Por estas razones, las técnicas de fragmentación horizontal se han aplicado en estas bases de datos para aprovechar sus ventajas potenciales en cuanto a optimización de consultas.

Otro factor para tomar en cuenta son las características dinámicas de las bases de datos multimedia, dado que se tiene un constante cambio de la carga de trabajo, por lo cual las estrategias de fragmentación estática son las menos adecuadas para este tipo de bases de datos.

Debido a que la mayoría de las propuestas presentadas en la literatura se enfocan en fragmentación horizontal estática, en este trabajo se presenta un enfoque para resolver este tipo de problemas con una técnica de fragmentación horizontal dinámica. Para esto, se revisaron los métodos de fragmentación de bases de datos multimedia desarrollados en los últimos diez años, por medio de la búsqueda en las principales bibliotecas digitales como ACM Digital Library, SpringerLink, ScienceDirect y IEEE Xplore, posteriormente se realizó un análisis comparativo para determinar sus ventajas y desventajas. Después se seleccionó uno de los métodos anteriormente analizados para realizar una comparación con el método que se desarrolló en este proyecto.

El uso de la fragmentación horizontal dinámica tiene la ventaja de adaptar un esquema de acuerdo con los cambios de la carga de trabajo, permitiendo que el esquema siga siendo óptimo a través del tiempo, lo cual es muy eficiente en el uso de bases de datos multimedia, permitiendo un menor tiempo de respuesta y costo de ejecución de las consultas.

1.4 Objetivo general y específicos

A continuación, se muestra el objetivo general y específicos.

1.3.1 Objetivo general

Desarrollar un método de fragmentación horizontal para bases de datos multimedia que adapte el esquema de fragmentación de acuerdo con los cambios de los patrones de acceso.

1.3.2 Objetivos específicos

- Estudiar y analizar el estado del arte de los métodos de fragmentación horizontal, estáticos y dinámicos, para bases de datos multimedia, así como de los modelos de costo utilizados para evaluar esquemas de fragmentación.
- Realizar un análisis comparativo de los algoritmos y de los modelos de costo identificados en el estado del arte para conocer sus ventajas y desventajas, además de determinar las bases de datos en las que se implementaron.
- Seleccionar las tecnologías que se utilizarán para el desarrollo del método.
- Diseñar el método y el modelo de costo para la fragmentación horizontal dinámica de bases de datos multimedia.
- Implementar el método y modelo de costo utilizando las tecnologías seleccionadas.
- Comparar el método y modelo de costo con un algoritmo y modelo seleccionados del análisis comparativo.

1.5 Justificación

Debido a la carencia de técnicas de fragmentación horizontal para bases de datos multimedia que adapten el esquema de acuerdo con los cambios en los patrones de acceso, es necesario un método de fragmentación horizontal dinámica para mejorar el desempeño de estas. Los proveedores de servicios de contenido multimedia requieren sistemas capaces de dar respuestas rápidas para obtener contenido y que tengan una gran disponibilidad.

Por medio de este trabajo se benefician los proveedores de servicios de contenido multimedia, los usuarios que renten o usen estos servicios, de igual forma se tienen beneficios para los investigadores del área de bases de datos, ya sea utilizando este método en sus proyectos o para usarlo como un punto de referencia en el desarrollo de nuevas técnicas de fragmentación para bases de datos multimedia. El método antes mencionado permite aumentar la velocidad de respuesta de las consultas y reducir sus costos de ejecución.

Capítulo 2. Estado de la práctica

La fragmentación de datos es un tema altamente abordado por trabajos en los que se busca optimizar los tiempos de respuesta de las consultas realizadas a las bases de datos. Para dar un enfoque que contribuya con este tema es necesario conocer los trabajos relacionados, los cuales se escogieron y analizaron meticulosamente. En este capítulo se describen una serie de trabajos relacionados con la fragmentación de datos, posteriormente se hace una comparación por medio de una tabla, y, por último, se concluye con los hallazgos obtenidos después de haber analizado este número considerable de trabajos.

2.1. Trabajos relacionados

En los sistemas de bases de datos distribuidas se utilizan técnicas de fragmentación, asignación y replicación para mejorar el rendimiento de estas. La principal preocupación del diseño de las bases de datos distribuidas es hacer la fragmentación de relaciones en una base de datos relacional, dado que la mayoría de técnicas de fragmentación no considera las relaciones de los datos al ser fragmentados. Esto provoca que las solicitudes de datos relacionados por parte del usuario y la recuperación de los datos consuman mucho tiempo si lo datos relacionados residen en diferentes fragmentos. Ramachandran et al. [4] presentaron un sistema que considera la relación de los datos durante la fragmentación y se centra en la fragmentación horizontal. El sistema propuesto utiliza un algoritmo para fragmentación horizontal basado en *clustering,* el cual cumple con los objetivos de una base de datos distribuida con desempeño mejorado. Se realizó una comparación de una fragmentación normal y una fragmentación basada en *clustering* sobre tres conjuntos de datos: *marketing* de un banco, ecocardiograma y un censo. Como resultados se obtuvieron mejoras de rendimiento en el proceso de consultas distribuidas en la fragmentación basada en *clustering*. Los autores concluyeron mencionando que la incorporación de técnicas de *clustering* dentro de la fragmentación ayudará a encontrar las relaciones de datos. Esto conducirá a mejores fragmentos y la información solicitada será recuperada fácilmente por el usuario. Como trabajos a futuro se realizará una comparación del algoritmo obtenido con las técnicas de fragmentación estándar y la implementación de fragmentación dinámica de las bases de datos distribuidas basada en técnica de *clustering*.

La invención de nuevas tecnologías, normalización de la industrias, *gadgets* y dispositivos ha producido enormes cantidades de datos que requieren de un sistema esencial de gestión y manipulación de datos. Los sistemas de bases de datos relacionales proveen manipulación eficaz de los datos estructurados, pero actualmente, la velocidad y la naturaleza de los datos usados/generados incrementa exponencialmente y estos sistemas se vuelven incompetentes a la hora de gestionar una cantidad enorme de datos no estructurados. Patil et al. [5] presentaron un estudio detallado sobre las dos bases de datos más populares: MongoDB para bases de datos NoSQL y MySQL para bases de datos relacionales, siguiendo un enfoque cualitativo que les proporciono las ventajas y limitaciones de estas. MongoDB es una base de datos orientada a documentos que provee alto rendimiento, y facilidad de escalabilidad asegurando una efectiva administración de datos con su prominente característica de auto fragmentación (*sharding*). Esta fragmentación divide la base de datos a través de múltiples servidores, incrementando la capacidad y escalabilidad como se requiera. Esta característica permite distribuir los datos en diferentes nodos para maximizar espacio en disco y balancear dinámicamente la carga de las consultas. Realizaron un experimento, el cual consistió en dos conjuntos de datos (un conjunto de datos en MongoDB y un conjunto de datos en MySQL) de páginas de registro e inicio de sesión de una página web, el estudio se realizó tras el ingreso de un millón de registros en las tablas. El análisis realizado obtuvo que el tiempo que tardan en realizar operaciones básicas de inserción y recuperación son sustancialmente menores en un sistema con MongoDB como base de datos, demostrando que MongoDB es más rápido que MySQL en carga de datos de forma dinámica.

La gestión de datos en la era del Big Data es un trabajo muy difícil, porque los servidores ordinarios no pueden cumplir los requisitos de alto rendimiento, rápido tiempo de respuesta y fuerte tolerancia a los fallos. La creación de un *clúster* es una forma rentable de gestionar el Big Data en donde los datos se dividen por algunos de sus atributos. Sin embargo, un plan de fragmentación inadecuado puede aumentar el número de transacciones que deban ejecutarse en múltiples fragmentaciones de datos. Algunos institutos y compañías comerciales introdujeron una gran cantidad de algoritmos para automatizar la fragmentación de bases de datos, pero independientemente del tipo de algoritmo que se adopte, solo se despliega un plan de fragmentación en la base de datos. Al mismo tiempo, diferentes requisitos de consulta necesitan diferentes planes de fragmentación que entran en conflicto.

La replicación es la forma más directa de obtener una fuerte tolerancia a los fallos y se adopta en muchos sistemas de bases de datos. En la mayoría de estos sistemas, todas las réplicas se fragmentan con el mismo plan de fragmentación que el nodo principal y la mayoría de las réplicas permanecen inactivas cuando el nodo principal está activo. Liming et al. [6] presentaron un método que permite identificar y resolver los conflictos por réplicas de fragmentación con diferentes planes de fragmentación. A través de este método las diferentes partes de la carga de trabajo utilizan un plan de fragmentación más apropiado, al mismo tiempo, las réplicas se usan de una manera más eficiente. El método recupera el esquema de la base de datos y analiza la carga de trabajo de muestra, clasifica los atributos de la tabla y la carga de trabajo en diferentes categorías y utiliza el algoritmo de ramificación y poda para encontrar los planes de fragmentación óptimos. Para evaluar el método realizaron varios experimentos para compararlo con el método tradicional de plan de fragmentación de una sola base de datos, se desplegaron de 2-6 planes por separado para evaluar el método. Como resultados, se obtuvo que entre más planes de fragmentación se tengan, los tiempos de transacciones de lectura disminuyen y los tiempos de transacciones de lectura-escritura son más largos con más de tres planes de fragmentación. En conclusión, los autores presentaron una arquitectura de dos niveles (servidor principal y servidor de réplicas) donde las réplicas con diferentes planes de fragmentación pueden funcionar correctamente y con mayor eficacia, aumentando la proporción de las transacciones en un solo sitio y la velocidad de repuesta de las transacciones.

Con el desarrollo de internet, la información de datos crece exponencialmente. Las fragmentaciones horizontales y verticales de las bases de datos son medios importantes para mejorar el rendimiento y la capacidad de gestión de datos, tanto para los servidores de un solo sistema como para los sistemas que no comparten nada (*shared-nothing*). Si embargo, las técnicas tradicionales de fragmentación no pueden tratar con consultas de *stream* de manera eficiente. En Guo et al. [7] presentaron WSPS (*Workload-driven Stream Partitioning System*, Sistema de fragmentación de *Stream* basado en la carga de trabajo) un sistema de fragmentación en línea en tiempo real. WSPS procesa consultas dinámicas a gran escala en entornos de computación en la nube y combina un marco de procesamiento de *Stream* con un algoritmo de fragmentación. Se realizó un experimento comparando WSPS con enfoques tradicionales de fragmentación: Hash, Schism y O^2P. Se utilizaron cuatro diferentes

conjuntos de datos para el experimento: TPCC (*Transaction Processing Performance Council C*, Consejo de Rendimiento del proceso de Transacciones C), TPCE (*Transaction Processing Performance Council E*, Consejo de Rendimiento del Procesamiento de Transacciones E), TPCH (Transaction Processing Performance Council H, Consejo de Rendimiento de Procesamiento de Transacciones H) y YCSB (*Yahoo! Cloud Serving Benchmark, Benchmark* de Servicio en la nube de Yahoo!). Los resultados del experimento mostraron que WSPS es más rápido que los enfoques anteriores y hasta produce resultados de fragmentación de buena calidad. Además, se obtuvieron resultados superiores a los enfoques existentes sobre cargas de trabajo cambiantes.

Con el desarrollo de la tecnología de internet y la computación en la nube, cada vez más aplicaciones tienen que enfrentarse a los retos del *Big Data*. Las bases de datos NoSQL son adecuadas para la gestión de *Big Data* por sus características de alta escalabilidad, así como alta disponibilidad y tolerancia a fallos. Las estrategias de fragmentación de datos juegan un papel importante en las bases de datos NoSQL, sin embargo, las estrategias de fragmentación existentes causan algunos problemas como la baja de escalabilidad, puntos de alta demanda (*hot spot*) y bajo rendimiento. Por ello, en Chen et al [8] presentaron una nueva estrategia de fragmentación de datos HRCH (*Hybrid Range Consistent Hash*), la cual puede fragmentar los datos de forma razonable. HRCH se basa en la fragmentación de rango y la fragmentación *hash*. Primero, se implementa la tecnología de *Consistent Hash*, y posteriormente la fragmentación de rango en cada nodo resultante de *Consistent Hash.* Se realizó la implementación de la estrategia HRCH en una base de datos NoSQL, posteriormente se realizaron experimentos para verificar la eficiencia y el rendimiento de la estrategia en la base de datos NoSQL. Los resultados mostraron que la estrategia puede mejorar la escalabilidad del sistema, el grado de procesamiento paralelo, la velocidad de carga de datos, y de igual forma, evita el problema de los puntos de alta demanda (*hot spot*). En conclusión, la estrategia es adecuada para fragmentar bases de datos NoSQL.

Las aplicaciones en la nube suelen depender de bases de datos distribuidas y compartidas que soporten un rápido crecimiento del volumen de datos. Las transacciones distribuidas (DTs, *Distributed Transactions)* que implican tuplas de datos de múltiples servidores pueden impactar negativamente en el rendimiento de las bases de datos, especialmente cuando las

transacciones son de corta duración y requieren una respuesta inmediata. El algoritmo de agrupación de grafos *k-way min-cut* resulta eficaz para reducir el número de estas transacciones, sin embargo, los beneficios de este esquema de fragmentación estática son efímeros en aplicaciones en la nube con patrones de carga de trabajo que varían dinámicamente y cuyo perfil de DT cambian con el tiempo. Kamal et al [9] presentaron un *framework* de re-fragmentación incremental proactivo y consciente de la carga de trabajo que redistribuye de forma transparente las tuplas de las bases de datos para garantizar las migraciones de datos al mínimo y un equilibrio de carga global. El *framework* está conformado por una técnica de clasificación de transacciones proactiva, dos estrategias de mapeo de *clúster* a fragmento y una técnica de búsqueda de datos distribuida y escalable Se realizó un experimento para medir la calidad del *framework* a través de métricas desarrolladas por los autores, aplicando el *framework* a una base de datos TPC-C y comparándolo con un *framework* de fragmentación estática. Los resultados del experimento revelaron que el *framework* minimiza el impacto de las transacciones distribuidas, el desequilibro de la carga global y las migraciones de datos físicos. Las métricas desarrolladas muestran una forma sofisticada para medir la calidad de la re-fragmentación sucesiva. Como trabajos a futuro se propone la agregación de replicación en un solo paso, aplicación de diferentes criterios de equilibrio en la heurística de mapeo de *clúster* a fragmento e incluir el costo de las migraciones de datos.

La escalabilidad de las aplicaciones OLTP (*On-Line Transactional Processing*, Procesamiento de Transacciones En Línea) dependen de la existencia de un óptimo diseño de base de datos. Este diseño define cómo los datos y la carga de trabajo de una aplicación se dividen o replican en los nodos de un clúster y cómo se enrutan las consultas y transacciones a los nodos. Por lo tanto, sin un diseño adecuado, un SGBD (Sistema Gestor de Bases de Datos) no funcionará mejor, debido a la sobre carga causada por el bloqueo de comunicación entre los nodos y los problemas de equilibrio de carga. En Pavlo et al. [10] presentaron un novedoso enfoque de fragmentación de bases de datos automático para sistemas OLTP de tipo empresarial que minimiza el número de transacciones distribuidas, incluye índices secundarios replicados, permite el manejo orgánico del enrutamiento de procedimientos almacenados y el escalamiento de la complejidad del esquema, tamaño de los nodos y el número de fragmentos. Los experimentos demostraron que estas opciones

tomadas en cuenta son importantes en los sistemas OLTP paralelos, y que el enfoque genera diseños de bases de datos que permiten mejorar el rendimiento hasta 16 veces más que otras soluciones.

Las transacciones distribuidas en redes basadas en TCP/IP de alta sobrecarga representan el cuello de botella dominante en los sistemas de bases de datos distribuidas, existen muchos esquemas con el objetivo de minimizar el número de transacciones entre fragmentos, sin embargo, con la nueva generación de redes rápidas, esta suposición ya no es válida. Se ha demostrado que las bases de datos distribuidas pueden escalar incluso cuando la mayoría de las transacciones son de fragmentación cruzada. Por ello en Zamanian et al. [11] presentaron su enfoque *Chiller* para la fragmentación de datos y la ejecución de transacciones, cuyo objetivo es minimizar la contención de datos en las transacciones locales y en las distribuidas. *Chiller* está basado en dos ideas complementarias: protocolo de confirmación basado en la reordenación de operaciones de transacciones para minimizar la duración de los bloqueos de los registros en disputa mediante la consignación temprana de dichos registros y una fragmentación consciente de la contención para que los registros más críticos sean actualizados sin coordinación adicional. *Chiller* fragmenta los datos de tal forma que los registros con más probabilidad que se accedan de forma conjunta se coloquen en la misma partición, los experimentos realizados por los autores mostraron que *Chiller* superó significativamente los enfoques existentes bajo cargas de trabajo con diferentes grados de contención.

La fragmentación, la asignación y la replicación son técnicas ampliamente utilizadas en las bases de datos relacionales para mejorar el rendimiento de las operaciones y reducir su costo en entornos distribuidos, sin embargo, en la mayoría de estos, no se incluyen los tres tópicos juntos. Los autores en [12] presentaron una aplicación web denominada FRAGMENT que adopta la técnica de trabajo seleccionada en la fase de análisis, ya que presenta un método de fragmentación y replicación, se aplica a un entorno de la nube, es de fácil implementación, se centra en mejorar el rendimiento de las operaciones ejecutadas en las bases de datos, muestra todo lo necesario para su implementación y se basa en un modelo de costos. FRAGMENT analiza las operaciones realizadas en cualquier tabla de una base de datos, propone esquemas de fragmentación basados en los atributos más costosos y asigna y replica

el esquema elegido por el usuario en un entorno distribuido en la nube, por medio de la aplicación web se realizan todas las operaciones requeridas para asignar y replicar fragmentos sobre un conjunto de sitios mediante el esquema que produce el método y atendiendo el problema de fragmentos traslapados. FRAGMENT desempeña una fragmentación estática, por lo cual como trabajo a futuro se propone agregar un modelo de fragmentación dinámica para que este se base en umbrales de costo para realizar fragmentaciones posteriores a la inicial.

Con el crecimiento de los dispositivos multimedia, grandes cantidades de datos multimedia son generados todos los días, el rápido acceso a estas grandes colecciones de datos significa cada vez más, grandes desafíos y la necesidad por algoritmos más eficientes. Por lo tanto, las características de las bases de datos multimedia, tales como la actualización de transacciones, facilidad de consultas y la indexación, se vuelven trascendentes cuando incrementa el número de objetos multimedia almacenados y tales grandes desafíos comienzan a aparecer. Por ello en Rodríguez-Arauz et al. [13] presentaron un sistema administrador de datos multimedia que utiliza métodos de fragmentación horizontal para optimizar las consultas basadas en contenido. Se realizó un experimento en donde se comparó el costo de ejecución de cuatro consultas basadas en contenido en una base de datos multimedia sin fragmentación y con fragmentación horizontal, usando un modelo de costos. Los resultados obtenidos revelaron que el costo de ejecución de cada consulta basada en contenido usando fragmentación horizontal es significativamente más bajo que cuando no se utiliza un método de fragmentación. Como trabajo a futuro se pretende la inclusión de un método de fragmentación vertical para mejorar la optimización de consultas basadas en contenido. Además de la inclusión de consultas complejas conjuntivas para la evaluación del esquema de fragmentación horizontal.

Los sistemas de gestión de bases de datos de procesamiento de transacciones en línea a menudo sirven cargas de trabajo que varían en el tiempo debido a las fluctuaciones diarias, semanales o estacionales de la demanda, debido al rápido crecimiento de esta. Para hacer frentes a estas fluctuaciones, un SGBD OLTP debe ser elástico; es decir, debe ser capaz de ampliar y reducir los recursos en respuesta a las fluctuaciones de carga y equilibrar dinámicamente la carga. En Taft et al. [14] presentaron *E-Store*, un *framework* de

fragmentación elástico para SGBDs OLTP distribuidos. Este *framework* escala automáticamente los recursos en respuesta a los picos de demanda, los eventos periódicos y los cambios graduales en la carga de trabajo de una aplicación. *E-Store* aborda los cuellos de botella localizados a través de una estrategia de colocación de datos en dos niveles: los datos menos accedidos se colocan en fragmentos grandes y los rangos más pequeños de tuplas accedidas frecuentemente se asignan explícitamente a nodos individuales. Se realizaron unos extensivos experimentos usando grandes conjuntos de datos y tres diferentes *benchmarks* para analizar la sensibilidad de los parámetros y el rendimiento de *E-Store*. Los resultados de los experimentos exhibieron que *E-Store* empieza a reconfigurar la base de datos después de 10 segundos de detectar alguna desviación de la carga o un pico de la carga, de igual forma permite aumentar el rendimiento hasta 4 veces y recudir la latencia hasta 10 veces.

Capturar y modelar la carga de trabajo transaccional durante un periodo de tiempo, y luego explotar esa información para la asignación y replicación de los datos, ha demostrado proporcionar beneficios significativos en el rendimiento, tanto en términos de latencia de las transacciones como el rendimiento general. Sin embargo, estos enfoques de asignación de datos con conciencia de la carga de trabajo pueden incurrir en gastos generales muy elevados y, además, pueden tener un peor rendimiento. En [15] los autores presentaron SWORD, un enfoque escalable de fragmentación y colocación de datos para cargas de trabajo OLTP, que incorpora un conjunto de técnicas novedosas para reducir significativamente los gastos generales incurridos durante la asignación inicial y durante la ejecución de la consulta. Los resultados obtenidos de los experimentos mostraron que el enfoque brinda un mecanismo flexible para determinar un punto ideal en términos de la relación de compresión del hipergrafo, proporcionando una calidad de fragmentación buena, una mejora en la eficiencia del enrutamiento y reducción sustancial de los costos de fragmentación. De igual forma se demostró que su técnica de fragmentación mitiga el impacto de los cambios de la carga de trabajo con un mínimo movimiento de datos y que el esquema de asignación de datos propuesto mejora naturalmente la resistencia a los fallos.

La principal forma de escalar las bases de datos para que funcionen en varias máquinas físicas son mediante la fragmentación horizontal. Colocando las particiones en diferentes nodos, a menudo es posible lograr una aceleración casi lineal. Además de mejorar la escalabilidad, la

fragmentación también permite mejorar la disponibilidad y aumentar la capacidad de gestión de las bases de datos. Aunque se han investigado varios esquemas de fragmentación automática, estos pueden ser eficaces para las consultas analíticas que exploran grandes conjuntos de datos, por desgracia, para cargas de trabajo que consisten en pequeñas transacciones que tocan unos pocos registros, ninguno de estos es ideal. Curino et al. [16] presentaron *Schism*, un novedoso sistema de fragmentación de datos basado en grafos para cargas de trabajo transaccionales. *Schism* representa una base de datos y su carga de trabajo mediante un grafo, en el que las tuplas se representan mediante nodos y las transacciones se representan mediante aristas que conectan las tuplas. Se realizaron experimentos sobre una variedad de cargas de trabajo derivadas de aplicaciones sintéticas y reales. Los resultados mostraron que *Schism* es muy práctico, demostró tiempos de ejecución modestos, excelente rendimiento de fragmentación, la capacidad de fragmentar una variedad de bases de datos OLTP y una facilidad de integración en las bases de datos existentes.

Para aumentar el rendimiento y minimizar la contención de recursos y datos, los sistemas de bases de datos distribuidas requieren la selección de un diseño físico distribuido que determine dónde colocar los datos y qué elementos de datos replicar y fragmentar. Decidir un diseño físico es difícil, ya que cada elección plantea un compromiso en el espacio de diseño, y una mala elección puede degradar significativamente el rendimiento. Las decisiones de diseños actuales suelen ser estáticas y no pueden adaptarse a los cambios de la carga de trabajo. Los autores en [17] presentaron *MorphoSys*, un sistema de bases de datos distribuidas que elige dinámicamente y altera su diseño físico en función de la carga de trabajo. *MorphoSys* toma decisiones de diseño integradas para todas las decisiones de fragmentación, replicación y asignación de datos sobre la marcha utilizando un modelo de costos. Se realizó una serie de experimentos utilizando la carga de trabajo de los siguientes benchmarks: YCSB, TPC-C, Twitter y de un pequeño banco, y comparándolo con otros enfoques tales como; *Clay*, *ADR*, *DynaMast*, *VoltDB*, entre otros, los resultados de estos experimentos mostraron que el sistema mejora el rendimiento 900 veces más sobre los otros enfoques y evita el uso de diseños estáticos que requieren información previa sobre la carga de trabajo.

Los optimizadores de consultas tradicionales eligen un plan de ejecución por consulta, basándose en estadísticas de primer orden sobre los datos subyacentes. Si embargo, las bases de datos del mundo real suelen contener datos sesgados con correlaciones complejas que dificultan la estimación de la selectividad de datos, y las estadísticas no son suficientemente potentes para capturar las propiedades estadísticas subyacentes de estos. Por ello, en [18] los autores presentaron un estudio de fragmentación horizontal durante el procesamiento de consultas con máxima compartición de resultados intermedios y un algoritmo voraz de baja sobrecarga que utiliza resúmenes estadísticos basados en modelos de gráficos. Se realizó una comparación del algoritmo de fragmentación horizontal voraz con el mejor plan monolítico mediante una enumeración exhaustiva de todos los planes posibles. Los resultados de la comparación sugieren que los tiempos de ejecución son más rápidos que la optimización tradicional para altas correlaciones, manteniendo el mismo rendimiento para las correlaciones bajas.

Los sistemas de bases de datos paralelas dividen horizontalmente grandes cantidades de datos estructurados con el fin de proporcionar capacidades de procesamiento de datos en paralelo para las cargas de trabajo analíticas en clústeres de uso compartido. Uno de los principales desafíos a la hora de fragmentar horizontalmente grandes cantidades de datos es reducir los costos de red para una carga de trabajo desterminada y un esquema de base de datos. Una técnica habitual es la compartición de tablas en función de su clave de unión, con el fin de evitar las costosas operaciones de unión remota. Sin embargo, los esquemas de fragmentación existentes están limitados en este sentido, ya que solo pueden coparticipar subconjuntos de tablas en esquema complejos que comparte la misma clave de unión. En Zamanian et al. [19] presentaron un esquema de fragmentación denominado partición de referencia basado en predicados que permite co-fragmentar conjuntos de tablas basándose en predicados de unión dados. Además, propusieron dos algoritmos de diseño de fragmentación automática para maximizar la localización de los datos. Se realizaron experimentos utilizando el *benchmark* TPC-H y tomando como referencia TPC-DS, se comparó el esquema con diferentes variantes de fragmentación. Los resultados de los experimentos demostraron que los algoritmos de diseño automatizado pueden fragmentar esquemas de bases de datos de diferente complejidad y, por lo tanto, reducen eficazmente el tiempo de ejecución de las consultas bajo una carga de trabajo.

Los sistemas de gestión de datos distribuidos suelen funcionar en *clúster* "elásticos" que pueden ampliarse o reducirse según la demanda. Estos sistemas se enfrentan a numerosos retos, como la fragmentación de datos, la replicación y el tamaño del *clúster*. Sin embargo, estos retos se han tratado de forma independiente, dejando a los administradores con poca idea de cómo la interacción de estas decisiones afecta el rendimiento de las consultas. Por ello, Marcus et al. [20] presentaron NashDB, un *framework* de distribución de datos que se basa en un modelo económico para equilibrar automáticamente la oferta y la demanda de fragmentos de datos, réplicas y nodos de *clúster*. NashDB adapta sus decisiones a las prioridades de las consultas y a los cambios en las cargas de trabajo, al tiempo que evita los nodos de *clúster* infrautilizados y las réplicas redundantes. En los experimentos para evaluar el *framework* se utilizaron dos diferentes tipos de cargas de trabajo de muchos conjuntos de datos. Se evaluó la capacidad de búsqueda de fragmentos con baja varianza, priorización de consultas, costos y latencia. Los resultados obtenidos de los experimentos revelaron que la implementación prototípica de NashDB muestra un rendimiento dominante en Pareto (en términos de costo de uso de *clúster* y latencia de ejecución de las consultas) en una serie de cargas de trabajo sintéticas y del mundo real.

Las técnicas de fragmentación horizontal se han utilizado en bases de datos multimedia para mejorar los costos de ejecución de consultas. Sin embargo, existen técnicas que se basan en algoritmos sobre la afinidad entre predicados para obtener un esquema de fragmentación horizontal. La afinidad mide cómo se accede a un par de predicados mediante las consultas (*togetherness*). La principal desventaja de esta medida es que solo involucra dos predicados y no muestra la unión de más de dos predicados. En Rodríguez-Mazahua et al. [21] presentaron un método de fragmentación horizontal para bases de datos multimedia basado en un algoritmo de agrupación jerárquica aglomerativa. La principal ventaja de este método es que no usa la afinidad para crear el esquema de fragmentación. El enfoque aglomerativo o también llamado *bottom-up* comienza formando un grupo separado por cada objeto, posteriormente mezcla objetos o grupos que estén más cercanos unos de otros, hasta que todos los grupos estén mezclados en un grupo o hasta que se obtengan las condiciones de conclusión. En este trabajo se describió el algoritmo mediante un ejemplo de administración de equipos en una compañía de ventas de maquinaria y se presentó detalladamente el modelo de costos. Se llevó a cabo la evaluación del algoritmo y se demostró que supera el

rendimiento para crear el esquema de fragmentación en la mayoría de los casos. Como trabajo a futuro se mencionó que se desarrollará un algoritmo de fragmentación híbrida para bases de datos multimedia tomando en cuenta consultas basadas en contenido.

2.2. Análisis comparativo

En la Tabla 2.1 se muestra un análisis comparativo de los trabajos anteriormente descritos, para que se observen las diferencias y las similitudes entre ellos de una mejor manera.

Tabla 2.1. Análisis comparativo de los trabajos relacionados

Artículo	Problema	Contribución	Tecnologías	Resultados
[4]	Las dificultades que enfrenta una fragmentación de datos al no considerar las relaciones de los datos en una base de datos relacional.	Metodo de fragmentación horizontal basado en las relaciones de datos y que incorpora una técnica de *clustering* que ayuda a encontrar dichas relaciones de datos.	PostgreSQL	Los resultados mostraron una mejora considerable del procesamiento de consultas distribuidas en la fragmentación basada en *clustering* que, en la fragmentación normal. De igual forma, se presentó el algoritmo que realiza fragmentación basada en *clustering.*
[5]	Dificultades que enfrentan las bases de datos relacionales ante el procesamiento de grandes cantidades datos no estructurados.	Análisis comparativo entre MySQL y MongoDB	MongoDB y MySQL	El análisis realizado obtuvo que el tiempo que tardan en realizar operaciones básicas de inserción y recuperación son sustancialmente menores en un sistema con MongoDB como base de datos, demostrando que MongoDB es más rápido que MySQL en carga de datos de forma dinámica.

Artículo	Problema	Contribución	Tecnologías	Resultados
[6]	Diferentes requisitos de consulta para implementar múltiples planes de fragmentación y los conflictos de múltiples planes de fragmentación'	Metodo que identifica y resuelve conflictos por réplicas de fragmentos con diferentes planes de fragmentación y una arquitectura de dos niveles para DBMS	MySQL	Los resultados obtenidos revelaron que el método aumenta la proporción de las transacciones en un solo sitio y la velocidad de repuesta de transacciones, gracias a la implementación de múltiples planes de fragmentación independientes en cada réplica.
[7]	Las desventajas y limitaciones de las fragmentaciones tradicionales ante el proceso de consultas de s*tream* en los entornos de la nube.	Algoritmo que maneja las consultas de *stream* y un enfoque que combina un *streaming framework* y algoritmo de fragmentación.	No se menciona	Los resultados de las simulaciones mostraron que WSPS es más rápido que los enfoques anteriores y hasta produce resultados de fragmentación de buena calidad. Además, se obtuvieron resultados superiores a los enfoques existentes sobre cargas de trabajo cambiantes.
[8]	Baja escalabilidad, puntos de alta demanda (*hot spot*) y bajo rendimiento que pueden provocar las estrategias de fragmentación	Análisis a los desafíos que se enfrenta la fragmentación de datos, una nueva estrategia de fragmentación HRCH y modificación de YCSB para realizar pruebas de rendimiento a HRCH	HBase	Los resultados obtenidos demostraron que la estrategia mejora la escalabilidad del sistema, el grado de procesamiento en paralelo, la velocidad de carga de datos y evita los puntos de alta demanda.
[9]	El impacto negativo de transacciones distribuidas en el rendimiento de las bases de datos y la ineficiencia de los esquemas de fragmentación estática en las aplicaciones de la nube	Técnica de clasificación de transacciones proactiva, dos estrategias de mapeo de clúster a fragmento, un conjunto de métricas de calidad y una técnica de búsqueda de datos distribuida y escalable	No se menciona	El *framework* minimiza el impacto de las transacciones distribuidas, el desequilibrio de la carga global y las migraciones de datos físicos.

Artículo	Problema	Contribución	Tecnologías	Resultados
[10]	Diseños de bases de datos que provocan bloqueos de comunicación entre los nodos y los problemas de equilibrio de carga.	Algoritmo de fragmentación automático de bases de datos basado en una adaptación de la técnica de búsqueda *large-neighborhood* y un nuevo modelo analítico de costos que estima el costo de coordinación y la distribución de la carga para una muestra de carga de trabajo.	H-Store	Los experimentos realizados revelaron que el enfoque genera diseños de bases de datos que mejoran el rendimiento en los SGBD hasta 16 veces más que otras soluciones.
[11]	Las dificultades que generan las transacciones distribuidas en los sistemas gestores de bases de datos distribuidas.	Esquema de fragmentación centrado en la contención de datos y una nueva técnica de ejecución de transacciones distribuidas, cuyo objetivo es actualizar los registros de alta contención sin coordinación adicional.	No se menciona	Lo resultados obtenidos en los experimentos mostraron que el enfoque supera a soluciones existentes bajo cargas de trabajo con diferentes grados de contención y fragmenta los datos de forma que los registros más recurrentes estén en el mismo fragmento.
[12]	El problema de traslape de fragmentos y la carencia de técnicas y/o métodos que no incluyen los tópicos de fragmentación, asignación y replicación.	Una aplicación web denomina *FRAGMENT* para realizar la fragmentación, asignación y replicación de una manera fácil y un algoritmo con un enfoque para solucionar el traslape de los fragmentos.	MySQL	Aplicación web que implementa un método de fragmentación, asignación y replicación que atiende el problema de fragmentos traslapados, generando esquemas adecuados con fragmentos disjuntos y completos.
[13]	Los desafíos de administración que genera el incremento del número de objetos multimedia almacenados en las bases de datos de contenido multimedia.	Diseño de un sistema para administrar datos multimedia y un método de fragmentación horizontal que mejora el costo de ejecución de consultas basadas en contenido.	MongoDB	Los resultados del experimento realizado revelaron que el costo de la ejecución de consultas basadas en contenido usando fragmentación horizontal es menor que cuando no se utiliza ningún método.

Artículo	Problema	Contribución	Tecnologías	Resultados
[14]	Las dificultades que se enfrentan los SGBDs OLTP a las variaciones de las cargas de trabajo y las fluctuaciones de estas.	*Framework* que consiste en dos sub sistemas, *E-Monitor* y *E-Planner*, el desarrollo de heurísticas inteligentes para realizar decisiones inteligentes	*H-Store*	Los resultados de los experimentos mostraron que *E-Store* empieza a reconfigurar la base de datos después de 10 segundos de detectar una desviación de la carga o un pico de carga, de igual forma, la reconfiguración permite aumentar el rendimiento hasta 4 veces y reducir la latencia hasta 10 veces más.
[15]	Enfoques de asignación que pueden incurrir en gastos generales muy elevados y provocar un peor rendimiento al sistema.	Modelado y compresiones eficaces de la carga de trabajo que reducen los gastos generales de fragmentación, re-fragmentación incremental para mitigar la degradación del rendimiento debido a cambios en la carga de trabajo, implementación de quorums para el control de costo de actualizaciones, mecanismo de replicación consciente de la carga de trabajo y un mecanismo de enrutamiento eficiente y escalable que minimiza el número de fragmentos involucrados en una consulta.	PostgreSQL	El enfoque proporcionó un mecanismo flexible para determinar el punto ideal para la compresión de hipergrafos, que permite una fragmentación buena, una mejora de eficiencia y reducción de costos, de igual forma la técnica de fragmentación mostró una mitigación al impacto provocado por los cambios de la carga de trabajo y demostró que el esquema de asignación de datos propuesto mejoró naturalmente la resistencia a los fallos.
[16]	La carencia de esquemas que manejen cargas de trabajo que consisten en pequeñas transacciones que acceden pocos registros.	Esquema basado en árboles de decisión para identificar predicados (rangos), heurística que incluye el muestreo y la agrupación de tuplas, para limitar el tamaño del grafo y reducirlo en tiempo de fragmentación,	MySQL	Los resultados de los experimentos mostraron que *Schism* es muy práctico demostrando tiempos de ejecución modestos, excelente rendimiento de fragmentación, la capacidad de fragmentar una variedad de bases de datos OLTP y una facilidad de integración en las bases de datos existentes.

Artículo	Problema	Contribución	Tecnologías	Resultados
[17]	Las dificultades de decidir un diseño físico sobre un sistema gestor de bases de datos distribuidas.	Una arquitectura para un sistema de base de datos distribuido, un modelo de costos que dirige las decisiones del diseño físico, un nuevo algoritmo de control de concurrencia y un protocolo de propagación de actualizaciones para apoyar la ejecución.	No se menciona	Los resultados de los experimentos revelaron que el sistema mejora el rendimiento 900 veces más sobre otros enfoques y evita el uso de diseños estáticos que requieren información previa sobre la carga de trabajo.
[18]	La dificultad de la selectividad que generan las bases de datos al contener datos sesgados con correlaciones complejas.	Estudio más formal del problema general, introducción a la noción de planes de unión condicionales, una representación del espacio de la búsqueda resultante de la fragmentación horizontal y la definición de fórmulas de costo recursivo para los planes de unión condicional.	PostgreSQL	Los resultados del experimento sugieren que los tiempos de ejecución son más rápidos que la optimización tradicional para altas correlaciones, manteniendo el mismo rendimiento para correlaciones bajas.
[19]	Los desafíos de reducir los costos de red para una carga de trabajo y la limitación de los esquemas de fragmentación al compartir tablas.	Esquema de fragmentación basado en predicados para cargas de trabajos analíticas en las que los datos se cargan en grandes cantidades.	MySQL	Los resultados de los experimentos demostraron que el algoritmo de diseño basado en la carga de trabajo es más eficiente para esquemas complejos con un mayor número de tablas.
[20]	Las dificultades que se presentan en sistemas de datos distribuidos al realizar técnicas de fragmentación y replicación. La carencia de métodos que realicen juntas las técnicas de fragmentación y replicación.	Algoritmo de fragmentación de datos que tiene en cuenta los patrones de acceso a los datos y las prioridades de consulta, algoritmo de replicación que equilibra el suministro de réplicas con las demandas de carga de trabajo, algoritmo para la transición entre dos esquemas diferentes de distribución de datos y una estrategia consciente de latencia para enrutar las solicitudes de acceso.	PostgreSQL	Los resultados obtenidos de los experimentos exhibieron un aumento del rendimiento (en términos de costo de uso de *clúster* y latencia de ejecución de consultas).

Artículo	Problema	Contribución	Tecnologías	Resultados
[21]	Las desventajas de utilizar métodos de fragmentación horizontal que se basan en algoritmos de afinidad.	Una técnica de fragmentación horizontal para bases de datos multimedia el cual está basado en un algoritmo de agrupación jerárquica aglomerativa.	No se menciona	Los resultados obtenidos de los experimentos demostraron que superan el rendimiento para crear esquemas de fragmentación.

Debido a la creciente importancia de los datos multimedia en diversas aplicaciones, se tiene un crecimiento rápido y constante en las bases de datos multimedia, lo cual provoca un aumento de costos en transmisión de datos y tiempo de respuesta de las consultas.

Con el análisis hecho a los artículos relacionados con este tema de tesis, se llega a la conclusión de que, aunque todos los trabajos reportan el uso de técnicas de fragmentación horizontal, la mayoría no consideran datos multimedia. Los que sí toman en cuenta estos datos, llevan a cabo una fragmentación estática [13], [21]. Por tal motivo, en este proyecto se desarrolló un método que permite fragmentar horizontalmente bases de datos multimedia y que adapta el esquema de fragmentación de acuerdo con los cambios en los patrones de acceso logrando una reducción de los costos de ejecución y tiempos de respuesta de las consultas.

2.3. Propuesta de solución

Para llegar a un resultado satisfactorio, se proponen una serie de pasos que dividen el trabajo en etapas, las cuales están sujetas a la metodología de desarrollo elegida, como se muestra en la Figura 2.1.

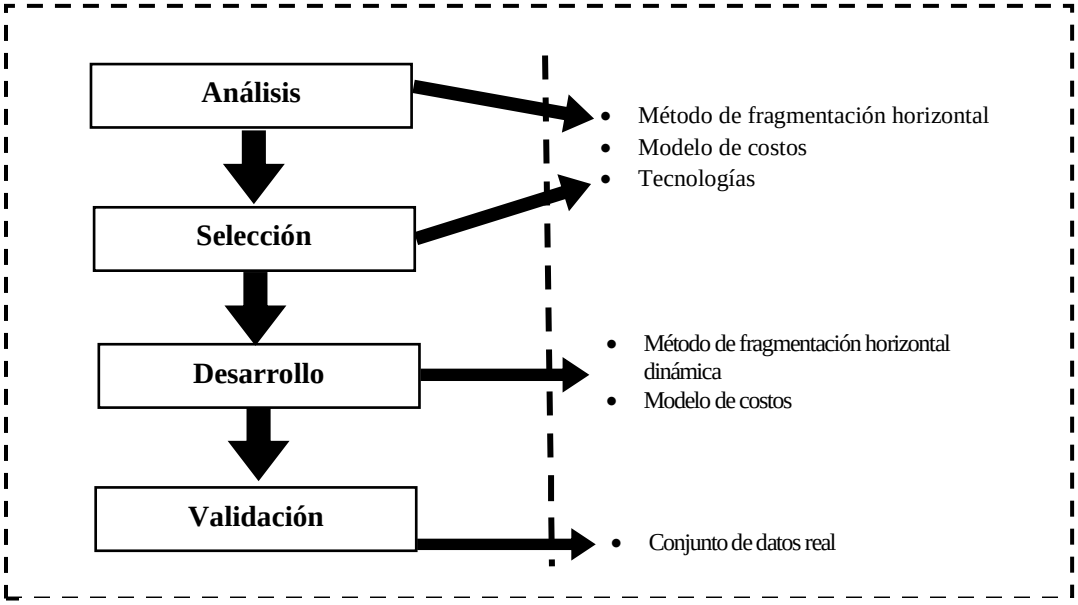


Figura 2.1 estructura de la propuesta de la solución

En la Figura 2.1 se observan los pasos propuestos para realizar el análisis comparativo, que se describe a continuación:

1. **Análisis:** en esta etapa del trabajo se analizaron diferentes artículos por medio de la búsqueda en las principales bibliotecas digitales como ACM Digital Library, SpringerLink, SciencieDirect y IEEE Xplore, donde se encontraron los métodos de fragmentación horizontal, modelos de costos y tecnologías empleadas para su desarrollo.
2. **Selección:** En esta etapa se seleccionó un método de fragmentación horizontal y un modelo de costos para compararlos con el método y modelo desarrollados, también se determinaron las tecnologías de los trabajos anteriormente analizados, la técnica elegida fue aquella que cumplió con la mayoría de los criterios de comparación: tener una solución completa, que considere la fragmentación horizontal dinámica que contenga un modelo de costos, que tenga facilidad de implementación y, por último, que se enfoque en mejorar el desempeño de las consultas.
3. **Desarrollo:** En esta etapa se desarrolló el método de fragmentación horizontal dinámico y el modelo de costos utilizando las tecnologías seleccionadas en la fase anterior.

4. **Validación:** En esta etapa se comparó el método de fragmentación y el modelo de costos desarrollado con el trabajo seleccionado del análisis comparativo por medio de un conjunto de datos reales. De esta manera se comprobó la efectividad de la solución propuesta.

A continuación, se muestra la Tabla 2.2, que presenta la alternativa de solución.

Tabla 2.2 Alternativa de solución

Aspecto	Propuesta
Marco de Trabajo	**JSF**
IDE	**NetBeans**
Metodología	**UWE**
SGBD	**MongoDB y MySQL**
Lenguaje de programación	**Java**

Diseñada para ser flexible, la tecnología de JavaServer Faces aprovecha la interfaz de usuario estándar existente y los conceptos de nivel web sin limitar a los desarrolladores a un lenguaje de marcado, protocolo o dispositivo cliente en particular. Las clases de componentes de la interfaz de usuario incluidas con la tecnología de JavaServer Faces encapsulan la funcionalidad del componente, no la presentación especifica del cliente, lo que permite que los componentes de la interfaz de usuario de JavaServer Faces se representen en varios dispositivos cliente. Al combinar la funcionalidad del componente de la interfaz de usuario con los visualizadores personalizados, que definen los atributos de representación para un componente de interfaz de usuario específico, los desarrolladores construyen etiquetas personalizadas en un dispositivo cliente particular. Para su comodidad, la tecnología JavaServer Faces proporciona un renderizado personalizado y una biblioteca de etiquetas personalizadas JSP para renderizar en un cliente HTML, permitiendo a los desarrolladores de aplicaciones Java, Enterprise Edition (Java EE), utilizar la tecnología JavaServer Faces en sus aplicaciones [22].

NetBeans es un entorno de desarrollo integrado de código abierto. Proporciona modularidad al código, ya que admite un enfoque modular, es decir, permite que las aplicaciones se

desarrollen como módulos (como componente de un software). La plataforma de NetBeans es básicamente un *framework* que simplifica el desarrollo de aplicaciones de escritorio Java. Es capaz de instalar módulos de forma dinámica. Además de Java, también admite otros lenguajes, incluido PHP, C, C++ y HTML 5. NetBeans IDE es el IDE oficial para Java 9. Con sus editores, analizadores de código y conversores, es posible actualizar las aplicaciones de forma rápida y sin problemas para usar nuevas construcciones del lenguaje de Java, como lambdas, operaciones funcionales y referencias de métodos. Con su editor Java que mejora constantemente, muchas funciones completas y una amplia gama de herramientas, plantillas y muestras, NetBeans IDE establece el estándar para el desarrollo con tecnologías de vanguardia listas para usar [23].

UWE (*Unified Modeling Language Web Engineering*, Ingeniería Web del Lenguaje Unificado de Modelado) es una metodología que permite especificar de mejor manera una aplicación Web en su proceso de creación, mantiene una notación estándar basada en el uso de UML (*Unified Modeling Language*) para sus modelos y sus métodos. La metodología define claramente la construcción de cada uno de los elementos del modelo.

En su implementación se contemplan las siguientes etapas y modelos:

- Análisis de requisitos. Plasma los requisitos funcionales de la aplicación Web mediante un modelo de casos de uso.
- Modelo de contenido. Define, mediante un diagrama de clases, los conceptos a detalle involucrados en la aplicación.
- Modelo de navegación. Representa la navegación de los objetos dentro de la aplicación y un conjunto de estructuras como son índices, menús y consultas.
- Modelo de presentación. Representa las interfaces de usuario por medio de vistas abstractas.
- Modelo de proceso. Representa el aspecto que tienen las actividades que se conectan con cada clase de proceso.

Como se hace notar, UWE provee diferentes modelos que permiten describir una aplicación Web desde varios puntos de vista abstractos. Cada uno de estos modelos se representa como

paquetes UML, dichos paquetes son procesos relacionados que se refinan en iteraciones sucesivas durante el desarrollo del UWE [24].

MongoDB es una base de datos distribuida de propósito general, basada en documentos, creada para desarrolladores de aplicaciones modernas y para la era de nube, con grandes posibilidades de escalabilidad y flexibilidad con las consultas e indexación que sea necesaria. El modelo de documentos de MongoDB es simple de aprender y usar para los desarrolladores, al mismo tiempo proporciona todas las capacidades necesarias para cumplir con los requisitos más complejos a cualquier escala. MongoDB es una base de datos distribuida en su núcleo, por lo que la disponibilidad y el escalamiento horizontal y la distribución geográfica se encuentran integradas y con facilidad de usar [25].

MySQL es la base de datos de código abierto más popular del mundo. Con su rendimiento comprobado, fiabilidad y facilidad de uso, MySQL se ha convertido en la principal opción de base de datos para aplicaciones basadas en Web, utilizada por propiedades Web de alto perfil, como Facebook, Twitter, YouTube, entre otros. Además, es una opción extremadamente popular como base de datos integrada, distribuida por miles de ISV (*Independent software vendor*, Vendedor Independiente Software) y OEM (*Original equipment manufacturer*, fabricante de equipos originales) [26].

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems, Java es una tecnología que se usa para el desarrollo de aplicaciones que convierte a la Web en un elemento más interesante y útil. Java es la base para prácticamente todos los tipos de red, además de un estándar global para desarrollar y distribuir aplicaciones móviles y embebidas, juegos, contenido basado en la web y software de empresa. Java está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible [27].

Capítulo 3. Aplicación de la metodología

Para llevar a cabo la aplicación de la metodología este capítulo se divide en dos partes. La primera realiza una búsqueda y un análisis profundo de los métodos de fragmentación horizontal para seleccionar la técnica que cumpla con ciertos criterios. La segunda describe el diseño de la aplicación Web para realizar la ejecución de la técnica propuesta. En este capítulo se aborda también la especificación de la técnica desarrollada y la manera en la que se implementó la fragmentación horizontal.

3.1 Análisis

Conforme el objetivo general y específicos, se realizó un análisis profundo de los trabajos relacionados con la fragmentación horizontal. Se llevó a cabo una búsqueda dentro de este análisis en algunas de las principales bibliotecas digitales de editoriales científicas: ACM, IEEE, Springer y Elsevier.

Como resultado del análisis, bajo algunos criterios principales, se obtuvo un método de fragmentación horizontal más adecuado para aplicar en este trabajo. Para realizar el estudio de los trabajos relacionados se siguió la metodología descrita a continuación en la Figura 3.1.

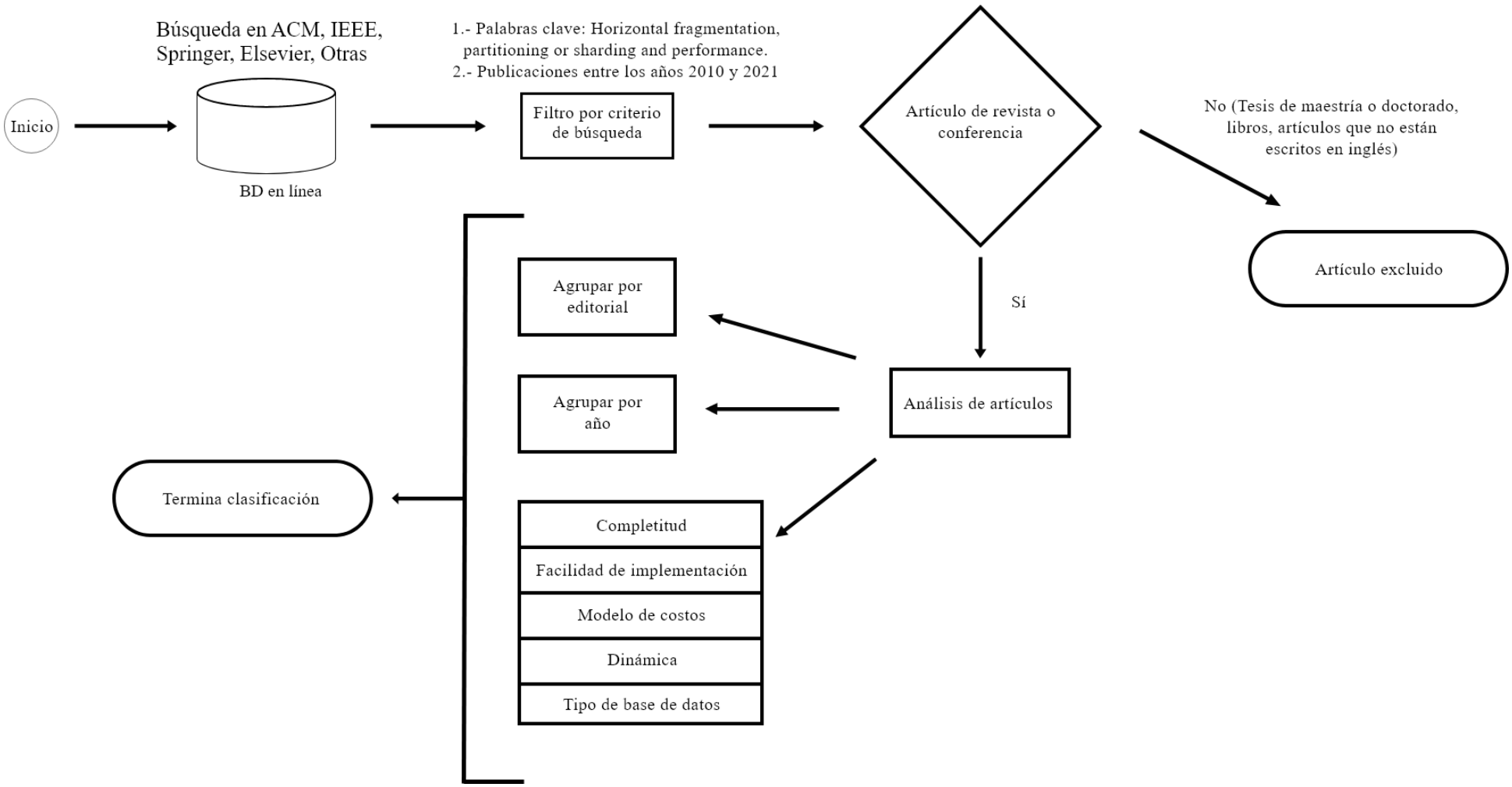


Figura 3.1 Metodología de búsqueda y evaluación de los trabajos relacionados

En la Figura 3.1 se muestran todas las etapas de la metodología propuesta. Como ya se mencionó, se realizó la búsqueda de trabajos en las principales bibliotecas digitales de editoriales científicas, ACM, IEEE, Springer y Elsevier. Los trabajos encontrados que no son publicados por dichas editoriales se categorizan en "Otras". La búsqueda consiste en encontrar todos los trabajos que contengan las siguientes palabras clave: *Horizontal Fragmentation* (Fragmentación horizontal), *partitioning (partición)* o *sharding* (fragmentación) y *performance* (desempeño). Los trabajos deben haberse publicado entre los años 2010 y 2021. Ya obtenidos todos los trabajos, se aplicó un filtro y se descartaron todos los trabajos que sean tesis de maestría o doctorado, así como libros y artículos que no estén escritos en inglés. Los artículos resultantes se clasificaron por editorial y por año. En esta última etapa se analizó cada uno de ellos bajo cinco rubros principales: Completitud, Facilidad de implementación, Modelos de costos, Fragmentación dinámica y Tipo base de datos. De esta manera, solo los trabajos que cumplieron con estos cinco rubros se seleccionaron.

El proceso de selección dio 46 artículos, los cuales fueron analizados por las cinco características, para registrar si cumplieron cada rubro. Las Tablas 3.1-3.5 describen el registro de cada artículo y el análisis de cada característica.

Tabla 3.1 Comparación de los trabajos relacionados de la editorial ACM

Artículo	Completitud	Facilidad de implementación	Modelo de costos	Dinámica	Tipo base de datos
Serafini et al. [28]	X	X	X	X	OLTP
Liming et al. [6]	X		X		Distribuida
Chen et al. [8]	X	X		X	NoSQL
Curino et al. [16]		X		X	OLTP
Tzoumas et al. [18]	X		X		Relacional
Taft et al. [14]		X		X	OLTP
Pavlo et al. [10]	X	X	X	X	OLTP
Quamar et al. [15]	X	X		X	Distribuidas en la nube
Zamanian et al. [19]	X	X		X	En paralelo
Abebe et al. [17]	X	X	X	X	Distribuida
Zamanian et al. [11]	X	X			Distribuida

Artículo	Completitud	Facilidad de implementación	Modelo de costos	Dinámica	Tipo base de datos
Marcus et al. [20]	X		X	X	Distribuida

Tabla 3.2 Comparación de los trabajos relacionados de la editorial IEEE

Artículo	Completitud	Facilidad de implementación	Modelo de costos	Dinámica	Tipo base de datos
Abdalla et al. [29]		X	X	X	Distribuida en la nube
Fasolin et al. [30]					Multimedia
Lim [31]	X	X	X		Distribuida en la nube
Herrmann et al. [32]	X	X	X		Distribuida
Kumar et al. [33]				X	Distribuida
Fetai et al. [34]			X	X	Distribuida
Sauer et al. [35]		X			NoSQL
Wu et al. [36]		X			NoSQL/Multimedia
Oonhawat et al. [37]	X	X			NoSQL
Lwin et al. [38]	X		X	X	Distribuida
Ramachandran et al. [4]	X		X		Relacional
Patil et al. [5]				X	NoSQL/Relacional
Guo et al. [7]	X	X		X	Distribuida en la nube
Kamal et al. [9]	X			X	OLTP
Peng et al. [39]	X		X	X	RDF
Castro-Medina et al. [12]	X	X	X		Distribuida en la nube

Tabla 3.3 Comparación de los trabajos relacionados de la editorial Elsevier

Artículo	Completitud	Facilidad de implementación	Modelo de costos	Dinámica	Tipo base de datos
Amer et al. [40]	X		X	X	Distribuida en la nube
Abdalla [41]	X	X	X	X	Relacional
Goli-Malekabadi et al. [42]		X			NoSQL
Wedashwara et al. [43]		X			Distribuida

Tabla 3.4 Comparación de los trabajos relacionados de la editorial Springer

Artículo	Completitud	Facilidad de implementación	Modelo de costos	Dinámica	Tipo base de datos
Hauglid et al. [44]	X	X	X	X	Distribuida en la nube
Liroz-Gistau et al. [45]	X	X		X	Relacional
Liroz-Gistau et al. [46]	X	X		X	Relacional
Rodríguez-Mazahua et al. [21]	X	X	X		Multimedia
Abdel et al. [47]	X	X	X	X	Distribuida en la nube
Elghamrawy et al. [48]	X	X			NoSQL
Elghamrawy et al. [49]		X	X		Relacional
Olma et al. [50]	X		X	X	Relacional
Salmi et al. [51]	X	X			Relacional
Feinerer et al. [52]	X	X			Relacional
Teng et al. [53]		X			NoSQL

Tabla 3.5 Comparación de los trabajos relacionados de otras editoriales

Artículo	Completitud	Facilidad de implementación	Modelo de costos	Dinámica	Tipo base de datos
Baron et al. [54]		X	X	X	Relacional
Khan [55]		X	X		Relacional
Rodríguez Arauz et al. [13]		X	X		Multimedia/NoSQL

Se observa en la Tablas 3.1 a 3.5 la comparación de todos los trabajos encontrados en las principales bibliotecas científicas digitales, que abordaron la mayoría de los temas de interés. Cada artículo se evaluó utilizando una metodología para determinar si cada artículo cumple con cada una de las características. De todos los trabajos analizados de observa que pocos artículos cumplen con todas las cualidades deseadas. Los artículos [10], [17], [28], [41] y [44] cumplen con la mayoría de las características, sin embargo, ninguno considera bases de datos multimedia. Por lo tanto, se tomaron en cuenta los siguientes tres artículos [12], [21] y [47]. En [12] la metodología se basa en un modelo de costos que permite determinar el atributo más costoso, de un conjunto de operaciones sobre una base de datos, la metodología

se encuentra completa y fácil de implementar, realiza una fragmentación estática y no contempla bases de datos de contenido multimedia. En [21] la metodología sí contempla bases de datos multimedia, utiliza un modelo de costos y se implementa de forma estática. En [47] la metodología se basa en un modelo de costos, y realiza la fragmentación de forma dinámica pero no contempla bases de datos de contenido multimedia. Para comparar la eficiencia de la metodología, se eligió el trabajo de Castro-Medina et al. debido a que es una aplicación fácil de implementar, está completa, utiliza un modelo de costos y utiliza una aplicación web para implementar los esquemas resultantes de su análisis, como en esta propuesta de tesis.

3.2 Selección

Los tres artículos ([12], [21] y [47]) involucran un modelo, son fáciles de implementar y están completos, se evaluaron y dos de ellos fueron descartados. En [47] se tiene el problema de traslape de fragmentos. En [21] sí se consideran las bases de datos multimedia, pero la forma en que realiza la fragmentación con un enfoque de agrupamiento jerárquico no es el más adecuado para la metodología propuesta.

El artículo elegido [12] no considera bases de datos multimedia y no realiza fragmentación dinámica, pero tiene un modelo de costos que se basan en los costos de creación, lectura, actualización y eliminación de datos. De igual forma, resuelve el problema de traslapes y considera la facilidad de implementación a través de una aplicación web.

La Figura 3.2 muestra el flujo de trabajo que consiste en 9 pasos, inicia con la configuración de la conexión de la base de datos; en el segundo paso, se definen los umbrales de operaciones y desempeño; en el paso tres, se carga el *log* de la base de datos para realizar un análisis de todas operaciones hechas; a partir de esto, en el paso 4, se crea la matriz de costo de operaciones basada en un modelo de costos multimedia; posteriormente en el paso 5 se construye la tabla de precedencias de atributos (ALP), a partir de la tabla ALP; en el paso 6 se crea un esquema inicial para la base de datos a fragmentar; en el paso 7 se aplica el esquema, en el paso ocho se genera el vigilante de fragmentación (VF), en este paso, el administrador de base de datos (DBA) puede terminar el proceso y dejar la parte estática de la metodología, para implementar la parte dinámica, es necesario realizar el paso 9, que

consiste en llevar a cabo un análisis permanente de *logs*, a través de la ejecución del VF; el cual internamente 1) estará evaluando su ejecución (hasta que el DBA crea necesario), 2) realizará cálculos para verificar que no se supere el umbral de operaciones ingresado por el DBA, si este es superado, entonces 3) realizará un análisis del costo del esquema actual, y lo comparará con el umbral de desempeño, si este es superado, 4) volverá a fragmentar la base de datos y asignar los fragmentos, posteriormente 5) creará vigilantes de fragmentación en los nodos secundarios, este VF estará realizando todas estas operaciones hasta que el DBA crea necesario.

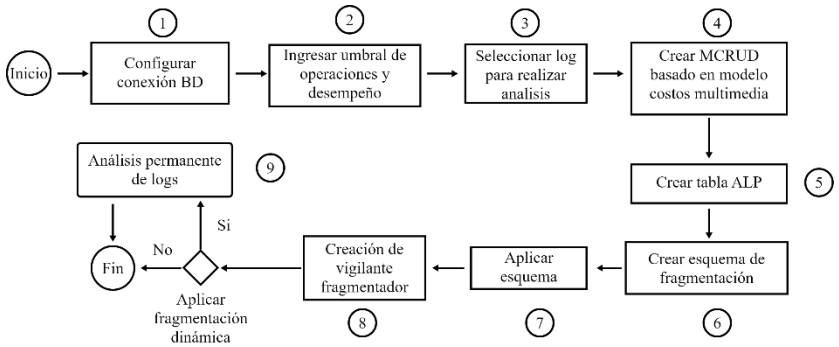


Figura 3.2 Flujo de trabajo

3.3 Desarrollo

En esta sección se describe el diseño de la aplicación siguiendo la metodología de desarrollo. Se llevan a cabo todos los diagramas y pasos de la metodología elegida para el desarrollo de la aplicación web.

La Figura 3.3 representa la arquitectura elegida la cual es “Modelo Vista Controlador” (MVC): en la vista se encuentran las páginas que interactúan con el usuario para solicitar información sobre las bases de datos y el acceso a ellas. Los *beans* administrados de JSF gestionan el flujo de datos que están en el controlador. En el modelo, se definen las reglas de negocio, que en este caso concreto será la aplicación de la fragmentación horizontal dinámica, además, de contemplar el cálculo de costos y la generación del vigilante de fragmentación para un gestor de contenido multimedia.

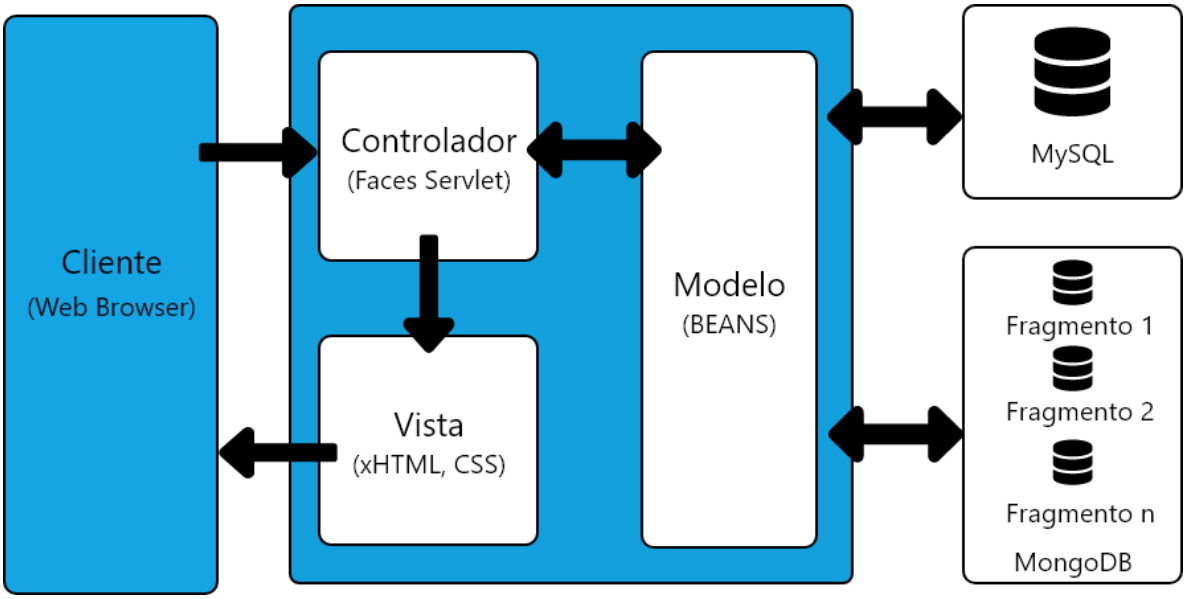


Figura 3.3 Arquitectura de la aplicación web para realizar la fragmentación horizontal dinámica.

3.3.1 Análisis de requisitos

El objetivo del análisis de requisitos es establecer los requisitos funcionales de la aplicación web, estos requisitos se representan mediante diagramas. El diagrama de casos de uso representa la relación actor-actividades, muestra las actividades a las que está relacionado cada actor.

En la Tabla 3.6 se definen los actores de la aplicación y una breve descripción del papel que desempeñan.

Tabla 3.6 Actores para la fragmentación horizontal dinámica

Actor	Descripción
Administrador de la base de datos (DBA)	El DBA tendrá la función de solicitar la fragmentación proporcionando la información necesaria y los archivos correspondientes para el proceso.
Vigilante fragmentador	El vigilante fragmentador tendrá la función, a través de la información ingresada por el DBA, de realizar el proceso de manera automática, ejecutándose en el servidor en que se encuentre la base de datos a fragmentar.

Una vez ya definidos los actores en la Tabla 3.6, en la Figura 3.4 se muestra el diagrama de casos de uso, el cual relaciona a los actores con sus actividades. Como se observa, solo están relacionados con una actividad.

Figura 3.4 Diagrama de casos de uso para la aplicación web y la fragmentación horizontal dinámica

En la Figura 3.4 se muestra el caso de uso "Fragmentar y asignar" con el estereotipo *Process* (Proceso), como lo sugiere la metodología UWE. Este caso de uso se describe con un diagrama de actividades. Se observa tal diagrama en la Figura 3.5

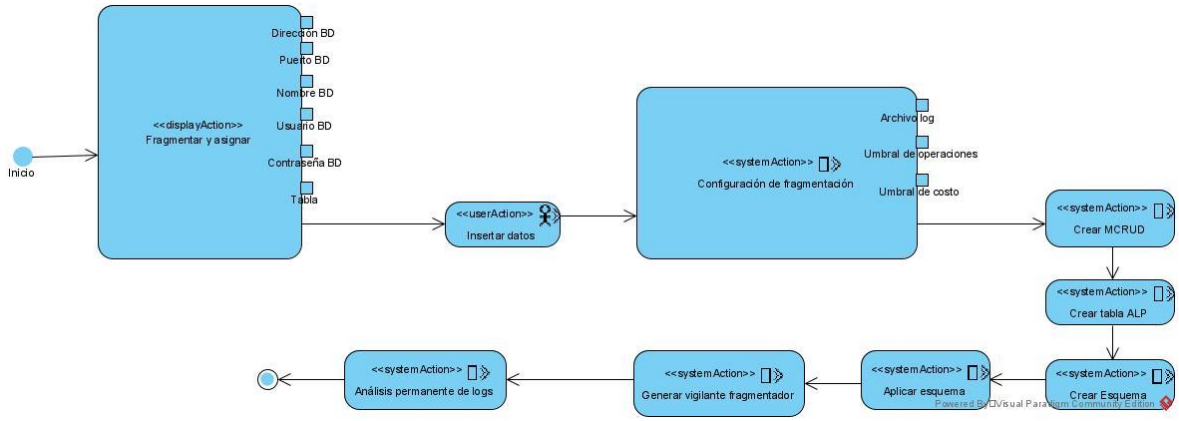


Figura 3.5 Diagrama de actividad del caso de uso "*Fragmentar y asignar*"

El diagrama de actividad del caso de uso "*Fragmentar y asignar*" muestra como el primer cuadro de acción el formulario para capturar toda la información necesaria de la base de datos: dirección de conexión, el puerto, nombre de la base de datos, nombre del usuario de la base de datos y la tabla a seleccionar. El segundo cuadro de acción presenta la actividad del usuario al capturar la información solicitada. En el tercer cuadro de acción se observa otro formulario, para ingresar el historial de consultas de la base de datos y dos umbrales para determinar si el esquema inicial no es adecuado para la base de datos. En el siguiente cuadro de acción se muestra la primera actividad del sistema, crear la matriz modificada de creaciones, lecturas, actualizaciones y eliminaciones (MCRUD). La siguiente actividad es crear la tabla de precedencia de localidad de atributos (ALP), por medio de esta tabla se determina el atributo por el cual se fragmentará la relación. En el siguiente cuadro de

actividad se realiza el esquema de fragmentación horizontal para bases de datos multimedia. En la siguiente actividad se aplica el esquema inicial a la relación de forma estática. Posteriormente, en la otra actividad se realizará la generación del vigilante fragmentador. Y como última actividad, se aplica la ejecución del vigilante fragmentador para realizar la parte de la fragmentación horizontal dinámica.

3.3.2 Diseño

3.3.2.1 Modelo conceptual

El modelo conceptual, también conocido como modelo de dominio, se encarga de describir cómo se relacionan los requisitos de la aplicación y de esta manera obtener el comportamiento del sistema. El modelo conceptual en este trabajo se representará por medio de tres diagramas principales: diagrama conceptual, diagrama lógico y diagrama físico de la base de la base de datos. Se observan estos diagramas en las Figuras 3.6, 3.7 y 3.8.

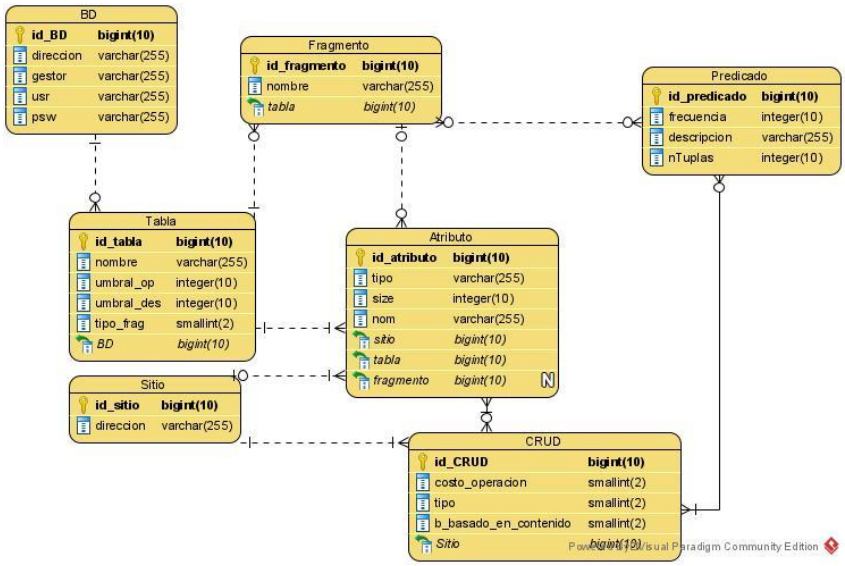


Figura 3.6 Diagrama conceptual de la aplicación.

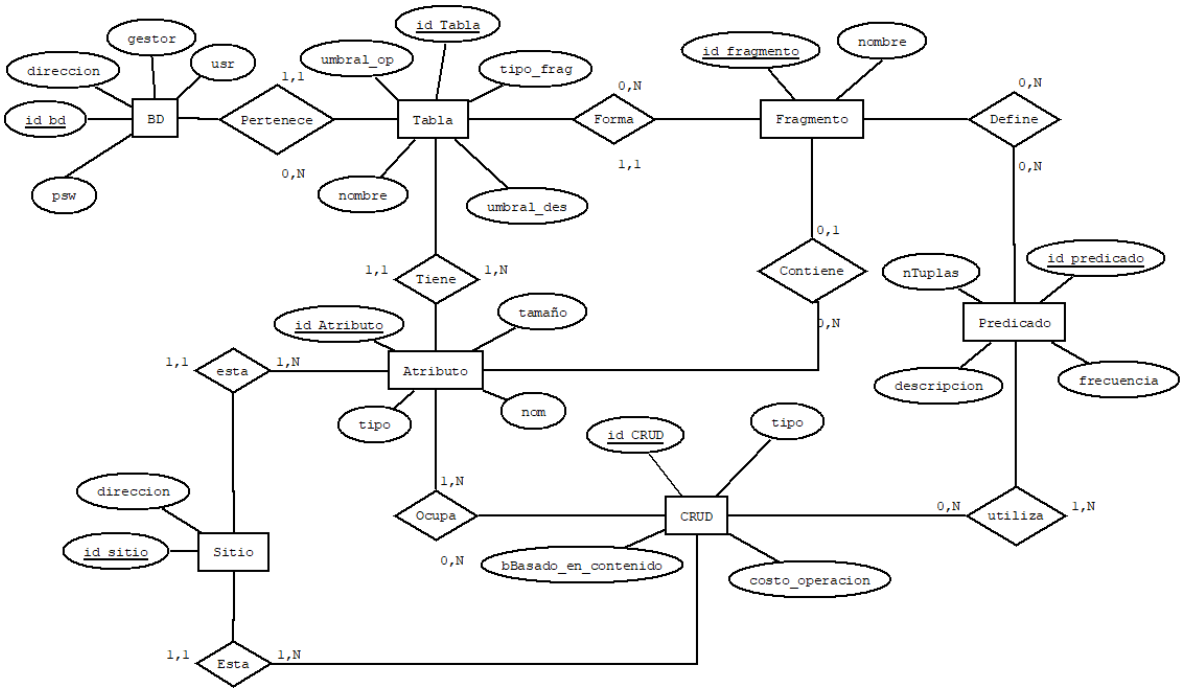


Figura 3.7 Diagrama lógico de la aplicación.

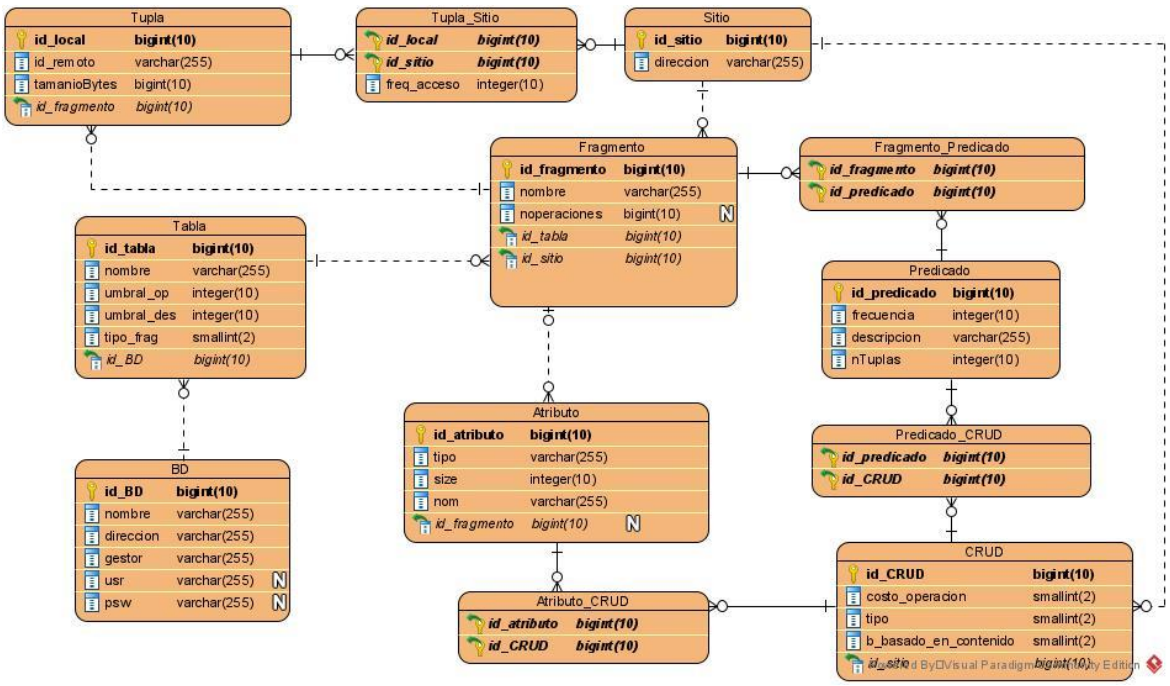


Figura 3.8 Diagrama físico de la aplicación

Tablas de base de datos

Tabla: La relación *Tabla* es la encargada de almacenar el nombre de la tabla a fragmentar, incluso el tipo de fragmentación que se realizará y los umbrales de operaciones y desempeño.

BD: La tabla *BD* almacena toda la información de la base de datos en la que se realizará la fragmentación horizontal dinámica, así como el gestor de la base de datos.

Fragmento: La tabla *Fragmento* almacena información del fragmento resultante al aplicar el método, de igual forma se almacena el número de operaciones que se considera en dicho fragmento.

Atributo: La tabla *Atributo* almacena el atributo por el cual se realizará el método y se obtendrán fragmentos de la relación, de igual forma, considera el nombre, el tipo y el tamaño del atributo

CRUD: La tabla *CRUD* es la encargada de almacenar cualquier operación registrada en el archivo del historial de consultas de la base de datos que se relaciona con el predicado. Se registra en esta tabla el tipo de operación: lectura, creación, actualización y eliminación, el costo de la operación y si se considera que está basado en contenido multimedia.

Predicado_CRUD: Esta tabla se encarga de mantener la cardinalidad entre las tablas *Predicado* y *CRUD*, ya que un *CRUD* posee la característica de tener muchos predicados y un predicado de estar en muchos *CRUD.*

Predicado: La tabla *Predicado* almacena todos los predicados de cada *CRUD* que se relaciona con la tabla a fragmentar y asignar. Con esta tabla se construirá la matriz *MCRUD.*

Atributo_CRUD: La tabla *Atributo_CRUD* relaciona todos los atributos utilizados en cada operación *CRUD*, de esta manera un atributo tiene la oportunidad de estar en muchas operaciones *CRUD* y una operación *CRUD* de utilizar muchos atributos.

Sitio: La tabla *Sitio* se encarga de almacenar todos los sitios a donde se hayan ejecutado *CRUD´*s relacionados con la tabla a fragmentar.

Tupla_Sitio: Se encarga de almacenar la frecuencia de acceso que se tiene en la tupla por cada sitio y de igual forma mantener la cardinalidad entre las tablas.

Tupla: La tabla *Tupla* se encarga de almacenar información de todas las tuplas que se encuentran en un fragmento, así como un id remoto, un id local y el tamaño de la tupla en bytes.

3.3.2.2 Modelo de navegación

En el modelo de navegación se conocen todos los caminos posibles dentro de la aplicación para la navegación de los usuarios. Este modelo propone crear mapas de navegación mediante diagramas de clase estereotipados, por medio de estos diagramas se describen todas las rutas posibles por las que se mueve un usuario dentro de un sitio o de una aplicación. En la Figura 3.9 se muestra el modelo de navegación de la aplicación.

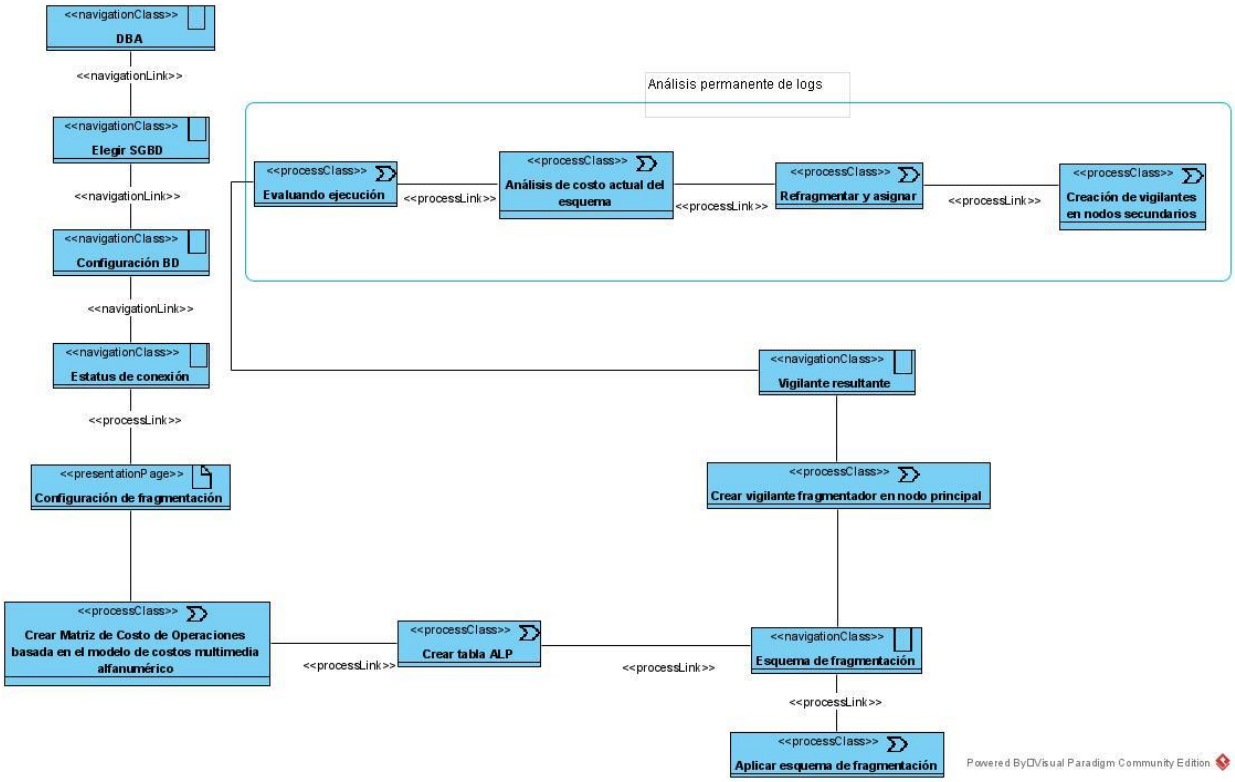


Figura 3.9 Modelo de navegación de la aplicación Web

En la Figura 3.9 se muestra el modelo de navegación del usuario DBA. La primera clase de navegación después del rol DBA es "*Configuración BD*", la cual es la encargada de mostrar el formulario para la obtención de la dirección y las credenciales de la base de datos que se desea fragmentar. Posteriormente, en la siguiente clase de navegación se encuentra el estatus de la conexión. Cuando la conexión a la base de datos sea exitosa, el siguiente paso es configurar la fragmentación, asignando los umbrales de operación y desempeño, así como la

obtención del archivo de carga. En la siguiente clase inicia el proceso de creación de la matriz CRUD basado en el modelo de costos multimedia alfanumérico. En la siguiente clase inicia el proceso para crear la tabla ALP. Una vez terminados los procesos anteriores, se presenta el esquema de fragmentación resultante para la base de datos. Después de ello se presentará la opción de aplicar esquema y de generar un vigilante de fragmentación, en la clase de aplicación de esquema, se aplicará un esquema inicial a la base de datos de manera estática, y la clase de "*Crear vigilante fragmentador en nodo principal*" se encarga de generar los archivos necesarios para aplicar la parte dinámica de la metodología. Posteriormente, se entrega el vigilante fragmentador. El análisis permanente de *logs*, es donde, de manera interna estará evaluando su ejecución, y donde se realizará un análisis del costo del esquema actual, en donde dependiendo el resultado del análisis se llevará a cabo el proceso de re-fragmentación y asignación, por último, se tiene el proceso de creación de vigilantes en nodos secundarios.

3.3.2.3 Modelo de presentación

En la Figura 3.10 se observa la página de *Configuración* del modelo de presentación, la cual se conforma de cinco elementos con el estereotipo *alternativeView*, el elemento con el nombre *SGBD* contiene un recuadro con el estereotipo *selection*, el cual permitirá seleccionar los gestores de bases de datos, el elemento con el nombre de *Formulario configurar conexión* contiene cinco recuadros *textInput* que permiten obtener toda la información requerida para llevar a cabo la fragmentación, el recuadro con el estereotipo *button* representa un botón para probar la conexión, el recuadro de *Tabla a fragmentar* mostrará todas las tablas disponibles en la base de datos. El recuadro de *Cancelar* representa un botón, el cual permite cancelar el proceso de fragmentación. Los recuadros restantes: *Mensaje error*, *Mensaje exitoso*, *Mensaje de carga* representan elementos modales, para mostrar mensajes al usuario.

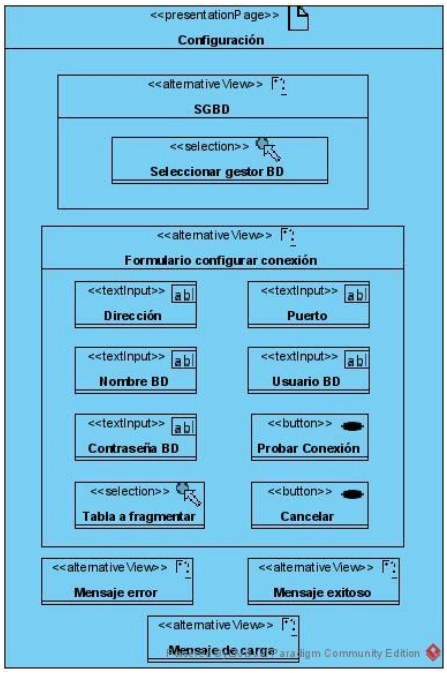


Figura 3.10 Página configuración del modelo de presentación.

En la Figura 3.11 se muestra la página de *Configuración de fragmentación* del modelo de presentación. El objetivo de esta página es configurar parámetros necesarios para el método según las necesidades del usuario, de igual forma, permite la carga del historial de consultas a la base de datos, la asignación de umbrales de operaciones y costo. El recuadro con el nombre *Formulario configuración* contiene dos botones con el estereotipo *button* los cuales le posibilita al usuario: 1) *Regresar*, volver a la página anterior para modificar información y 2) *Generar esquema*, el cual permite generar el esquema a partir de la información ingresada e ir a la siguiente página. El recuadro de abajo con el nombre *Mensaje de carga* tiene el propósito de informar al usuario que la aplicación está realizando una actividad.

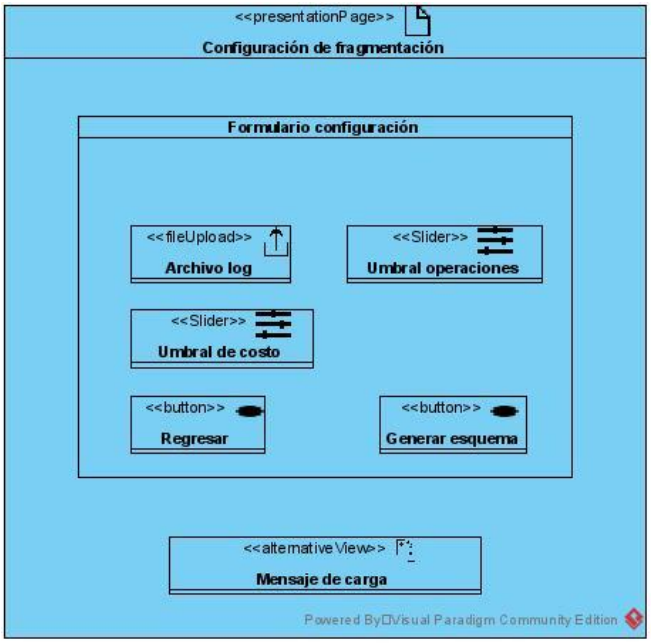


Figura 3.11 Configuración de fragmentación

En la Figura 3.12 se muestra la página *Esquema* del modelo de presentación. Esta página tiene la intención de mostrar el esquema resultante al usuario, para revisar si el resultado es el deseado antes de aplicar la metodología. Además, permitirá implementar el esquema resultante de una forma simple y generar el vigilante fragmentador adecuado para la base de datos para implementar la parte dinámica del método. Debajo del elemento *Esquema resultante*, se encuentra un elemento con el estereotipo *progressBar* el cual representa una barra de progreso que informa al usuario el porcentaje aplicado del esquema de fragmentación en la base de datos. De igual forma se tiene tres botones: 1) para regresar a la pantalla anterior, 2) un botón para generar el vigilante fragmentador y 3) otro botón para aplicar el esquema de manera estática. Los dos últimos elementos con el estereotipo *alternativeView* son modales, en donde *Mensaje de instrucciones* le proporcionará toda la información necesaria para implementar la parte dinámica de la metodología y *Token* le proveerá al usuario una clave única para poder implementar la parte dinámica de la metodología.

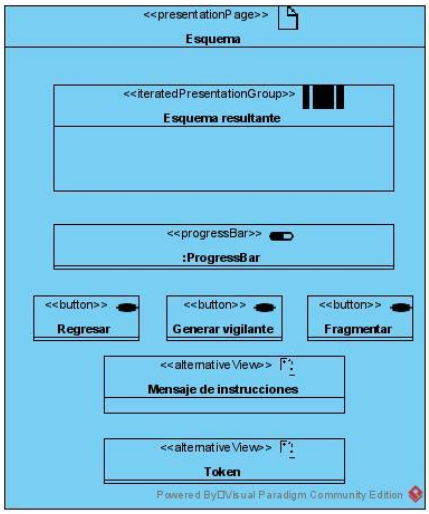


Figura 3.12 Página Esquema del modelo de presentación.

3.3.2.4 Modelo de procesos

El diagrama de procesos se basa en el diagrama de actividades, agregando diversos controles que enriquecen la especificación de la aplicación. En las Figuras 3.13 y 3.14 se presentan los diagramas de procesos de la aplicación, agregando diversos componentes como los nodos de decisión y la representación de los objetos dentro del flujo de actividades.

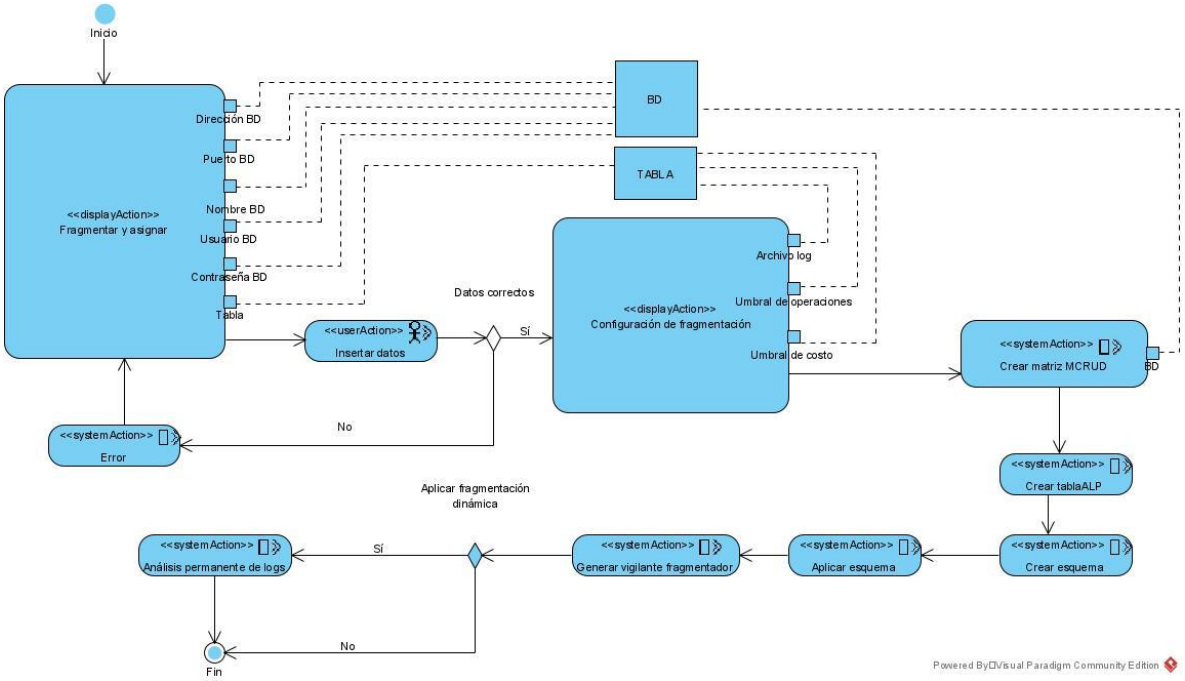


Figura 3.13 Diagrama *Fragmentar y asignar* del modelo de procesos.

El diagrama de procesos *Fragmentar y asignar* inicia desplegando un formulario para la captura de la dirección, el puerto, el nombre de la base de datos, el nombre de usuario y la contraseña de la base de datos, de igual forma, se captura la tabla a la cual se le aplicará la fragmentación horizontal dinámica. Se muestra posteriormente en el diagrama la representación del objeto BD (Base de datos) y TABLA (Tabla seleccionada para fragmentar), el BD se utiliza como parámetro en la acción del sistema *Crear matriz MCRUD* y TABLA para la *Configuración de fragmentación*. La *Configuración de fragmentación* ocurrirá si la información ingresada de parte del DBA en *Insertar datos* es correcta, entonces se presenta el formulario para cargar el archivo *log* que contiene un historial de todas las operaciones hechas en la base de datos y los umbrales de operaciones y costo. En la acción del sistema *Crear matriz MCRUD* no solo intervine el objeto BD, ya que, al tener una conexión exitosa, los demás objetos (Tabla, Atributo y CRUD, entre otros) se utilizan en este cuadro de acción del sistema.

La siguiente acción del sistema es *Crear tabla ALP*, la cual determinará los atributos que se utilizarán para fragmentar las tablas. Posteriormente, en *Crear esquema* utilizará los atributos para llevar a cabo su objetivo, de igual forma en esta acción, se presenta el esquema resultante antes de realizar la acción de *Aplicar esquema.* En *Aplicar esquema* el sistema realiza las tareas necesarias para modificar la base de datos de acuerdo con el esquema resultante. Después de esto, en la acción *Generar vigilante fragmentador*, el sistema genera un archivo JAR, que es necesario para aplicar la parte dinámica de la metodología. El DBA puede terminar el proceso en este punto y dejar la parte estática de la metodología, en caso contrario, se inicia la acción de *Análisis permanente de logs*.

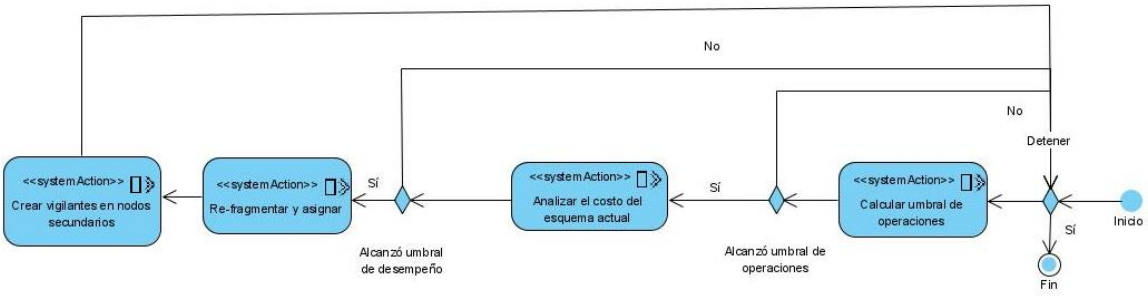


Figura 3.14 Diagrama *Análisis permanente de logs* del modelo de procesos.

En el diagrama de procesos *Análisis permanente de logs* se describe el proceso interno de esta acción mencionada en el diagrama anterior, el cual inicia evaluando su ejecución (hasta que el DBA lo considere necesario), posteriormente en la acción *Calcular umbral de operaciones*, se realizan las operaciones necesarias para obtener el umbral actual de operaciones y compararlo con el umbral considerado en el esquema actual, si este es sobrepasado, inicia la acción *Analizar el costo del esquema actual* y se revisa que este costo no supere el umbral de desempeño, de ser así, inicia la acción de *Re-fragmentar y asignar* en donde el sistema aplicará un nuevo esquema a la base de datos. Por último, se realiza la acción de *Crear vigilantes en nodos secundario*.

3.3.2.5 Arquitectura

Se realizó una serie de diagramas UML para modelar de manera adecuada la arquitectura que permita extender a más gestores de bases de datos sin afectar el flujo del proceso de la fragmentación horizontal para bases de datos multimedia, a continuación, se describe cada parte de la arquitectura.

Se creó la interfaz "*DataAccessRelational*" en donde se encapsula todo el comportamiento de una base de datos relacional, tales como la ejecución de consultas, la ejecución de comandos de conexión y desconexión a la base de datos (Figura 3.15). De igual forma, se creó una clase "*DataAccesMongoDB*" que encapsula las operaciones que se permiten realizar en una base de datos en MongoDB (Figura 3.15).

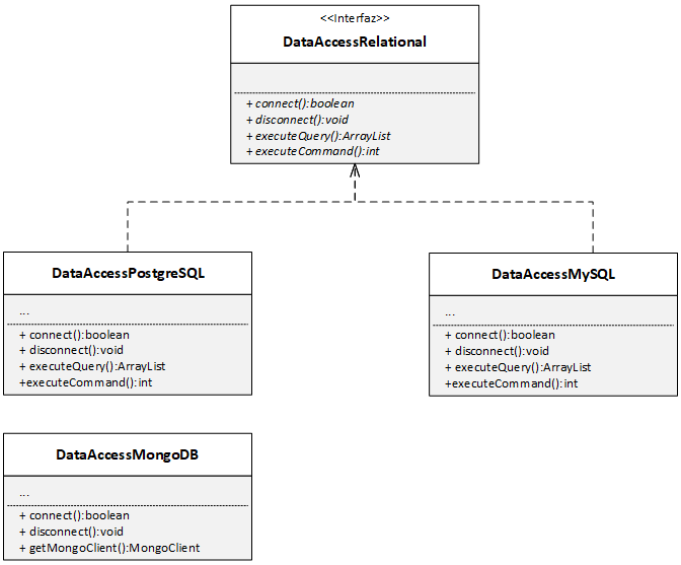


Figura 3.15 Arquitectura de la aplicación (*DataAccess*)

Después, se determinó una clase abstracta "*Filter*", la cual establece el comportamiento para filtrar la información obtenida de un archivo *log* y dar el tratamiento adecuado a la información para posteriormente determinar las operaciones encontradas del archivo (Figura 3.16), debido a que cada gestor de bases de datos tiene un formato único al momento de guardar la descripción de sus operaciones realizadas a las bases de datos, además de darle el formato adecuado, que permita determinar operaciones.

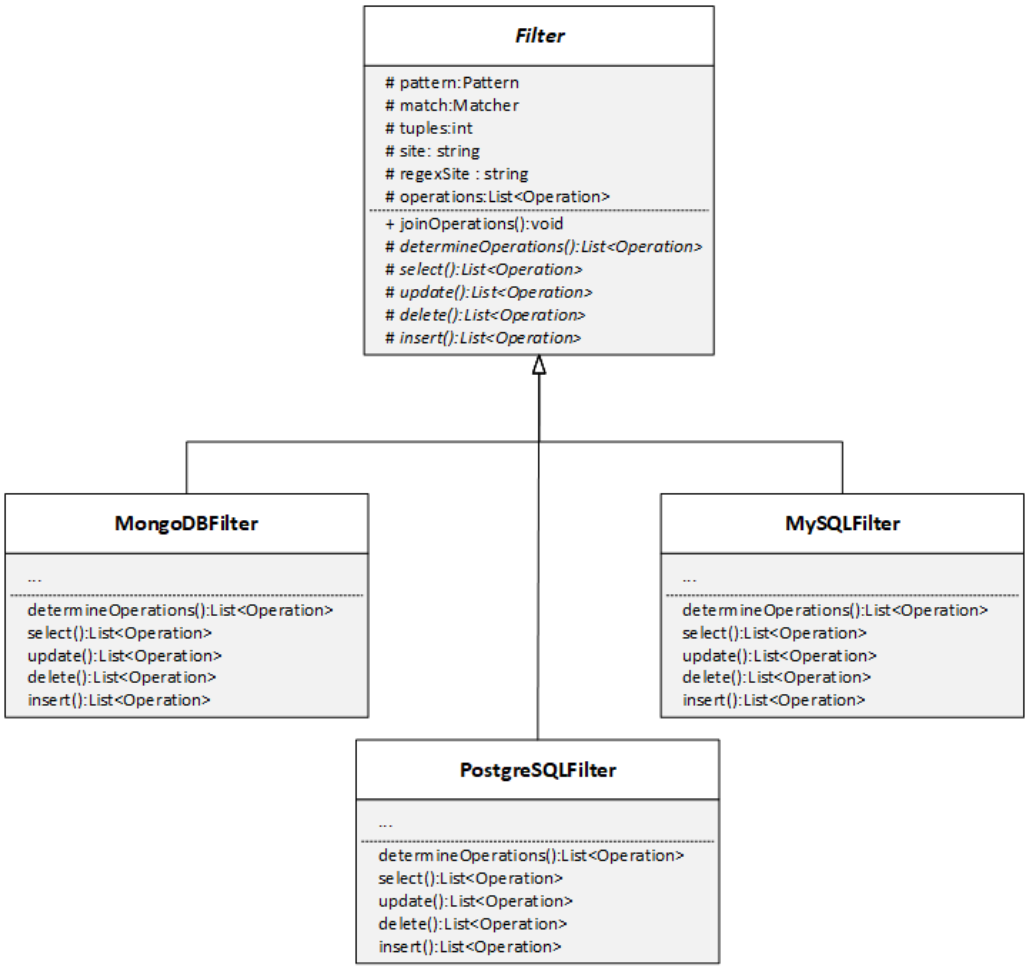


Figura 3.16 Arquitectura de la aplicación (*Filter*)

De igual forma, se creó una clase abstracta "*LogReader*" que encapsula los procesos necesarios para determinar las operaciones encontradas en el archivo *log* y crear objetos que faciliten la implementación del método de fragmentación (Figura 3.17).

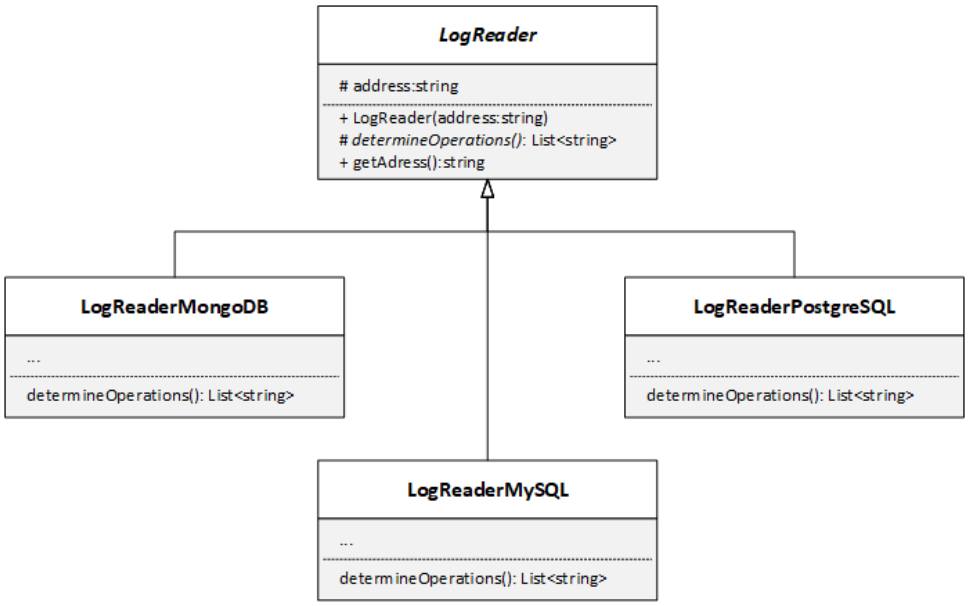


Figura 3.17 Arquitectura de la aplicación (*LogReader*)

Después, se creó otra interfaz "*BuilderFragments*" que tiene como objetivo determinar el comportamiento para construir los fragmentos de un esquema en un gestor de bases de datos (Figura 3.18). Esto se debe a que cada gestor de bases de datos tiene formas particulares de crear relaciones o colecciones de datos. Por ello, es necesario determinar para cada gestor, la forma en cómo se crean dichas relaciones o colecciones de datos.

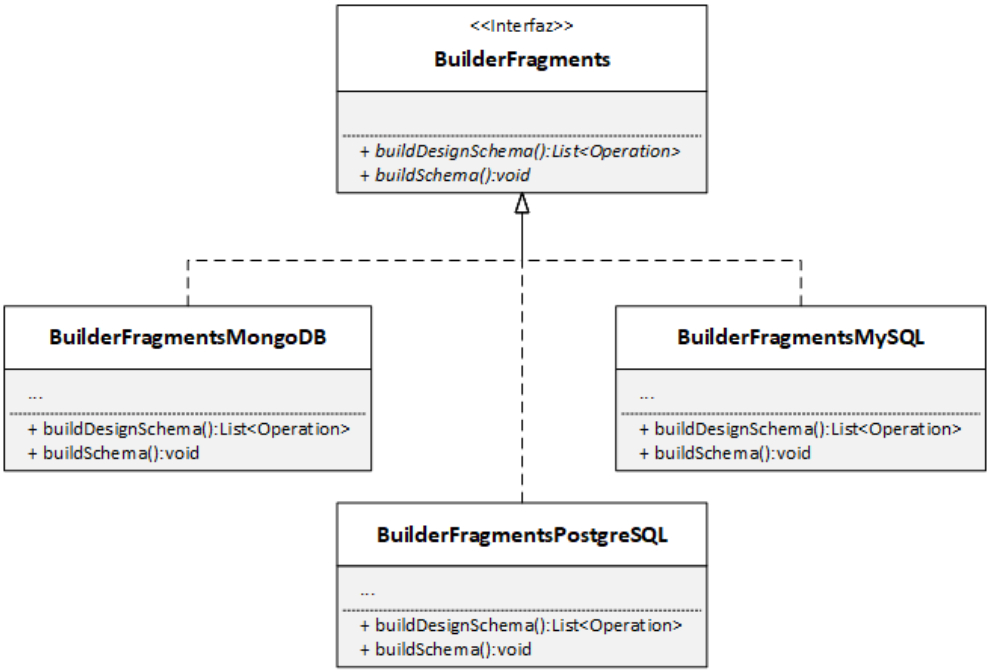


Figura 3.18 Arquitectura de la aplicación (*BuilderFragments*)

Una vez determinada cada clase abstracta e interfaz, se obtuvo la siguiente arquitectura para realizar fragmentación horizontal para bases de datos multimedia (Figura 3.19). Como se aprecia en la Figura 3.19 en la clase "*DHFragmentation*" con ayuda del polimorfismo y el uso adecuado de las interfaces en el proceso de fragmentación horizontal es posible implementar en varios gestores de bases de datos sin tener que cambiar la lógica del proceso. Obteniendo una arquitectura que es capaz de extenderse a más gestores de bases de datos si es necesario.

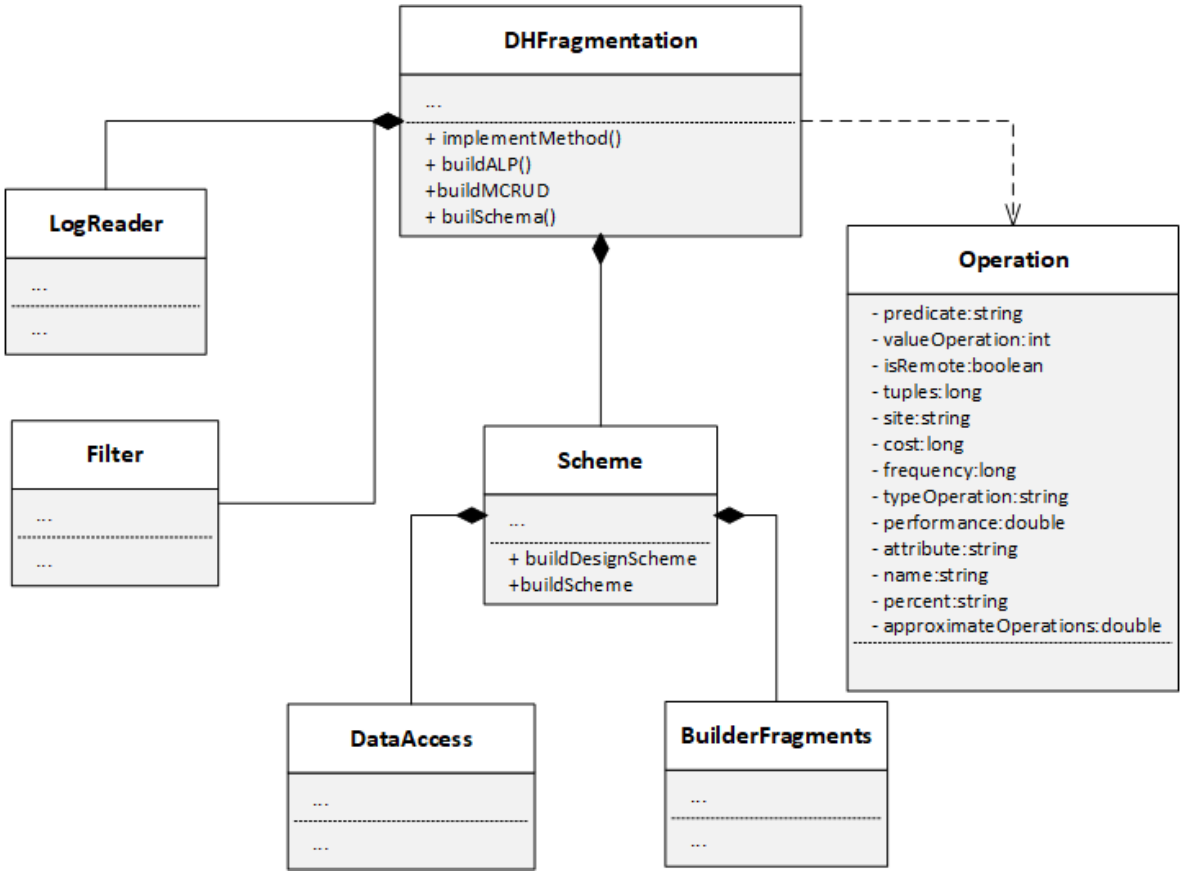


Figura 3.19 Arquitectura de la aplicación

3.3.3 Implementación

Durante esta investigación se desarrolló una aplicación web nombrada XAMANA y un programa de consola empaquetado en un archivo JAR como vigilante de fragmentación, desde los cuales se aplica el método de fragmentación horizontal dinámica para bases de datos multimedia. La aplicación web presenta toda la información generada y el esquema propuesto por parte de la fragmentación inicial de la metodología, que se aplicará físicamente a lo largo de los sitios detectados. El vigilante de fragmentación realiza la parte dinámica de

la metodología, tratando de adaptar el esquema, conforme determina las operaciones de cada fragmento y evaluando si se superan de forma individual los umbrales de operaciones o desempeño. La aplicación web y el vigilante de fragmentación implementan la metodología desarrollada, donde se considera la columna que tiene un mayor impacto en las operaciones encontradas de la tabla que se desee implementar la metodología, obteniendo un esquema óptimo de forma operativa.

La aplicación web permite analizar el historial de operaciones de una base de datos, conectarse a un servidor de forma remota, seleccionar una tabla de un esquema encontrado en el gestor de bases de datos, mostrar información previa del análisis y aplicar el esquema a los sitios encontrados en el análisis previo. El vigilante de fragmentación supervisa y analiza el historial de operaciones del gestor de bases de datos de cada sitio en donde se ubique cada fragmento creado por la aplicación web.

Se definieron clases que permiten asignar de forma adecuada el objeto de cada clase abstracta e interfaz para el gestor de bases de datos multimedia que se desee fragmentar, como se aprecia en las siguientes Figuras 3.20 - 3.22. En el capítulo 4 se describe de forma detallada la implementación de la metodología.

```java
public class FilterCreator implements Serializable {

    public Filter createFilter(String optionManager,String site) {
        Filter filter = null;

        switch (optionManager) {
            case "MySQL":
                filter = new FilterMySQL(site);
                break;
            case "Postgres-XL":
            case "PostgreSQL":
                filter = new FilterPostgreSQL(site);
                break;
            case "MongoDB": //MongoDB
                filter = new FilterMongoDB(site);
                break;
        }

        return filter;

    }

}
```

Figura 3.20 Clase para crear objetos de tipo *Filter*

```java
public class LogReaderCreator implements Serializable{

    public LogReader createLogReader(String optionManager, String addrees) {
        LogReader logReader = null;

        switch (optionManager) {
            case "MySQL":
                logReader = new LogReaderMySQL(addrees);
                break;
            case "Poostgre-XL":
            case "PostgreSQL":
                logReader = new LogReaderPostgreSQL(addrees);
                break;
            case "MongoDB":
                logReader = new LogReaderMongoDB(addrees);
                break;
        }

        return logReader;
    }

}
```

Figura 3.21 Clase para crear objetos de tipo *LogReader*

```java
public class BuilderFragmentsCreator {

    public BuilderFragments createBuilderFragments(bd bd) {
        BuilderFragments BF = null;

        switch (bd.getTipoBD()) {
            case "MySQL":
                BF = new BuilderFragmentsMySQL(bd);
                break;
            case "Postgre-XL":
            case "PostgreSQL":
                BF = new BuilderFragmentsPostgreSQL(bd);
                break;
            case "MongoDB":
                BF = new BuilderFragmentsMongoDB(bd);
                break;
        }

        return BF;
    }


}
```

Figura 3.22 Clase para crear objetos de tipo *BuilderFragments*

3.4 Validación

Para validar la metodología desarrollada en este proyecto de tesis se presenta un caso de estudio y una comparación con otra técnica seleccionada del análisis realizado al estado del arte. El caso de estudio se enfoca en la implementación del método en una base de datos llamada “HITO” en un gestor de bases de datos MongoDB, fragmentando una colección de documentos de 1870 registros en donde se encuentra contenido multimedia que son imágenes y sus descriptores. La comparación de la metodología muestra los tiempos de repuesta de algunas operaciones realizadas al conjunto de fragmentos obtenidos de cada enfoque, para corroborar la eficiencia de la técnica desarrollada en este proyecto contra otra.

Capítulo 4. Resultados

En este capítulo se muestran los resultados obtenidos del análisis del estado del arte y se describe la implementación de la metodología a través de la aplicación web y el vigilante de fragmentación. De igual forma se presenta la comparación del método propuesta contra otra técnica.

4.1 Resultados del análisis

Para terminar la etapa de análisis, en esta sección se muestran los resultados obtenidos a través de gráficas y tablas. De esta forma se visualizan con mayor claridad los datos presentados en la Tablas 3.1 a 3.5.

La Figura 4.1 muestra una comparación del número de publicaciones hechas por editorial: 16 de estas publicaciones se concentran en la editorial IEEE, donde la mayoría de los trabajos, describen de forma completa la técnica, son fáciles de implementar, se basan en modelos de costos y están orientados a la fragmentación estática (por ejemplo [12], [31], [32]). En ACM se encuentran con mayor frecuencia métodos capaces de adaptar un esquema y sencillos de ejecutar ([8] , [10], [14], [15], [16], [17], [19]). La mayoría de las estrategias de Elsevier son fáciles de implementar ([41], [42], [43]). En Springer se centraron en procedimientos fáciles de implementar y completos ([44], [45], [46], [21], [47], [48], [51], [52]). Por último, las técnicas de otras editoriales utilizaron modelos de costos y son fáciles de implementar ([13], [54], [55]).

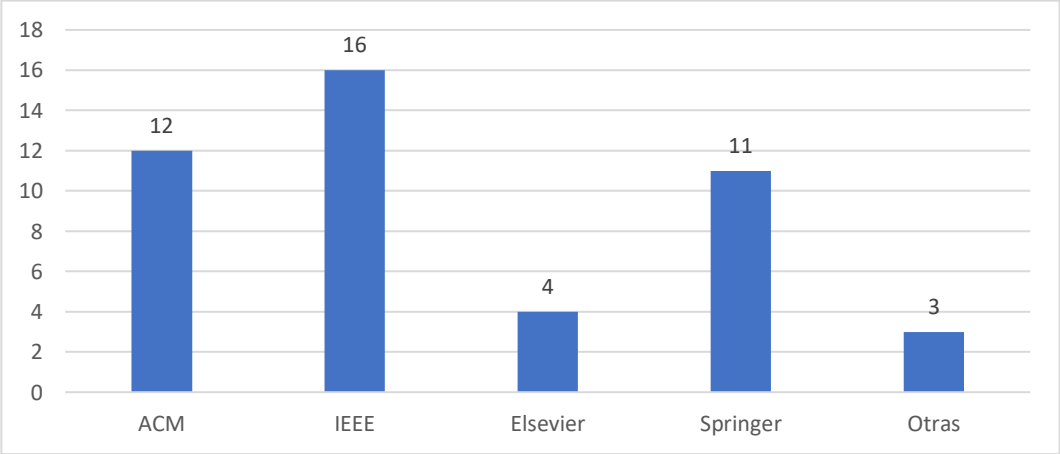


Figura 4.1 Cantidad de artículos por editorial

En la Figura 4.2 se presenta el número de artículos por año de publicación. El gráfico muestra que el mayor número de publicaciones se encuentra entre los años 2014 y 2020. En el año 2020, los artículos se centraron en los métodos de fragmentación horizontal con facilidad de implementación y el uso de modelos de costos [12], [13], [17].

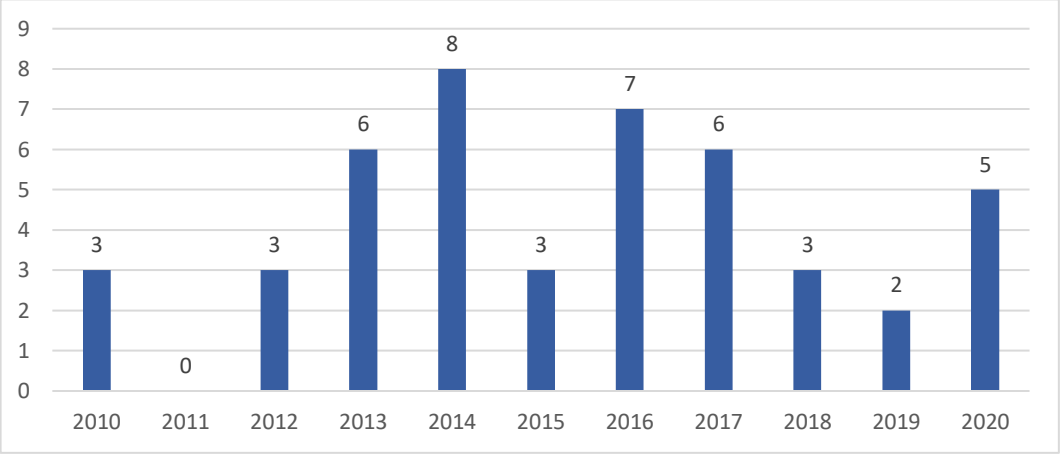


Figura 4.2 Cantidad de artículos por año de publicación.

La gráfica de la Figura 4.3 muestra los artículos que presentan una facilidad de implementación, tomando en cuenta los conocimientos específicos para implementar las técnicas de cada publicación; hay 34 de 46 artículos con esta característica. La mayoría de ellos son fáciles de implementar porque son completos, es decir, incluyen todo el proceso y la información para replicarlo. La Figura 4.4 muestra el número de publicaciones que ofrecen mayor integridad en su estrategia de fragmentación.

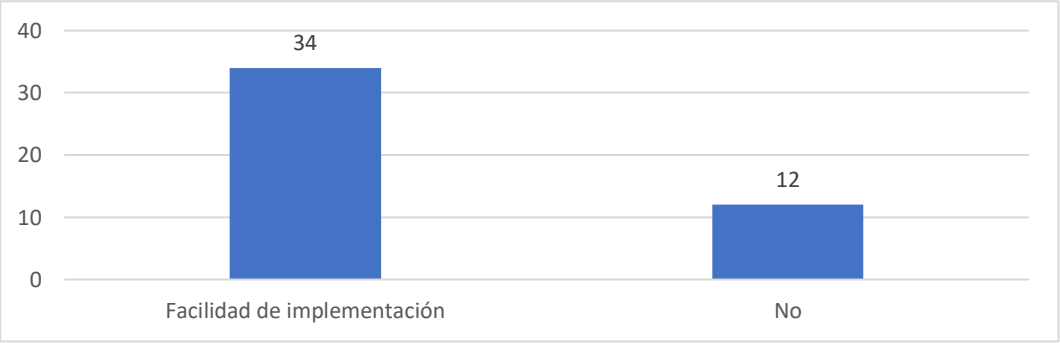


Figura 4.3 Cantidad de artículos por año facilidad de implementación.

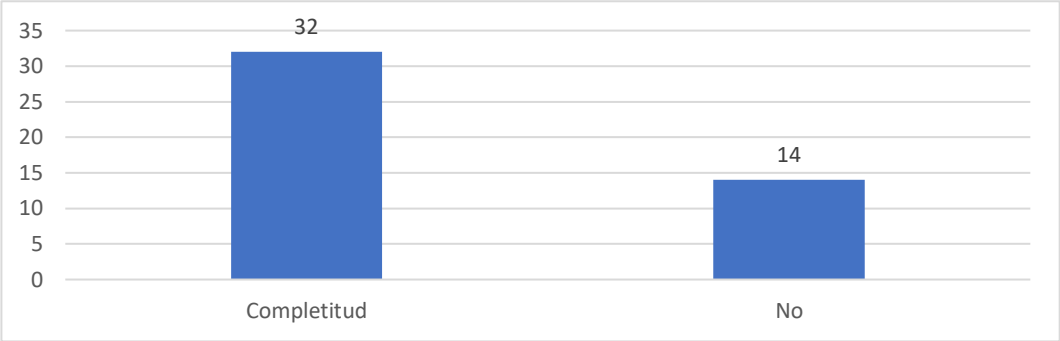


Figura 4.4 Cantidad de artículos por completitud.

La Figura 4.5 muestra una gráfica que relaciona el número de artículos que tienen en cuenta un modelo de costos; 23 de 46 métodos utilizaron un modelo de costos [6], [10], [17], [18], [20] y [28] de ACM; [12], [29], [31], [32], [34], [38] y [39] de IEEE; [44], [21], [47], [49], y [50] de Springer; [40] y [41] de Elsevier; y [13], [54] y [55] de otras editoriales. La mayoría de los enfoques con modelo de costos se centran en la reducción del acceso a datos no relevantes ([6], [10], [12], [13], [17], [20], [28], [31], [34], [38], [39], [40], [42], [44], [21], [47], [49], [54], [55]) y en la E/S de disco ([18], [29], [32], [50]), que pretenden optimizar la ejecución de las consultas en los sistemas de gestión de datos.

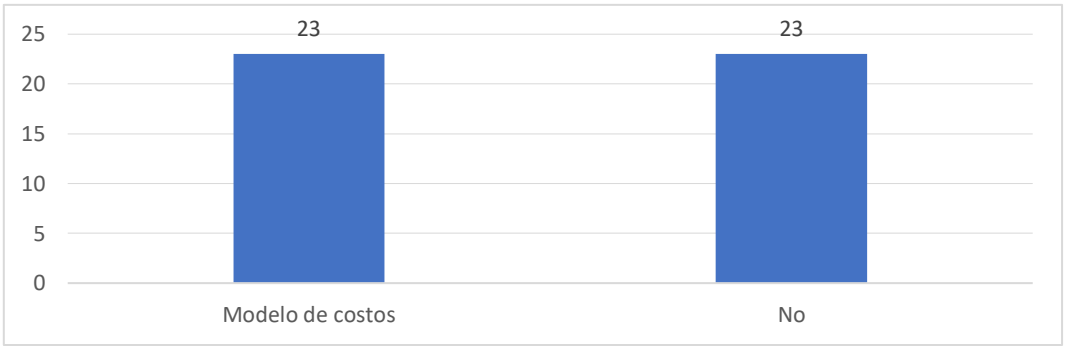


Figura 4.5 Cantidad de artículos por modelo de costos.

La Figura 4.6 presenta una comparación del número de trabajos que consideran un método de fragmentación horizontal dinámica ([5], [7], [8], [9], [10], [14], [15], [16], [17], [19], [20], [28], [29], [33], [34], [38], [39], [40], [41], [44], [45], [46], [47], [50], [54],) y los que no ([4], [6], [11], [12], [13], [18], [30], [31], [32], [35], [36], [37], [42], [43], [21] [48], [49], [51], [52], [53], [55]). A partir del gráfico, se observa que las estrategias de fragmentación dinámica están tomando un gran interés, debido a la necesidad de automatizar la implementación de métodos que adapten el esquema de fragmentación horizontal (EFH) de

la base de datos en función a los cambios en el patrón de acceso o la carga de trabajo. Sin embargo, existen artículos que aún incluyen fragmentaciones estáticas (es decir, implementan un EFH que no considera los cambios de la información).

Figura 4.6 Cantidad de artículos por fragmentación dinámica.

La Figura 4.7 presenta los tipos de bases de datos utilizados en cada trabajo. El gráfico muestra que la mayoría de los métodos se aplicaron en: bases de datos distribuidas ([4], [6], [11], [17], [20], [32], [33], [38], [41], [43]) distribuidas en la nube ([7], [12], [15], [29], [31], [34], [40], [44], [47]), y relacionales ([18], [46], [45] [49], [50], [51], [52], [55], [54],). Del mismo modo, existe una tendencia a utilizar técnicas de fragmentación horizontal en las bases de datos NoSQL ([5], [8], [35], [37], [48], [53]) y Multimedia ([13], [30], [36], [42], [21]). Sin embargo, todas las estrategias que consideran datos multimedia implementan fragmentación estática y solo las publicaciones [13] y [21] tienen en cuenta un modelo de costos para determinar el EFH a utilizar.

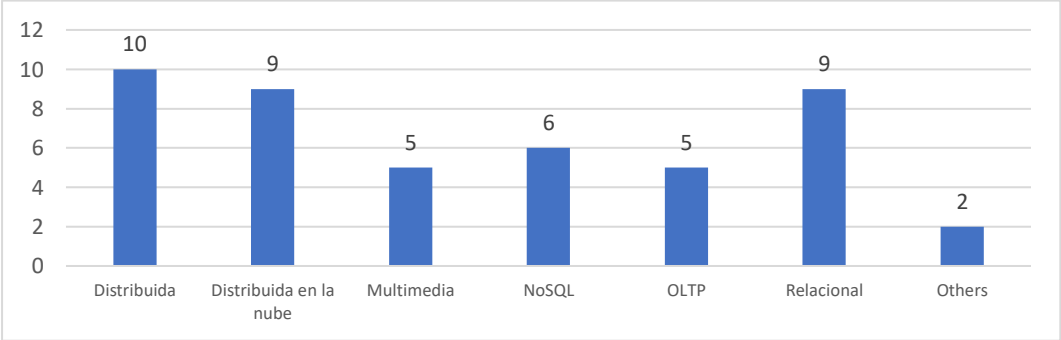


Figura 4.7 Cantidad de artículos por tipo de base de datos.

La Figura 4.8 muestra los sistemas gestores de bases de datos (SGBD) utilizados en los artículos, se observa que la mayoría de las publicaciones no mencionan el SGBD utilizado, sin embargo, la mayoría de las estrategias fueron implementadas en MongoDB ([5], [13], [36], [37], [35], [42], [53]), PostgreSQL ([4], [15], [18], [20], [32], [50]) y MySQL ([6], [12], [16], [19]). De los métodos propuestos para los sistemas de gestión de contenido multimedia, tres de los cinco utilizaron MongoDB ([13], [36], [42]) y solo [30] usó Oracle.

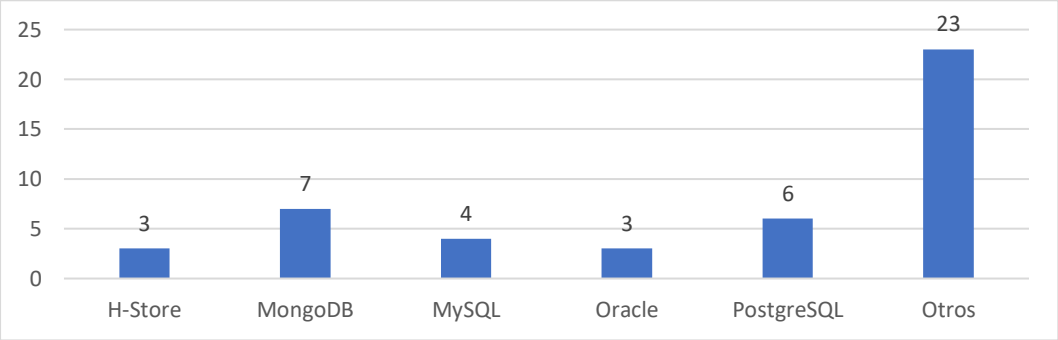
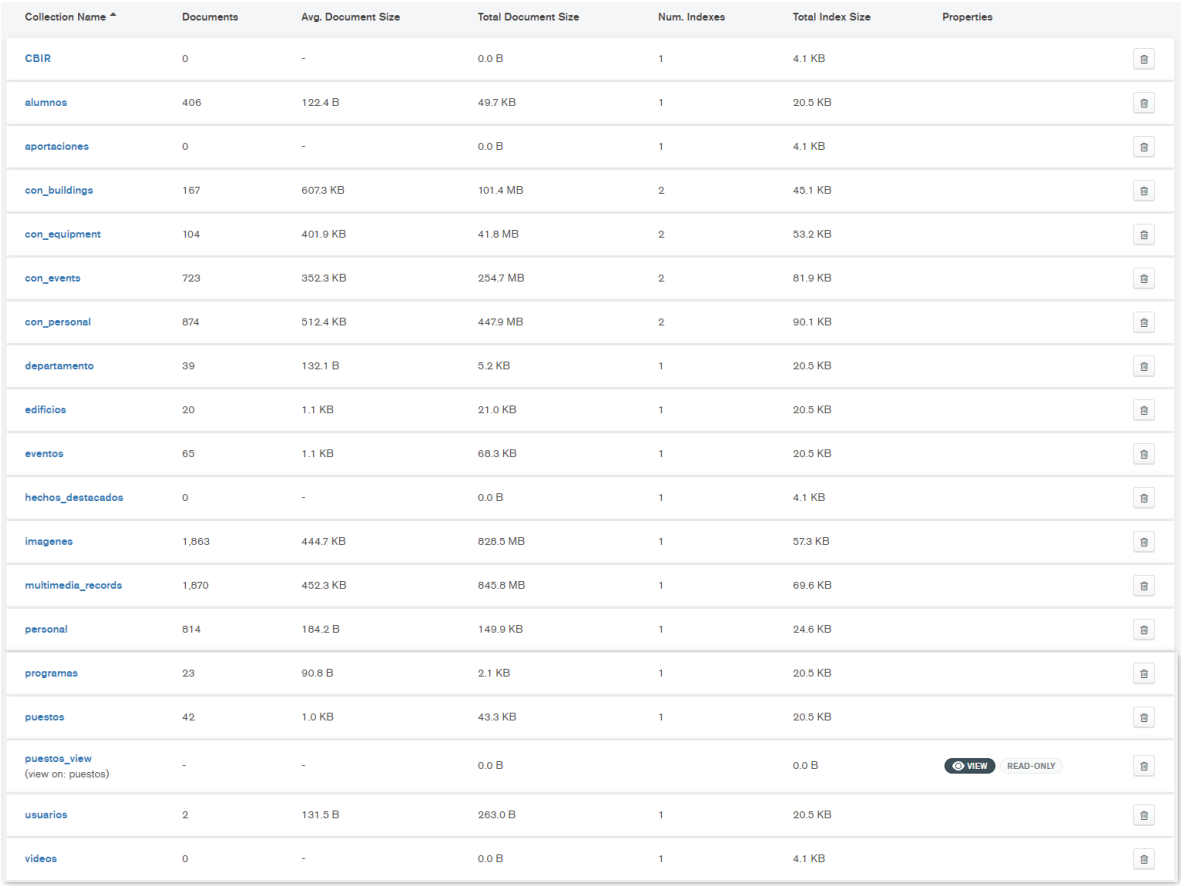


Figura 4.8 Cantidad de artículos por gestor de bases de datos.

4.2 Base de datos HITO

HITO es una base de datos del Instituto Tecnológico de Orizaba, la cual almacena colecciones de tipo multimedia, como fotografías de alumnos, de edificios, de eventos, personal, equipo, videos, entre otras. Toda la base de datos se encuentra en MongoDB, la cual es una base de datos NoSQL, es decir, una base de datos no relacional, y las tablas son colecciones de documentos, con contenido multimedia y alfanumérico, cada documento cuenta con un identificador único en toda la base de datos, para poder identificar una tupla con otra. En la Figura 4.9 se muestra una captura de las colecciones almacenadas en dicha base de datos.



Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
CBIR	0	-	0.0 B	1	4.1 KB	
alumnos	406	122.4 B	49.7 KB	1	20.5 KB	
aportaciones	0	-	0.0 B	1	4.1 KB	
con_buildings	167	607.3 KB	101.4 MB	2	45.1 KB	
con_equipment	104	401.9 KB	41.8 MB	2	53.2 KB	
con_events	723	352.3 KB	254.7 MB	2	81.9 KB	
con_personal	874	512.4 KB	447.9 MB	2	90.1 KB	
departamento	39	132.1 B	5.2 KB	1	20.5 KB	
edificios	20	1.1 KB	21.0 KB	1	20.5 KB	
eventos	65	1.1 KB	68.3 KB	1	20.5 KB	
hechos_destacados	0	-	0.0 B	1	4.1 KB	
imagenes	1,863	444.7 KB	828.5 MB	1	57.3 KB	
multimedia_records	1,870	452.3 KB	845.8 MB	1	69.6 KB	
personal	814	184.2 B	149.9 KB	1	24.6 KB	
programas	23	90.8 B	2.1 KB	1	20.5 KB	
puestos	42	1.0 KB	43.3 KB	1	20.5 KB	
puestos_view (view on: puestos)	-	-	0.0 B		0.0 B	VIEW READ-ONLY
usuarios	2	131.5 B	263.0 B	1	20.5 KB	
videos	0	-	0.0 B	1	4.1 KB	

Figura 4.9 Colecciones de la base de datos HITO.

4.3 Aplicación Web XAMANA

La pantalla inicial del sistema es “Configuración”, donde se muestra una lista desplegable, que permite seleccionar el gestor de base de datos, en donde se implementa el método y un botón de conectar como se aprecia en la Figura 4.10.

Al hacer clic en el botón de conectar, se muestra un modal, en donde se aprecia un formulario que permite capturar la información de la base de datos (dirección, puerto, nombre de la base de datos), la información del administrador de la base de datos (usuario y contraseña), una lista desplegable para seleccionar la relación o colección a fragmentar que se encuentre en la base de datos, y dos botones; uno para comprobar la conexión, y otro para cancelar el proceso, en la Figura 4.11 se presenta el formulario.

Figura 4.10 *Configuración.*

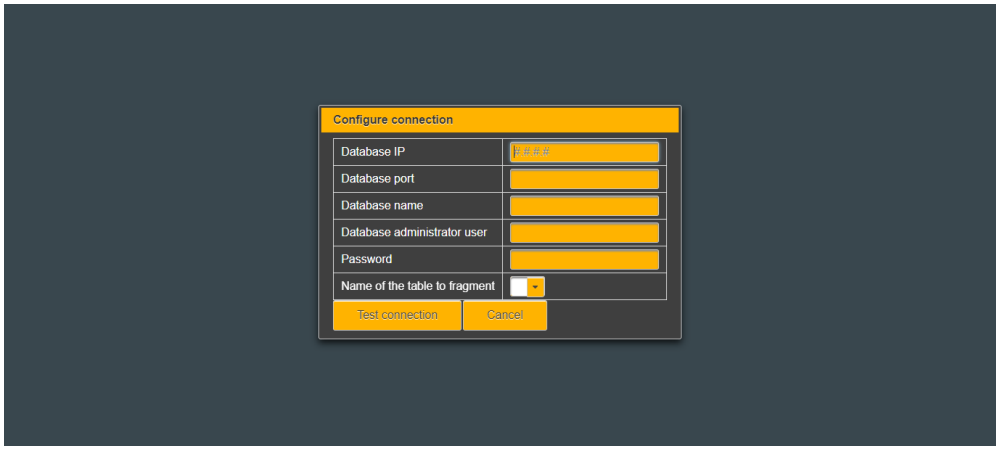


Figura 4.11 *Configuración* - formulario de configuración de conexión.

Una vez que el administrador ingrese todos los datos solicitados y presione el botón "*Test connection*", el sistema comprueba la conexión, de ser exitosa, se presenta un diálogo, indicando que la conexión a la base de datos fue exitosa y en la parte inferior se muestra un botón que permite ir a la página de "*Configuración de fragmentación*", como se muestra en la Figura 4.12.

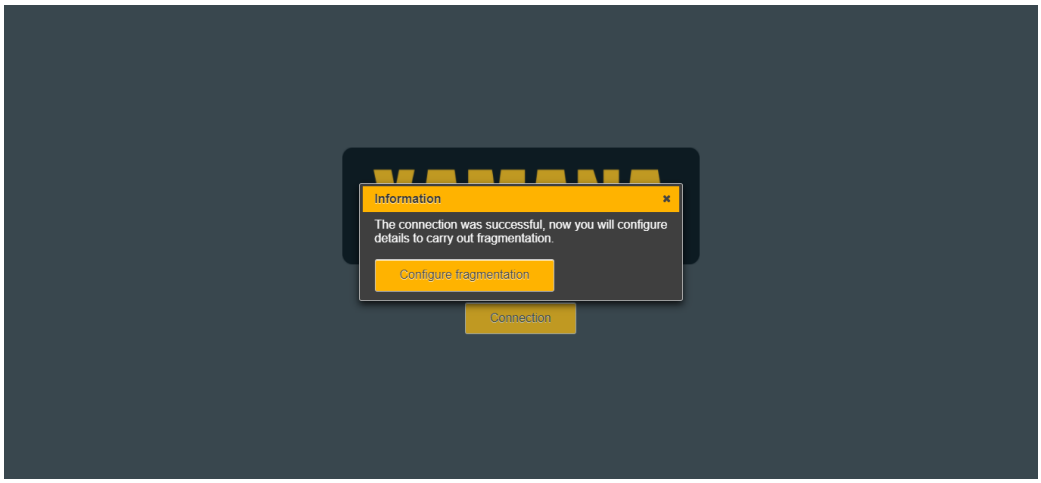


Figura 4.12 *Configuración* - modal de conexión exitosa.

En la pantalla de *Configuración de Fragmentación*, se presenta un formulario, que permite seleccionar el tipo de fragmentación, si considera consultas basadas en contenido (la técnica solo se enfoca en fragmentación horizontal y no considera consultas basadas en contenido) y dos controles *slider*, que permiten asignar un valor a los umbrales de operación y desempeño, los *sliders* permiten asignar valores de -100 a 100, estos valores indican la sensibilidad del vigilante de fragmentación al momento de realizar una re-fragmentación, donde los valores negativos o menores causan que el vigilante de fragmentación sea más sensible y los valores positivos hasta 100 hacen que sea menos sensible el vigilante de fragmentación. Una vez configurada la fragmentación, el administrador debe hacer clic al botón de generar esquema para realizar el análisis e ir a la página de *Esquema* (Figura 4.13). La aplicación muestra un modal con animación de carga, para indicar que la ampliación está realizando el análisis, como se observa en la Figura 4.14.

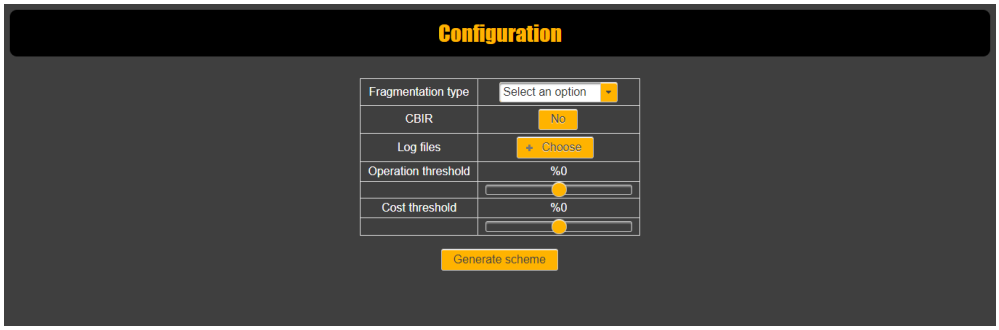


Figura 4.13 *Configuración de fragmentación.*

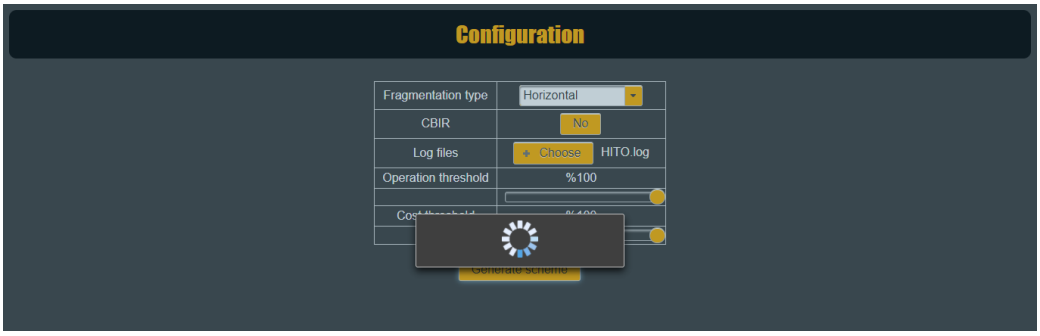


Figura 4.14 *Configuración de fragmentación* - modal de carga.

Después de realizar el análisis de todas las operaciones encontradas en el archivo *log* se presenta la página *Esquema* con un modal inicial que muestra una breve instrucción de cómo aplicar la parte dinámica del método como se aprecia en la Figura 4.15

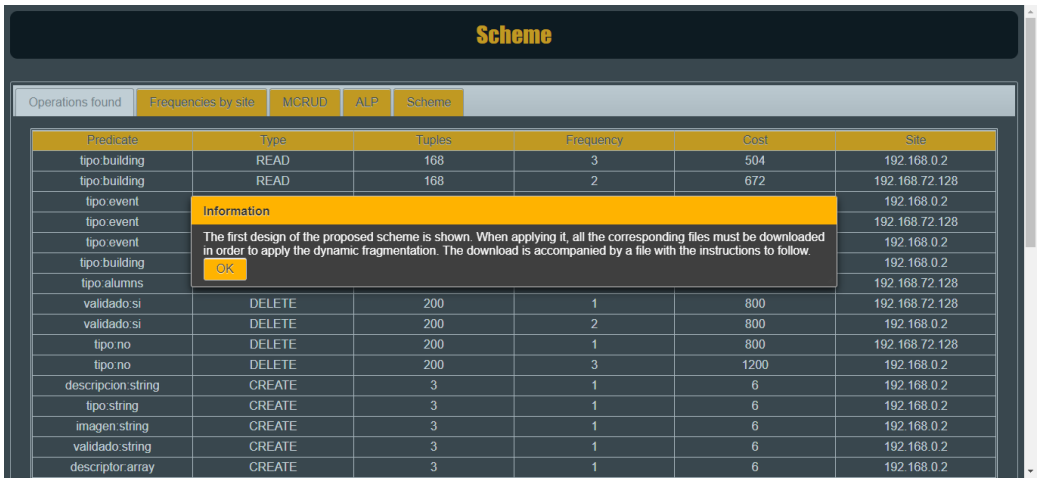


Figura 4.15 *Esquema* - modal con instrucciones.

Una vez que el administrador de base de datos cierra el modal, se observan cinco pestañas (Figura 4.16) que muestran la información obtenida del análisis de las operaciones encontradas en el archivo *log*, para que el administrador visualice de forma detallada el proceso que realizó la aplicación web y verifique si el esquema propuesto por la aplicación web es adecuado, de igual forma, se presenta una barra de progreso que indica el porcentaje de implementación de la fragmentación por parte de la aplicación web y dos botones, uno para realizar la fragmentación y otro para obtener el vigilante de fragmentación. La primera pestaña muestra las operaciones encontradas en el archivo *log* en forma de tabla, en donde se aprecia el predicado de cada operación, el tipo de operación, las tuplas afectadas, la frecuencia de ejecución, el costo determinado por el modelo de costos multimedia alfanumérico, el sitio donde se ejecutó la operación como se observa en la Figura 4.16. El costo de cada operación se determina con el tipo de operación, la frecuencia de la operación, el sitio donde se realizó la operación y las tuplas afectadas. Con ayuda del modelo de costos, se asigna un valor para cada operación dependiendo el tipo, asignando un valor de 1 a las operaciones de lectura (*READ*), un valor de 2 si son operaciones de creación (*CREATE*) y eliminación (*DELETE*) y un valor de 3 si son operaciones de actualización (UPDATE). La ubicación en donde se ejecuta cada operación afecta el costo, dado que una operación realizada de forma remota es más costosa, por la transmisión de datos, por lo cual, si una operación es realizada de forma remota, (es decir, que la operación se realizó de un servidor diferente al principal) el valor se duplica. Otro factor que se toma en cuenta es el número de tuplas afectadas, esto es necesario, porque cada tupla puede incluir o no contenido multimedia, y cada objeto impacta de forma negativa en los tiempos de respuesta de las operaciones y en la transmisión de datos. Para simplificar esta explicación se muestra un ejemplo de cómo se determina cada costo en la Tabla 4.1 (en este caso el sitio 192.168.0.2 es el servidor principal).

Tabla 4.1 Ejemplo de cálculo de costo de operaciones.

Predicado	Tipo	Sitio	Tuplas afectadas	Frecuencia	Costo Tipo * tuplas * frecuencia * sitio
Tipo = building	READ	192.168.0.2	5	2	1 * 5 * 2 * 1 = 10
Tipo = building	READ	192.168.72.128	5	2	1 * 5 * 2 * 2 = 20
Tipo = event	UPDATE	192.168.0.2	3	3	3 * 3 * 3 * 1 = 27
Tipo = event	UPDATE	192.168.72.128	3	3	3 * 3 * 3 * 2 = 54
Validado = si	DELETE	192.168.0.2	2	1	2 * 2 * 1 * 1 = 4
Validado = si	DELETE	192.168.72.128	2	1	2 * 2 * 1 * 2 = 8
Tipo = alumno	CREATE	192.168.0.2	1	5	2 * 1 * 5 * 1 =10
Tipo = alumno	CREATE	192.168.72.128	1	5	2 * 1 * 5 * 2 =20

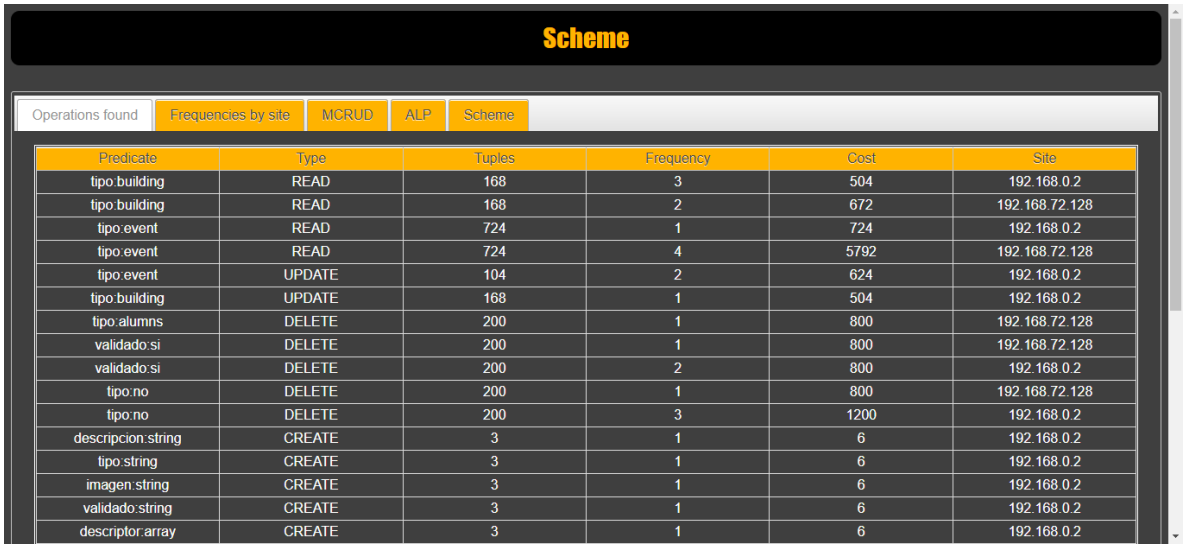


Figura 4.16 *Esquema* – Operaciones encontradas.

La segunda pestaña muestra la frecuencia de cada operación por sitio (Figura 4.17), donde el administrador visualiza cómo se determinó el sitio más adecuado, para ubicar cada posible fragmento por medio de la frecuencia, el sitio con mayor frecuencia por cada predicado es la ubicación más adecuada para almacenar cada posible fragmento. De igual forma, se aprecia el costo general de cada predicado. El recuadro amarillo en la parte inferior derecha muestra la relación de cada sitio con su abreviación.

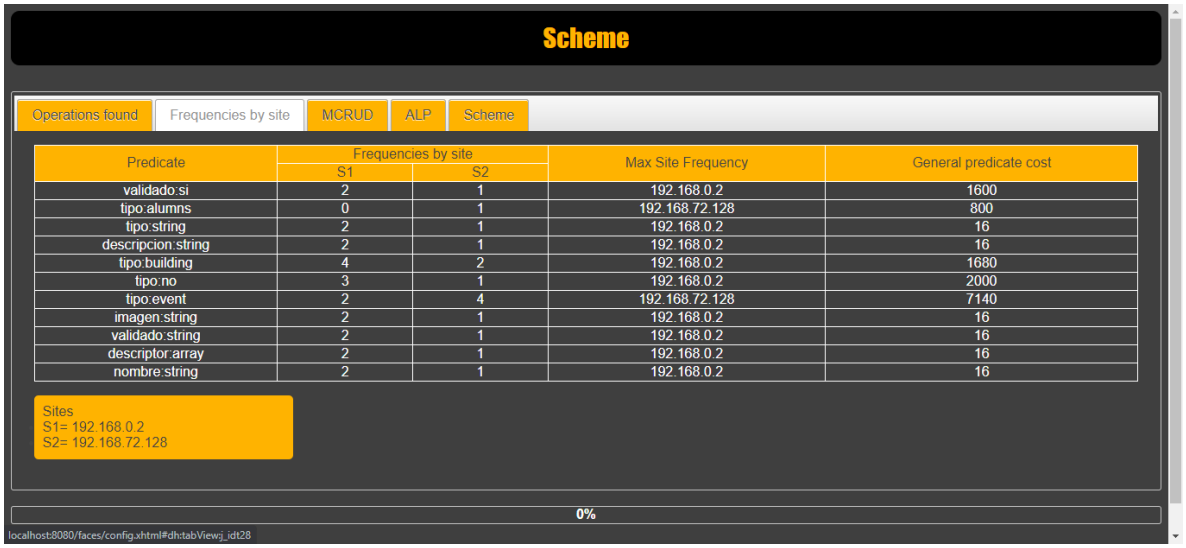


Figura 4.17 *Esquema* – Frecuencias por sitio.

Una vez determinado el sitio con mayor frecuencia y el costo general de cada predicado, se procede a construir la matriz MCRUD en forma de tabla (Figura 4.18), donde se observa cada predicado, el sitio donde se ubicaría el posible fragmento, y el costo de cada predicado, esta matriz muestra de una forma más clara cada posible fragmento y su ubicación.

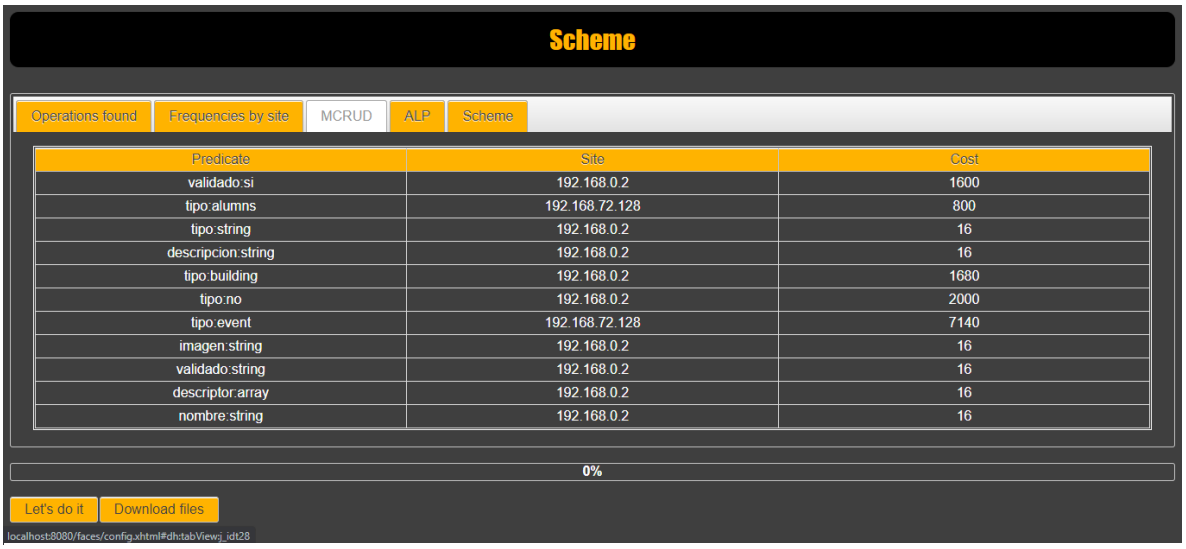


Figura 4.18 *Esquema* – MCRUD.

Una vez construida la matriz MCRUD, con ayuda de esta información, se procede a construir la tabla ALP en donde se determina el atributo que tienen un mayor impacto en las operaciones encontradas en el archivo *log*, en la tabla se observa (Figura 4.19) el atributo y el costo de dicho atributo.

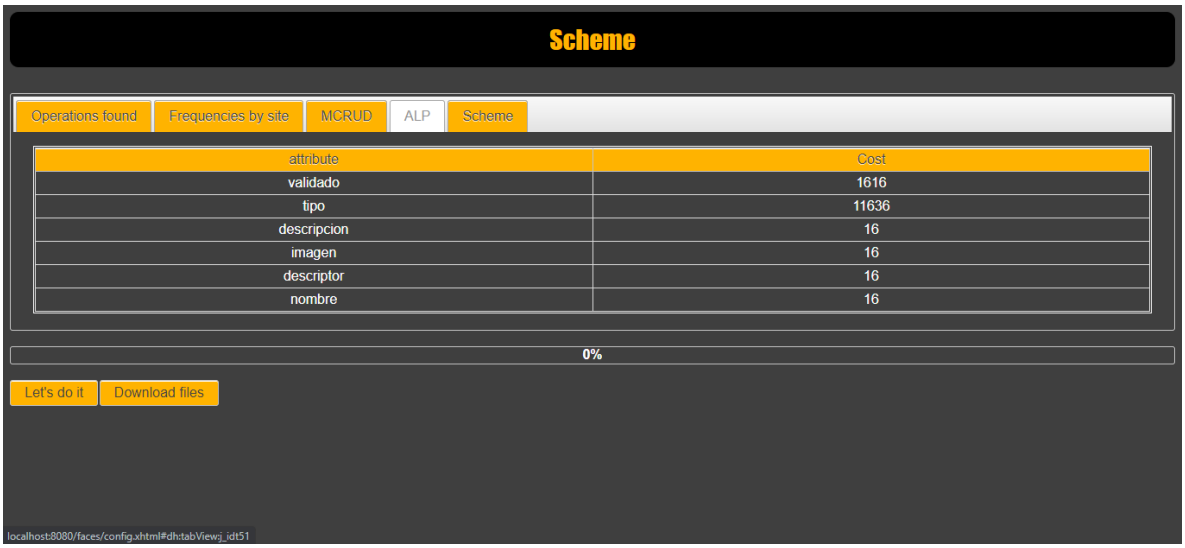


Figura 4.19 *Esquema* – ALP.

Con ayuda de la matriz MCRUD y la tabla ALP se determina un posible esquema como se visualiza en la Figura 4.20. En esta tabla se aprecia el nombre del fragmento, el predicado que determinó cada fragmento, el sitio donde se ubicará el fragmento, el número de tuplas o filas que almacenará cada fragmento, el porcentaje que representa cada fragmento de la tabla o relación original, las operaciones iniciales/aproximadas de cada fragmento y el valor de desempeño inicial de cada fragmento. Si el esquema es adecuado a las necesidades del administrador de base de datos, el administrador puede implementar el esquema con el primer botón "*Let´s do it*" y la aplicación comenzará a crear cada fragmento en el sitio correspondiente y asignará cada tupla considerando el predicado de cada fragmento, de igual forma, con ayuda de la barra de progreso, indicará el porcentaje de construcción del esquema. Con el botón "*Download files*" el administrador puede obtener las instrucciones y el vigilante de fragmentación para iniciar con la parte dinámica del método.

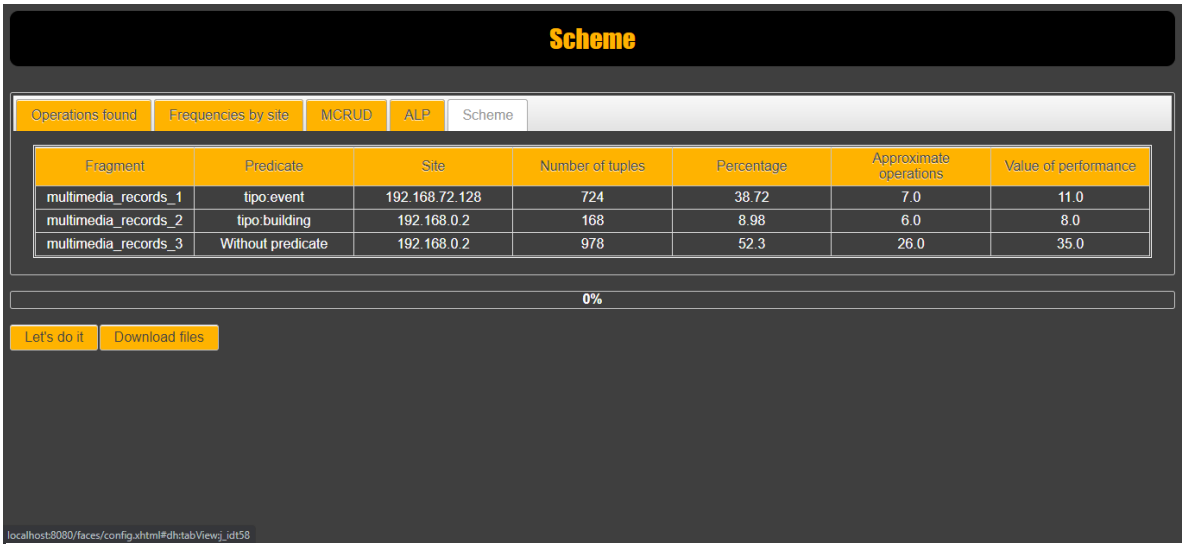


Figura 4.20 Esquema – Información del esquema.

Una vez terminada la implementación del esquema, se muestra un modal en donde se observa un *token* (Figura 4.21), el *token* es necesario para que el administrador de base de datos implemente la parte dinámica del método si lo considera necesario, en caso contrario, se mantendrá un esquema estático. Con ayuda de MongoDB Compass, se observa el esquema implementado en la base de datos "HITO". Se visualiza que cada instancia de la aplicación está conectada a cada sitio de la red, con el mismo puerto, cada fragmento se identifica con el nombre que se presenta en el esquema propuesto por la aplicación web como se ve en la Figura 4.22.

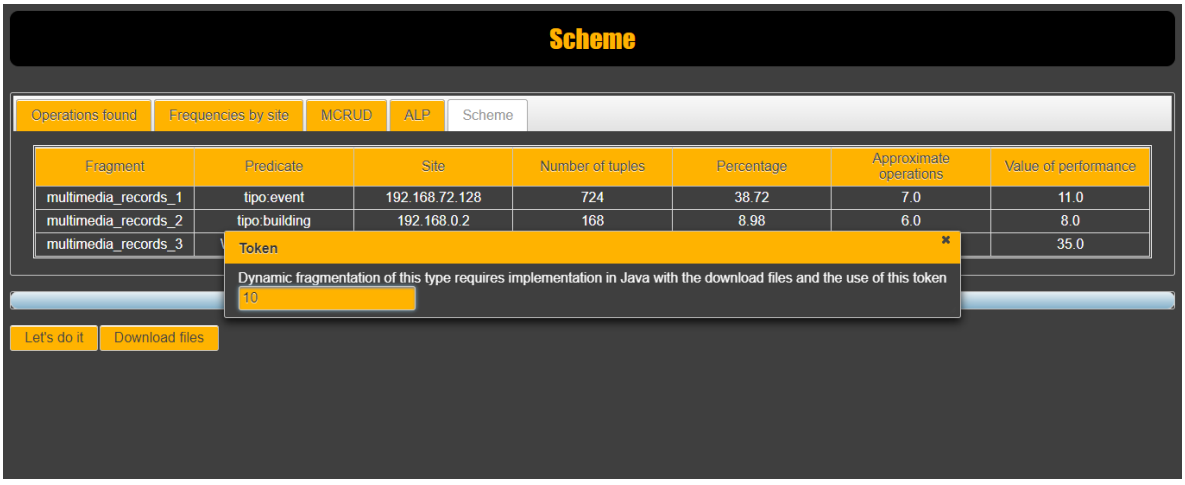


Figura 4.21 Esquema – *token*.

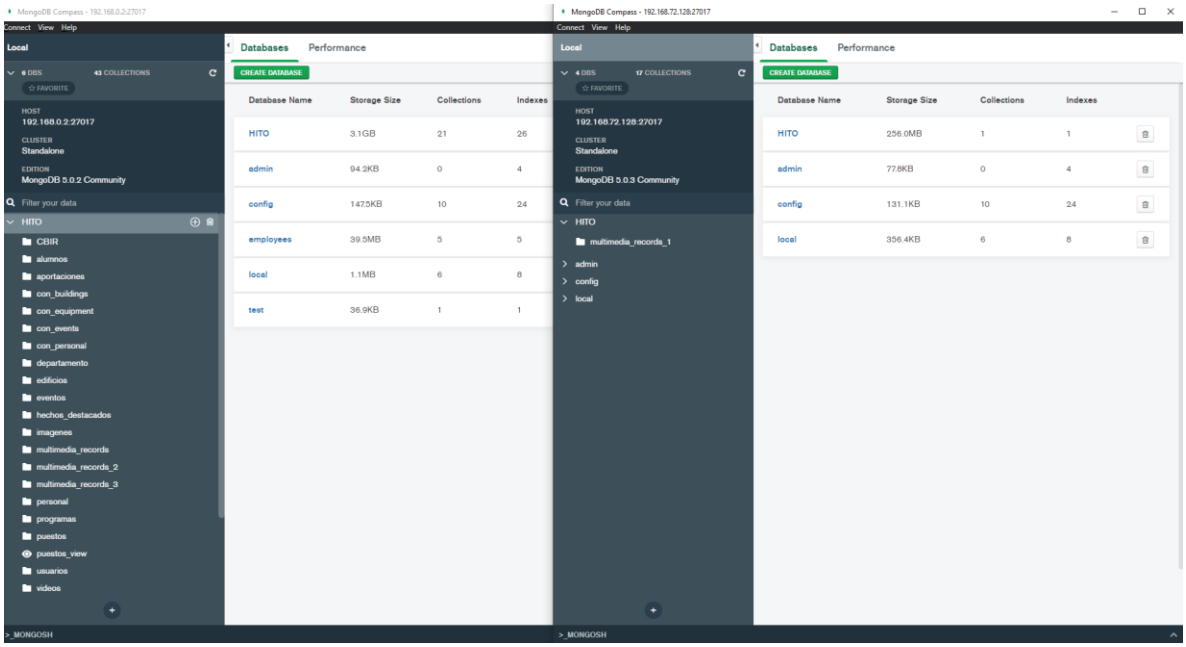


Figura 4.22 Esquema – MongoDB Compass.

4.4 Vigilante de fragmentación

Para implementar la parte dinámica de la metodología, es necesaria la ejecución de un programa de consola en formato JAR, que es el Vigilante de fragmentación, dado que este método tiene como objetivo distribuir todos los fragmentos derivados de una colección de datos a través de varios servidores, para mantener el rendimiento de la base de datos.

Cada vigilante de fragmentación solo re-fragmentará los fragmentos que se encuentren en su sitio (servidor) y si el proceso afecta a otro sitio, con ayuda de una base de conocimientos registrará dicho ajuste, por lo tanto, cada vigilante de fragmentación evaluará si es necesario actualizar la información de los fragmentos que supervisa por cada sitio.

En la Figura 4.23 se observa que cada vigilante de fragmentación se comunicará con una base de conocimientos y si es necesario realizará el registro de un nuevo sitio o la actualización de fragmentos que se encuentren en cada sitio.

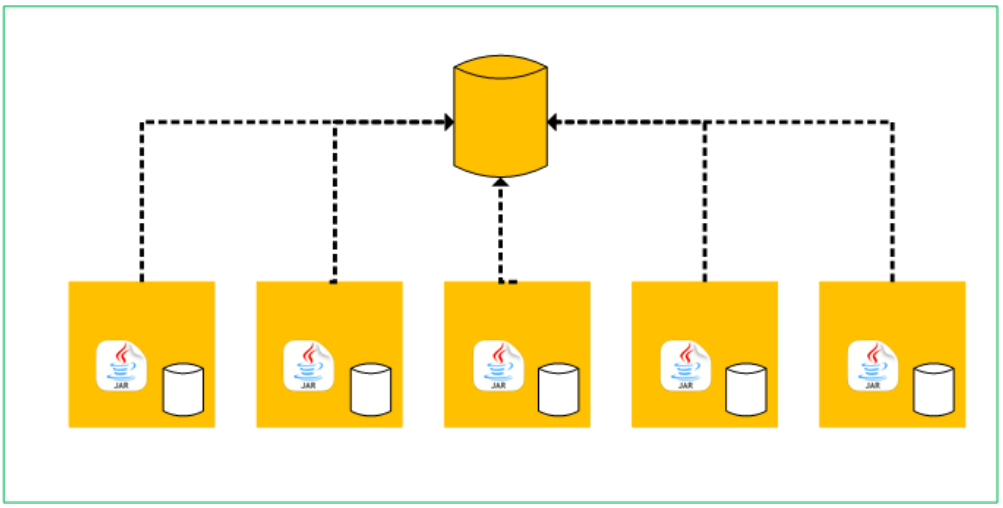


Figura 4.23 Vigilantes de fragmentación supervisando fragmentos por sitio.

El vigilante de fragmentación debe ejecutarse en la consola o terminal del servidor, una vez que el administrador de base de datos haya ejecutado el vigilante de fragmentación, el programa mostrará una impresión de bienvenida y de inmediato deberá ingresar el *token* obtenido de la aplicación web. Si el administrador de base de datos ingresa un *token* inválido el sistema le indicará que el *token* no es válido (verifica que el token sea un número entero positivo) y, de igual forma, el vigilante de fragmentación evaluará si el *token* está relacionado con un esquema a supervisar. La impresión de bienvenida se observa en la Figura 4.24

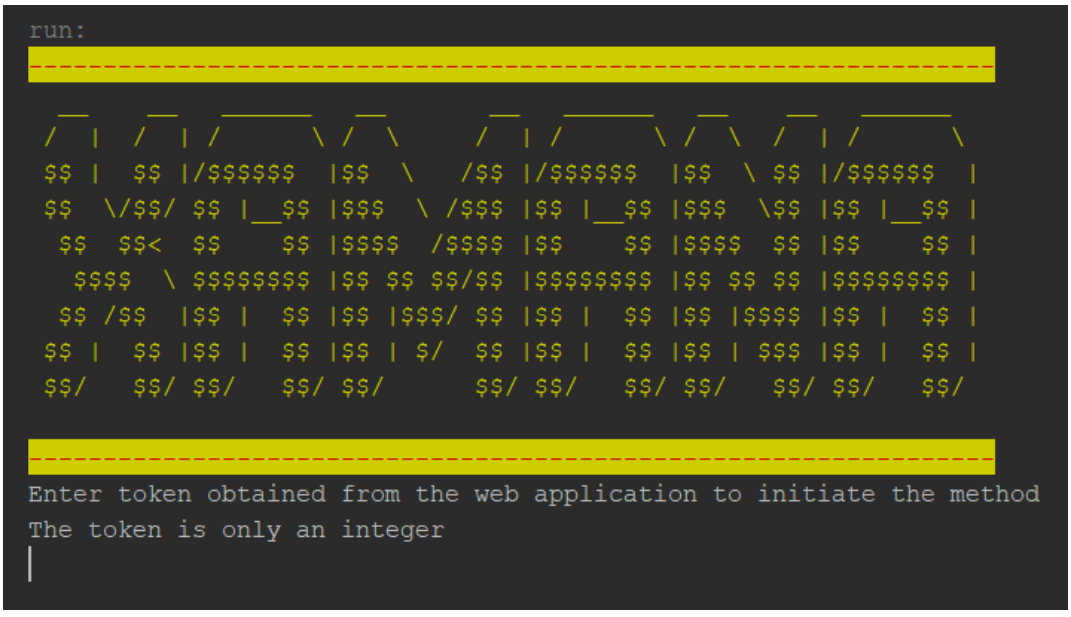


Figura 4.24 Vigilante de fragmentación - impresión de bienvenida.

Una vez ingresado el *token*, el vigilante de fragmentación muestra información relacionada con el *token*, el nombre de la base de datos, la tabla o colección de datos a fragmentar y el usuario para acceder a la base de datos (Figura 4.25). Es necesario que el administrador de base de datos verifique esta información, porque puede existir múltiples esquemas de fragmentación de la base de datos y es necesario asegurarse que el vigilante de fragmentación está supervisando el esquema deseado. El vigilante de fragmentación envía una petición para confirmar si se desea o no implementar la fragmentación dinámica de la metodología.

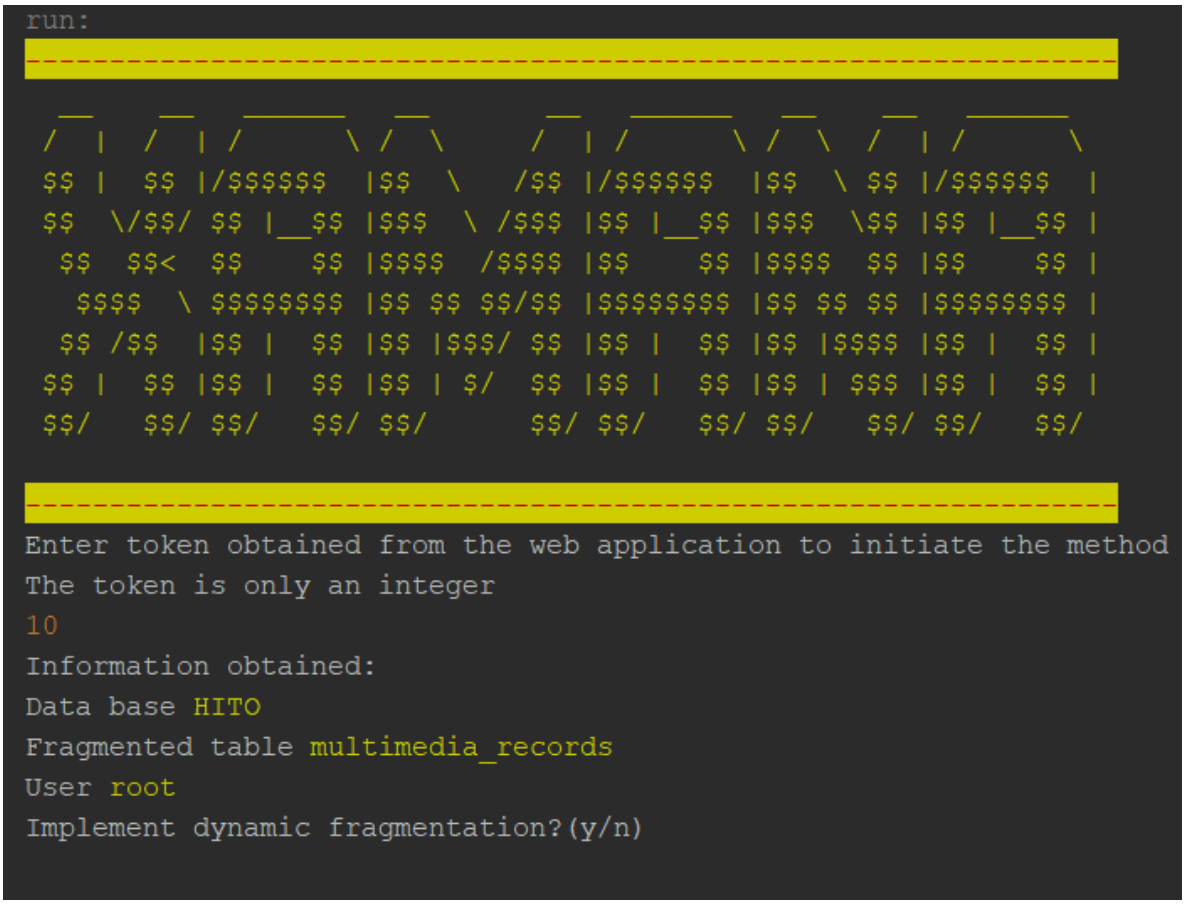


Figura 4.25 Vigilante de fragmentación - información de la fragmentación.

Una vez confirmada la información, el vigilante de fragmentación realiza una petición, para ingresar la ubicación del archivo *log* (Figura 4.26) en donde el gestor de bases de datos registra todas las operaciones que se realizan al esquema, y puede realizar el análisis de esta

información. En este caso se ingresa la ubicación de un archivo *log* (con operaciones ya determinadas) para demostrar el correcto funcionamiento del vigilante de fragmentación.

```
Implement dynamic fragmentation?(y/n)
y
Configure database log
Example database log location: C:/Program Files/Manager/Server/v.v/log/manager.log
Enter database log location:
C:/Logs/example.log
Log file path: C:/Logs/example.log
```

Figura 4.26 Vigilante de fragmentación - ubicación del archivo *log*.

En la Figura 4.27 se observa la ejecución del vigilante de fragmentación en otro sitio (servidor), en este caso en una máquina virtual con el sistema operativo Ubuntu, la ejecución del vigilante de fragmentación en este entorno se hizo a través de la terminal del sistema con ayuda de un archivo JAR. Como se muestra en la Figura 4.27 se ingresó el mismo *token* obtenido de la aplicación web.

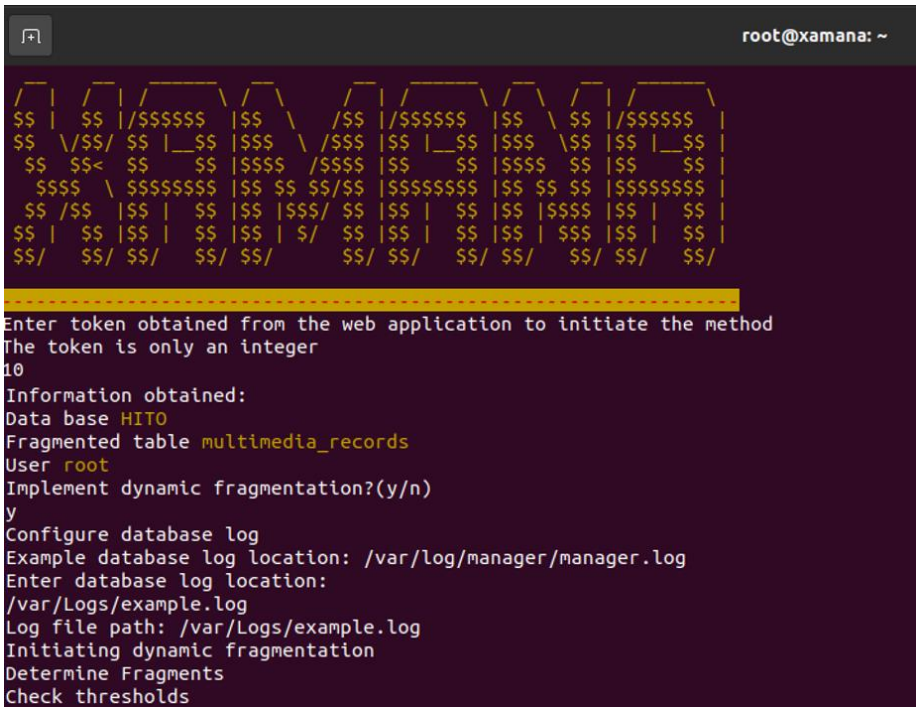


Figura 4.27 Vigilante de fragmentación - ejecución del vigilante de fragmentación en una máquina virtual.

Una vez ingresada la ubicación del archivo *log* el vigilante de fragmentación determina los fragmentos que debe supervisar en el sitio, y calcular los umbrales de cada fragmento, los

umbrales de fragmentación son las operaciones aproximadas y el valor de desempeño que se muestra en la aplicación web, los umbrales se calculan de la siguiente manera.

Cada fragmento tiene un valor de operaciones iniciales y un valor de desempeño inicial, y a cada tabla o colección de datos se le asigna por parte del administrador de base de datos un umbral de operaciones y desempeño, que es un porcentaje de las operaciones iniciales y el valor de desempeño inicial.

Para cada fragmento es necesario determinar su umbral de operaciones de la siguiente manera:

$$\text{Umbral de operaciones} = (PUO * NIO) /100$$

Donde:

PUO es el porcentaje de umbral de operaciones que se asignó en la aplicación web.

NIO es el número inicial de operaciones determinado por la aplicación web.

De igual forma para cada fragmento es necesario determinar el umbral de desempeño, el cual se calcula de la siguiente manera:

$$\text{Umbral de desempeño} = (PUD * VD) / 100$$

Donde:

PUD es el porcentaje de umbral de desempeño que se asignó en la aplicación web.

VD es el valor de desempeño determinado por la aplicación web.

Una vez determinado cada fragmento con su respectivo umbral de operaciones y desempeño, inicia el proceso de verificar los umbrales y determinar cada operación encontrada en el archivo *log* y el vigilante estará haciendo este proceso hasta que alguno de los fragmentos supere sus umbrales. En la Figura 4.28 se presenta una impresión de la información que obtiene el vigilante de fragmentación de cada sitio, cabe resaltar que esta información no se muestra en la versión final del programa. En la imagen se observa una impresión de un puntero. El puntero tiene el objetivo de indicar al vigilante de fragmentación donde debe iniciar su análisis de operaciones, es necesario para evitar que el programa realice un análisis de operaciones o registros anteriores y duplique las operaciones obtenidas.

En la Figura 4.28 se visualiza el nombre de cada fragmento, las operaciones actuales con el umbral de operaciones a superar y el desempeño actual con el umbral de desempeño que debe superar. En el fragmento *multimedia_records_3* se superó el umbral de desempeño, pero no el umbral de operaciones, para que el vigilante de fragmentación actualice el esquema actual es necesario superar ambos umbrales.

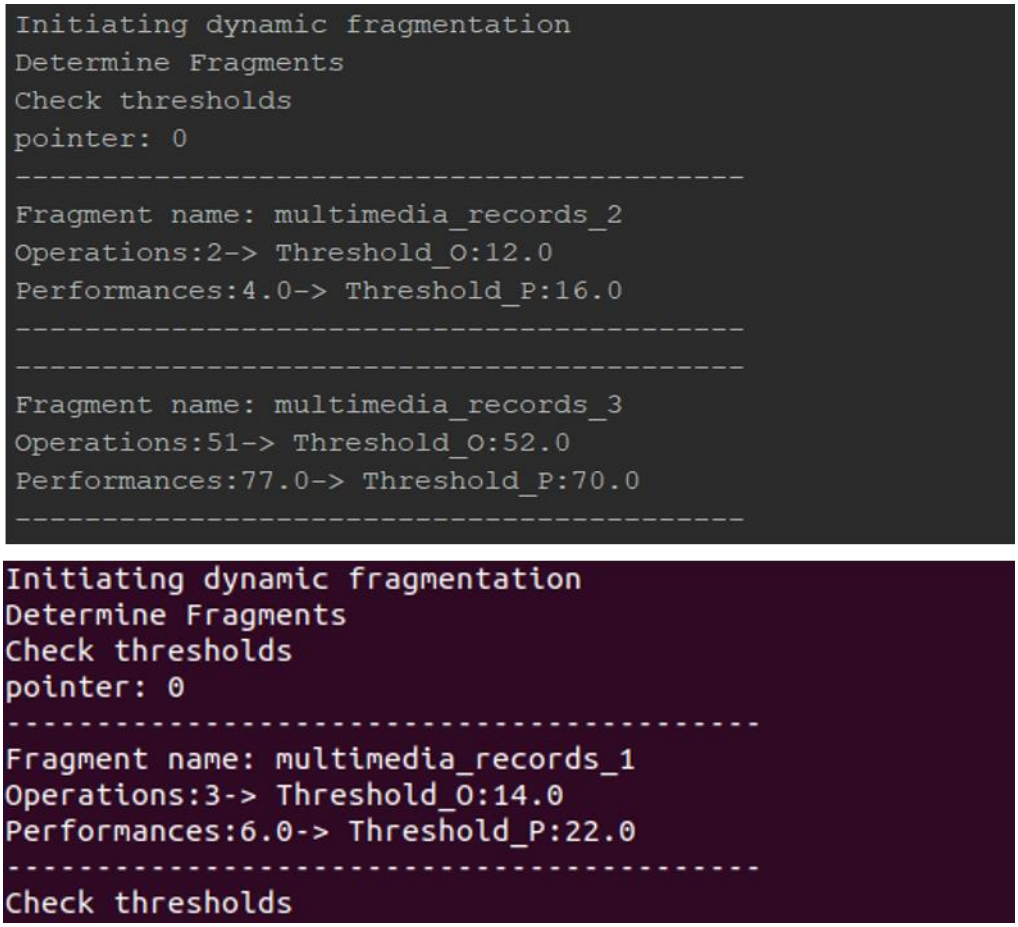


Figura 4.28 Vigilante de fragmentación – información obtenida del vigilante de fragmentación de cada sitio.

En la parte superior de la Figura 4.28 se observa que el vigilante de fragmentación en el sitio uno supervisa dos fragmentos, y en la parte inferior de la Figura 4.28 el vigilante de fragmentación en el sitio dos vigila un solo fragmento. El esquema de la Figura 4.29 representa los sitios y los fragmentos que se encuentran en cada sitio. En el esquema se muestra el predicado de cada fragmento, que es el atributo y el valor que debe tener cada tupla de la colección.

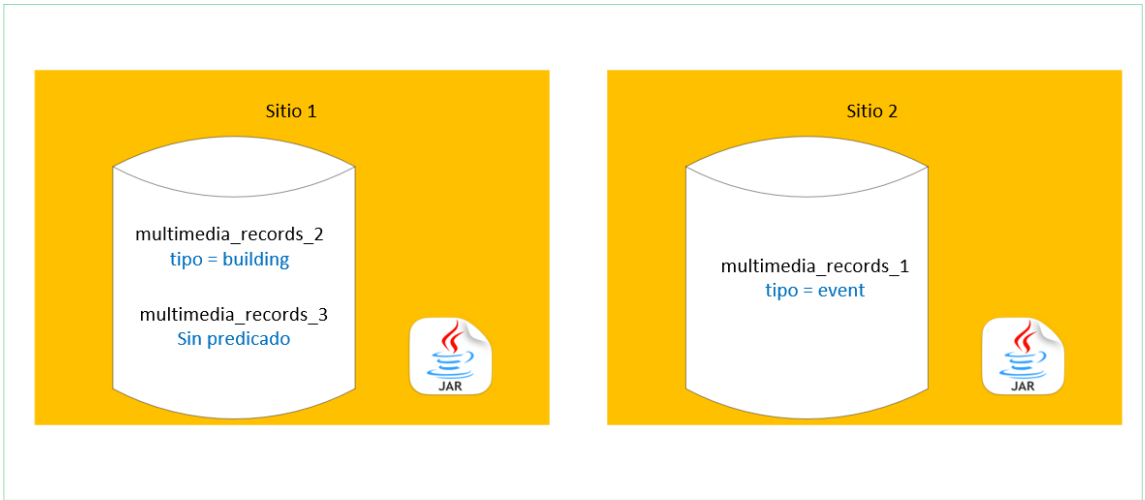


Figura 4.29 Esquema de cada sitio

En la Figura 4.29 se observa que el sitio 1 tiene el fragmento *multimedia_records_3* sin predicado, dado que la metodología es capaz de agrupar las tuplas que no cumplan con un predicado determinado por el análisis de operaciones, tanto en la aplicación web, como en el vigilante de fragmentación.

Si alguno de los fragmentos supera los umbrales de operación, inicia con el proceso de actualizar el esquema (Figura 4.28) de manera interna el vigilante de fragmentación obtiene las frecuencias por sitio, construye la matriz MCRUD, la tabla ALP y determina un nuevo esquema en el sitio, si realizó algún ajuste a un sitio remoto, el vigilante de fragmentación de ese sitio puede determinar los nuevos fragmentos agregados.

En este caso, se agregaron dos nuevas operaciones que corresponden al fragmento *multimedia_records_3*, una vez realizado esto el vigilante de fragmentación evalúa si los dos umbrales fueron superados, si son superados, inicia el proceso de actualizar el esquema como se aprecia en la Figura 4.30. En este caso al log de ejemplo se le agregaron dos líneas que representan operaciones al fragmento *multimedia_records_3*, por lo tanto, el vigilante de fragmentación obtiene estas operaciones y determina que supero los dos umbrales, y da inicio al proceso de actualización de esquema o re-fragmentación

Figura 4.30 Vigilante de fragmentación - actualización del esquema.

Durante el proceso de actualización del esquema, el vigilante de fragmentación verifica el sitio en donde se ubicará cada fragmento, si el sitio no es donde se encuentra el vigilante de fragmentación, actualiza el estado del sitio, en este caso las operaciones del archivo *log* de ejemplo tienen operaciones remotas del sitio dos, por lo tanto, el vigilante de fragmentación del sitio 1 agregará un nuevo fragmento al sitio 2 y actualizará el estado del sitio. El vigilante de fragmentación del sitio 2 detecta esta actualización y comienza a determinar los nuevos como se observa en la Figura 4.31. En este caso el vigilante de fragmentación del sitio 1 determinó que era necesario crear un nuevo fragmento en el sitio 2, en este caso *multimedia_records_3_1*.

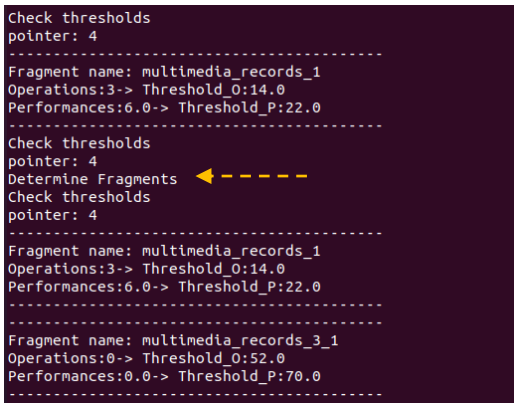


Figura 4.31 Vigilante de fragmentación - actualización de información del sitio 2.

Una vez que el vigilante de fragmentación del sitio 1 termina la actualización del esquema, inicia con el proceso de determinar fragmentos, dado que el fragmento original *multimedia_records_3* es eliminado una vez terminado el proceso de actualización como se observa en la Figura 4.32

```
Update Scheme
Determine Fragments
Check thresholds
pointer: 56
----------------------------------------
Fragment name: multimedia_records_2
Operations:2-> Threshold_O:12.0
Performances:4.0-> Threshold_P:16.0
----------------------------------------
----------------------------------------
Fragment name: multimedia_records_3_2
Operations:0-> Threshold_O:52.0
Performances:0.0-> Threshold_P:70.0
----------------------------------------
```

Figura 4.32 Vigilante de fragmentación - actualización de información del sitio

Por lo tanto, el esquema inicial de todos los sitios se actualiza, como se muestra en la Figura 4.33, ahora existen dos fragmentos en cada sitio, y el fragmento original *multimedia_records_3* es eliminado para no duplicar información.

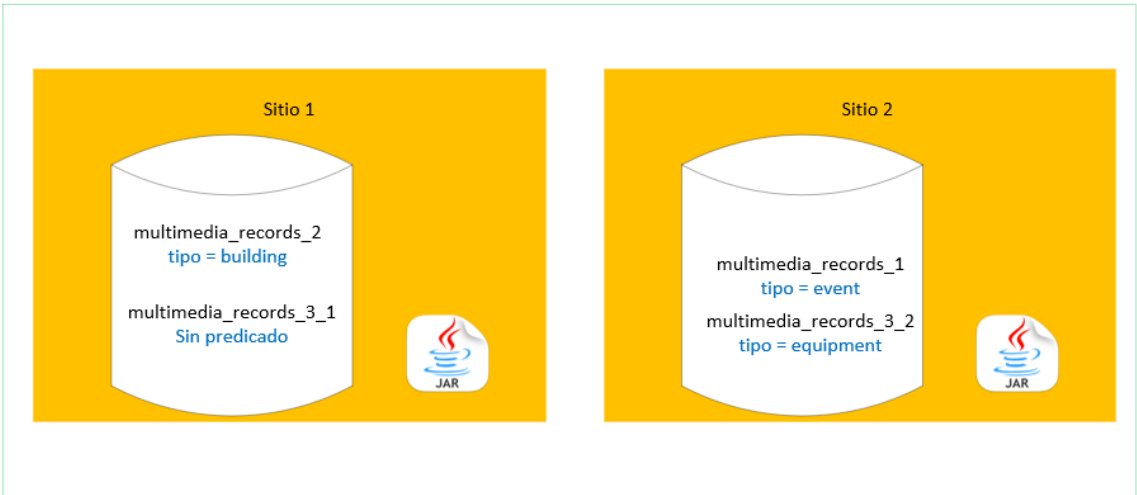


Figura 4.33 Esquema de cada sitio actualizado

Por último, con ayuda de MongoDB Compass se presentan los fragmentos actuales en cada sitio, como se observa en la Figura 4.34.

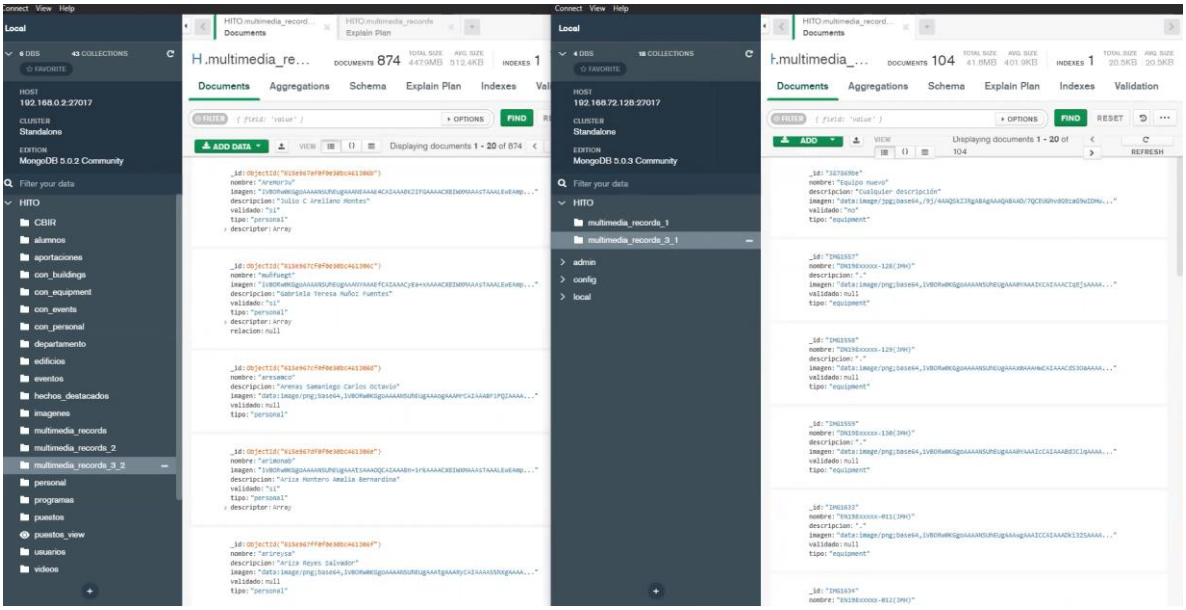


Figura 4.34 MongoDB Compass - fragmentos obtenidos con ayuda del vigilante de fragmentación

4.5 Comparación del rendimiento de la metodología propuesta

Para medir la eficiencia del método desarrollado en este proyecto de tesis, se realizó una comparación de tiempos de ejecución de las consultas más significativas, cuando la colección esta sin fragmentar, fragmentada con la técnica propuesta en este proyecto de tesis, fragmentada con el método propuesto por Castro-Medina et al. [12] y fragmentada por el vigilante de fragmentación de la metodología propuesta (re-fragmentación).

Para ello se tomaron en cuenta las operaciones de la sección 4.4 y se presentan en la Tabla 4.2

Tabla 4.2 Operaciones del archivo *log* HITO

Predicado	Tipo	Sitio	Tuplas afectadas	Frecuencia
Tipo=building	READ	192.168.72.128	168	3
Tipo=building	READ	192.168.0.2	168	2
Tipo=event	READ	192.168.72.128	724	4
Tipo=event	READ	192.168.0.2	724	1
Tipo=event	UPDATE	192.168.0.2	724	2
Tipo=building	UPDATE	192.168.0.2	168	1
Validado=si	DELETE	192.168.0.2	21	2
Validado=si	DELETE	192.168.72.128	21	1
Validado=no	DELETE	192.168.0.2	4	3
Validado=no	DELETE	192.168.72.128	4	1
Validado=no	CREATE	192.168.72.128	2	1
Validado=no	CREATE	192.168.0.2	2	1
Validado=no	CREATE	192.168.0.2	1	1

Se procede a construir la matriz MCRUD con ayuda del método de Castro-Medina et al. [12] cómo se observa en la Tabla 4.3.

Tabla 4.3 Matriz MCRUD de Castro-Medina et al. *[12]*

Predicado	Número de tuplas	Sitio	Costo (Frecuencia * Costo)
Tipo=building	168	192.168.72.128	3 * 1 = 3
Tipo=building	168	192.168.0.2	2 * 1 =2
Tipo=event	724	192.168.72.128	4 * 1 = 4

Predicado	Número de tuplas	Sitio	Costo (Frecuencia * Costo)
Tipo=event	724	192.168.0.2	1 * 1 = 1
Tipo=event	724	192.168.0.2	2 * 3 = 6
Tipo=building	168	192.168.0.2	1 * 3 = 3
Validado=si	21	192.168.0.2	2 * 2 = 4
Validado=si	21	192.168.72.128	2 * 1 = 2
Validado=no	4	192.168.0.2	2 * 3 = 6
Validado=no	4	192.168.72.128	2 * 1 = 2
Validado=no	2	192.168.72.128	2 * 1 = 2
Validado=no	2	192.168.0.2	2 * 1 = 2
Validado=no	1	192.168.0.2	2 * 1 =2

Con ayuda de la matriz MCRUD se procede a construir la tabla ALP, en donde se determina el atributo más costoso del conjunto de operaciones, la Tabla 4.4 presenta los atributos y su respectivo costo.

Tabla 4.4 Tabla ALP de Castro-Medina et al. *[12]*

Atributo	Cálculo	Valor
Tipo	3 + 2 + 4 + 1 + 6 + 3	19
Validado	4 + 2 + 6 + 2 + 2 +2 +2	20

En la Tabla 4.5 se presenta el esquema obtenido con la técnica propuesta por Castro-Medina et al. [12], se obtuvieron 3 fragmentos y todos se alojaron en un solo sitio.

Tabla 4.5 Fragmentos obtenido por Castro-Medina et al. *[12]*

Predicado	Fragmento	Sitio
Validado=si	Multimedia_records_1	192.168.0.2
Validado=no	Multimedia_records_2	192.168.0.2
Sin predicado	Multimedia_records_3	192.168.0.2

Después se procede a implementar la metodología propuesta en este proyecto de tesis, se inicia con el cálculo de costos de cada operación, a diferencia del método de [12], la técnica propuesta en este proyecto sí considera las tuplas afectadas en cada operación. Los costos obtenidos se presentan en la Tabla 4.6.

Tabla 4.6 Matriz MCRUD del método propuesto.

Predicado	Número de tuplas	Sitio	Costo (Frecuencia * Costo * tupla)
Tipo=building	168	192.168.72.128	3 * 1 * 168 = 504
Tipo=building	168	192.168.0.2	2 * 1 * 168 = 336
Tipo=event	724	192.168.72.128	4 * 1 * 724 = 2,896
Tipo=event	724	192.168.0.2	1 * 1 * 724 = 724
Tipo=event	724	192.168.0.2	2 * 3 * 724 = 4,344
Tipo=building	168	192.168.0.2	1 * 3 * 168 = 504
Validado=si	21	192.168.0.2	2 * 2 * 21 = 84
Validado=si	21	192.168.72.128	2 * 1 * 21 = 42
Validado=no	4	192.168.0.2	2 * 3 * 4 = 24
Validado=no	4	192.168.72.128	2 * 1 * 4 = 8
Validado=no	2	192.168.72.128	2 * 1 * 2 = 4
Validado=no	2	192.168.0.2	2 * 1 * 2 = 4
Validado=no	1	192.168.0.2	2 * 1 * 1 = 4

Posteriormente se procede a construir la tabla ALP, como se observa, en la Tabla 4.7 muestra los atributos considerados en el cálculo de cada valor y el costo de cada uno.

Tabla 4.7 Tabla ALP del método propuesto.

Atributo	Cálculo	Valor
Tipo	504 + 336 + 2,896 + 724 + 4,344 + 504	8,804
Validado	84 + 42 + 24 + 8 + 4 + 4 + 4	170

En la Tabla 4.7 se aprecia una mayor diferencia de costos, dado que la técnica propuesta toma en cuenta el contenido de cada tupla, debido a que puede existir o no un objeto multimedia, los cuales se caracterizan por ser objetos con un mayor tamaño, por toda la información que se necesita para representar dicho objeto. En la Tabla 4.8 se presenta el esquema propuesto por el método de este proyecto de tesis.

Tabla 4.8 Fragmentos obtenido por el método propuesto.

Predicado	Fragmento	Sitio
Tipo=event	Multimedia_records_1	192.168.72.128
Tipo=building	Multimedia_records_2	192.168.0.2
Sin predicado	Multimedia_records_3	192.168.0.2

En la Figura 4.35 se presenta los esquemas propuestos por cada método. En ella se observan los esquemas obtenidos por cada enfoque (4.35b y 4.35c), de igual forma se presenta el esquema re-fragmentado por el método propuesto a través del vigilante de fragmentación (4.35d).

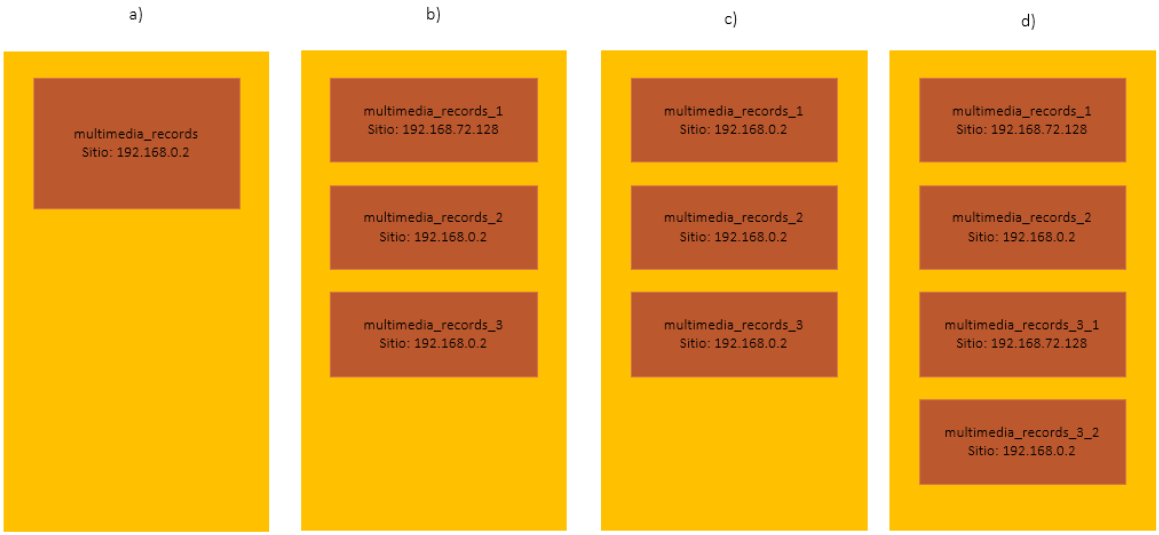


Figura 4.35 a) Sin fragmentar, b) Esquema obtenido de la técnica propuesta, c) Esquema obtenido del método de Castro-Medina et al. *[12]* y d) Esquema obtenido del método propuesto (re-fragmentación).

Por último, se presenta la Tabla 4.9 en donde se muestran las consultas más significativas para la colección de datos sin fragmentar y los tiempos de respuesta en cada caso.

Tabla 4.9 Tiempos de ejecución de consultas.

Consulta	Sin fragmentar	Método propuesto	Método de Castro Medina et al. [12]	Método propuesto (re-fragmentación)
Select * from multimedia_records where tipo= event	0.002048	**0.001092**	0.0035600	**0.001092**
Select * from multimedia_records where tipo= building	0.0045600	**0.001304**	0.0038900	**0.001304**
Select * from multimedia_records where tipo= equipment	0.001224	0.000889	0.0037900	**0.000856**
Select * from multimedia_records where tipo= personal	0.001136	0.000889	0.0038900	**0.000824**

Como se aprecia en la Tabla 4.9 los tiempos de respuesta mejoran con la colección fragmentada por el método propuesto y el método de Castro-Medina, aunque los tiempos de respuesta son aún menores con el enfoque propuesto en este proyecto de tesis, dado el enfoque de este, que toma en cuenta los objetos de tipo multimedia, y esos tiempos aún mejoran cuando se aplica la parte dinámica de la técnica a través del vigilante de fragmentación, siendo evidentes los beneficios de un método que es capaz de adaptar el esquema según los patrones de acceso a la colección de datos.

Capítulo 5. Conclusiones y recomendaciones

5.1 Conclusiones

Los métodos de fragmentación horizontal son muy utilizados en la industria. Existe un gran número de métodos y aplicaciones que facilitan la implementación de dichos métodos. La implementación de estas técnicas mejora el rendimiento de las bases de datos al ubicar la información en los sitios en donde más se utilizan.

A pesar de que existe un gran número de métodos de fragmentación horizontal, lamentablemente la mayoría no considera el contenido multimedia, y si lo hace, es de una forma estática. Hoy en día, con la existencia de múltiples aplicaciones de varios dominios, como aplicaciones para dispositivos móviles, aplicaciones web y aplicaciones de escritorio, se genera una gran cantidad de datos a un ritmo muy rápido, de los cuales destaca el contenido multimedia, que al incluir objetos con un mayor tamaño (debido a toda la información que se necesita para representar un objeto multimedia) provoca un crecimiento muy rápido a las bases de datos multimedia. Estas aplicaciones no solo generan datos, de igual forma necesitan recopilar (recuperar) toda esta información generada para su uso, y como constantemente se encuentran generando datos, la manera en cómo se accede a la información cambia constantemente, por lo tanto, si se realiza una fragmentación estática a una base de datos de contenido multimedia, el esquema resultante se degrada a un ritmo muy rápido, provocando una disminución en el rendimiento de la base de datos. Para resolver este problema, en este proyecto, se desarrolló un método de fragmentación dinámica para bases de datos multimedia.

Los resultados obtenidos con este trabajo beneficiarán en gran medida a los proveedores de almacenamiento de contenido multimedia, a los investigadores que necesiten utilizar algún método de fragmentación horizontal dinámica para bases de datos multimedia o bases de datos que no contemplan dicho contenido, dado que el enfoque de la técnica propuesta contempla la existencia o no de los datos multimedia.

Una vez realizado el análisis de trabajos relacionados, se desarrolló el método de fragmentación horizontal para base de datos multimedia por medio de una aplicación web y

un vigilante de fragmentación. La aplicación web implementa la parte estática del método con ayuda de un análisis de un archivo *log* (historial de operaciones de la base de datos). El vigilante de fragmentación realiza la parte dinámica del método, el cual supervisa de manera independiente los fragmentos que se ubiquen en cada sitio y con ayuda del *log* del gestor de bases de datos que se encuentre en el sitio, determina las operaciones que se realicen a cada fragmento y si es necesario realizar un re-fragmentación al esquema actual.

La técnica propuesta en este trabajo intenta facilitar la fragmentación de datos y automatizar dicho proceso para reducir la complejidad de la fragmentación y mantener el rendimiento de la base de datos al actualizar constantemente el esquema si es necesario.

5.2 Recomendaciones

Al aplicar el método en un entorno real, se observaron sus ventajas y desventajas. Por lo tanto, como continuación de esta investigación, sería interesante incluir un mecanismo que permita agrupar información de fragmentos en nuevos fragmentos tomando en cuenta las operaciones analizadas por el vigilante de fragmentación, para no realizar una derivación de un solo fragmento, si no construir fragmentos con información de varios fragmentos dispersos en todos los sitios de una red.

Además, como trabajo a futuro se sugiere que se realice un mecanismo o programa que sea capaz de revertir el esquema propuesto a la tabla o colección de datos original, esto con el objetivo de facilitar el trabajo a los administradores de bases de datos que deseen ya no implementar la técnica y pueda restablecer el esquema original de su base de datos.

Productos académicos

Abraham Castillo García, Lisbeth Rodríguez Mazahua, Felipe Castro Medina, Beatriz A. Olivares Zepahua, María A. Abud Figueroa.

Design of a dynamic horizontal fragmentation method for multimedia databases.

2nd International Workshop on Enterprise Decision-Making Applying Artificial Intelligence Techniques (WEDMAIT 2021).

Estado: Presentado.

Abraham Castillo García, Lisbeth Rodríguez Mazahua, Felipe Castro Medina, Beatriz A. Olivares Zepahua, María A. Abud Figueroa.

A review of horizontal fragmentation methods considering multimedia data and dynamic access patterns.

10th International Conference on Software Processes Improvement (ISBN: 978-3-030-89908-0).

Páginas: 69-82

Estado: Publicado.

Referencias

[1] G. Zhen, Z. M. Zhang, E. P. Xing y C. Faloutsos, «Multimodal Data Mining in a Multimedia Database Based on Structured Max Margin Learning,» *ACM Transactions on Knowledge Discovery from Data*, vol. 10, nº 3, pp. 23:1-23:30, 2016.

[2] M. T. Özsu y P. Valduriez, Distributed and Parallel Database Design, Cham: Springer International Publishing, 2020.

[3] D. Bell y J. Grimson, Distributed Database System, Addison Wesley, 1992, pp. 122-126.

[4] R. Ramachandran, D. P. Nair y J. Jasmi, «A horizontal fragmentation method based on data semantics,» de *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 2016.

[5] M. M. Patil, A. Hanni, C. H. Tejeshwar y P. Patil, «A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing — Sharding in MongoDB and its advantages,» de *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017.

[6] D. Liming, L. Weindong y S. Jie, «Coexistence of Multiple Partition Plan Based Physical Database Design,» de *Proceedings of the 5th International Conference on Communications and Broadband Networking*, New York, NY, USA, 2017.

[7] M. Guo y H. Kang, «The Implementation of Database Partitioning Based on Streaming Framework,» de *2016 13th Web Information Systems and Applications Conference (WISA)*, 2016.

[8] Z. Chen, L. Zhang, S. Yang, S. Tan, L. He, G. Zhang y H. Yang, «The data partition strategy based on hybrid range consistent hash in NoSQL database,» de *Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service*, New York, NY, USA, 2013.

[9] J. M. M. Kamal, M. Murshed y R. Buyya, «Workload-Aware Incremental Repartitioning of Shared-Nothing Distributed Databases for Scalable Cloud Applications,» de *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014.

[10] A. Pavlo, C. Curino y S. Zdonik, «Skew-aware automatic database partitioning in shared-nothing, parallel OLTP systems,» de *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2012.

[11] E. Zamanian, J. Shun, C. Binnig y T. Kraska, «Chiller: Contention-centric Transaction Execution and Data Partitioning for Modern Networks,» de *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2020.

[12] F. Castro-Medina, L. Rodriguez-Mazahua, A. López-Chau, M. A. Abud-Figueroa y G. Alor-Hernández, «FRAGMENT: A Web Application for Database Fragmentation, Allocation and Replication over a Cloud Environment,» *IEEE Latin America Transactions,* vol. 18, nº 6, pp. 1126-1134, 2020.

[13] M. . J. Rodríguez Arauz, L. Rodriguez-Mazahua, M. L. Arrioja-Rodríguez, M. A. Abud-Figueroa, S. G. Peláez-Camarena y L. d. C. Martínez-Méndez, «Design of a Multimedia Data Management System that Uses Horizontal Fragmentation to Optimize Content-based Queries,» de *IMMM 2020, The Tenth International Conference on Advances in Information Mining and Management*, 2020.

[14] R. Taft, E. Mansour, M. Serafini, J. Duggan, A. J. Elmore, A. Aboulnaga, A. Pavlo y M. Stonebraker, «E-store: fine-grained elastic partitioning for distributed transaction processing systems,» *Proceedings of the VLDB Endowment,* vol. 8, nº 3, p. 245–256, 2014.

[15] A. Quamar, K. A. Kumar y A. Deshpande, «SWORD: scalable workload-aware data placement for transactional workloads,» de *Proceedings of the 16th International Conference on Extending Database Technology*, 2013.

[16] C. Curino, E. Jones, Y. Zhang y S. Madden, «Schism: a workload-driven approach to database replication and partitioning,» *Proceedings of the VLDB Endowment,* vol. 3, nº 1, pp. 48-57, 2010.

[17] M. Abebe, B. Glasbergen y K. Daudjee, «MorphoSys: automatic physical design metamorphosis for distributed database systems,» *Proceedings of the VLDB Endowment,* vol. 13, nº 13, p. 3573–3587, 2020.

[18] K. Tzoumas, A. Deshpande y C. S. Jensen, «Sharing-aware horizontal partitioning for exploiting correlations during query processing,» *Proceedings of the VLDB Endowment,* vol. 3, nº 1, p. 542–553, 2010.

[19] E. Zamanian, C. Binnig y A. Salama, «Locality-aware Partitioning in Parallel Database Systems,» de *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2015.

[20] R. Marcus, O. Papaemmanouil, S. Semenova y S. Garber, «NashDB: An End-to-End Economic Method for Elastic Database Fragmentation, Replication, and Provisioning,» de *Proceedings of the 2018 International Conference on Management of Data*, New York, NY, USA, 2018.

[21] L. Rodríguez-Mazahua, G. Alor-Hernández, M. A. Abud-Figueroa y S. . G. Peláez-Camarena, «Horizontal Partitioning of Multimedia Databases Using Hierarchical Agglomerative Clustering,» de *Nature-Inspired Computation and Machine Learning*, Cham, 2014.

[22] «JavaServer Faces Technology,» Oracle, [En línea]. Available: https://www.oracle.com/java/technologies/javaserverfaces.html. [Último acceso: 25 Febrero 2021].

[23] «NetBeans IDE - Overview,» [En línea]. Available: https://netbeans.org/features/index.html. [Último acceso: 25 Febrero 2021].

[24] C. Nieves Guerrero, J. Ucán Pech y V. Menéndez Domínguez, «UWE en Sistema de Recomendación de Objetos de Aprendizaje. Aplicando Ingeniería Web: Un Método en Caso de Estudio,» *REVISTA LATINOAMERICANA DE INGENIERIA DE SOFTWARE*, vol. 2, nº 1, pp. 137-143, 2014.

[25] «What Is MongoDB?,» [En línea]. Available: https://www.mongodb.com/what-is-mongodb. [Último acceso: 25 Febrero 2021].

[26] «Deploy Cloud Applications with MySQL Database,» [En línea]. Available: https://www.oracle.com/mysql/. [Último acceso: 25 Febrero 2021].

[27] «Conozca más sobre la tecnología Java,» [En línea]. Available: https://www.java.com/es/about/. [Último acceso: 25 Febrero 2021].

[28] M. Serafini, R. Taft, A. J. Elmore, A. Pavlo, A. Aboulnaga y M. Stonebraker, «Clay: fine-grained adaptive partitioning for general database schemas,» *Proceedings of the VLDB Endowment*, vol. 10, nº 4, pp. 445-456, 2016.

[29] H. I. Abdalla y A. A. Amer, «Dynamic horizontal fragmentation, replication and allocation model in DDBSs,» de *2012 International Conference on Information Technology and e-Services*, 2012.

[30] K. Fasolin, R. Fileto, M. Krugery, D. S. Kasterz, M. . R. P. Ferreirax, R. . L. F. Cordeirox, A. J. M. Trainax y C. Traina, «Efficient Execution of Conjunctive Complex Queries on Big Multimedia Databases,» de *2013 IEEE International Symposium on Multimedia*, 2013.

[31] L. Lim, «Elastic data partitioning for cloud-based SQL processing systems,» de *2013 IEEE International Conference on Big Data*, 2013.

[32] K. Herrmann, H. Voigt y W. Lehner, «Cinderella — Adaptive online partitioning of irregularly structured data,» de *2014 IEEE 30th International Conference on Data Engineering Workshops*, 2014.

[33] R. Kumar y N. Gupta, «An extended approach to Non-Replicated dynamic fragment allocation in distributed database systems,» de *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, 2014.

[34] I. Fetai, D. Murezzan y H. Schuldt, «Workload-driven adaptive data partitioning and distribution — The Cumulus approach,» de *2015 IEEE International Conference on Big Data (Big Data)*, 2015.

[35] B. Sauer y W. Hao, «Horizontal cloud database partitioning with data mining techniques,» de *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 2015.

[36] Q. Wu, C. Chen y Y. Jiang, «Multi-source heterogeneous Hakka culture heritage data management based on MongoDB,» de *2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*, 2016.

[37] B. Oonhawat y N. Nupairoj, «Hotspot management strategy for real-time log data in MongoDB,» de *2017 19th International Conference on Advanced Communication Technology (ICACT)*, 2017.

[38] N. K. Z. Lwin y T. M. Naing, «Non-Redundant Dynamic Fragment Allocation with Horizontal Partition in Distributed Database System,» de *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, 2018.

[39] P. Peng, L. Zou, L. Chen y D. Zhao, «Adaptive Distributed RDF Graph Fragmentation and Allocation based on Query Workload,» *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, nº 4, pp. 670-685, 2019.

[40] A. A. Amer, A. A. Sewisy y T. M. A. Elgendy, «An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems (DDBSs),» *Heliyon*, vol. 3, nº 12, 2017.

[41] H. I. Abdalla, «A synchronized design technique for efficient data distribution,» *Computers in Human Behavior*, vol. 30, nº 1, pp. 427-435, 2014.

[42] Z. Goli-Malekabadi, M. Sargolzaei-Javan y M. K. Akbari, «An effective model for store and retrieve big health data in cloud computing,» *Computer Methods and Programs in Biomedicine*, vol. 132, pp. 75-82, 2016.

[43] W. Wedashwara, S. Mabu, M. Obayashi y T. Masanao, «Combination of genetic network programming and knapsack problem to support record clustering on distributed databases,» *Expert Systems with Applications*, vol. 46, pp. 15-23, 2016.

[44] J. O. Hauglid, N. H. Ryeng y K. Nørvåg, «DYFRAM: dynamic fragmentation and replica management in distributed database systems,» *Distributed and Parallel Databases*, pp. 157-185, 2010.

[45] M. Liroz-Gistau, R. Akbarinia, E. Pacitti, F. Porto, P. Valduriez, S. W. Liddle, K.-D. Schewe, A. M. Tjoa y X. Zhou, «Dynamic Workload-Based Partitioning for Large-Scale Databases,» de *Database and Expert Systems Applications*, Berlin, Heidelberg, 2012.

[46] M. Liroz-Gistau, R. Akbarinia, E. Pacitti, F. Porto, P. Valduriez, A. Hameurlain, J. Küng y R. Wagner, «Dynamic Workload-Based Partitioning Algorithms for Continuously Growing Databases,» de *Transactions on Large-Scale Data- and Knowledge-Centered Systems XII*, Berlin, Heidelberg, Springer, 2013, pp. 105-128.

[47] A. E. Abdel Raouf, N. L. Badr y M. F. Tolba, «Distributed Database System (DSS) Design Over a Cloud Environment,» de *Multimedia Forensics and Security: Foundations, Innovations, and Applications*, Cham, Springer International Publishing, 2016, pp. 97-116.

[48] S. M. Elghamrawy, «An Adaptive Load-Balanced Partitioning Module in Cassandra Using Rendezvous Hashing,» de *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016*, Cham, 2016.

[49] S. M. Elghamrawy y A. E. Hassanien, «A partitioning framework for Cassandra NoSQL database using Rendezvous hashing,» *The Journal of Supercomputing*, vol. 73, nº 10, pp. 4444-4465, 2017.

[50] M. Olma, M. Karpathiotakis, I. Alagiannis, M. Athanassoulis y A. Ailamaki, «Adaptive partitioning and indexing for in situ query processing,» *The VLDB Journal*, vol. 29, nº 1, pp. 569-591, 2020.

[51] C. Salmi, M. Chaabani y M. Mezghiche, «A Formalized Procedure for Database Horizontal Fragmentation in Isabelle/HOL Proof Assistant,» de *Model and Data Engineering*, Cham, 2018.

[52] I. Feinerer, E. Franconi y P. Guagliardo, «Lossless Horizontal Decomposition with Domain Constraints on Interpreted Attributes,» de *Big Data*, Berlin, Heidelberg, 2013.

[53] D. Teng, J. Kong y F. Wang, «Scalable and flexible management of medical image big data,» *Distributed and Parallel Databases*, vol. 67, nº 2, pp. 235-250, 2016.

[54] C. Baron y N. M. Iacob, «A New Dynamic Data Fragmentation and Replication Model in DDBMSs. Cost Functions,» *Knowledge Horizons - Economics*, vol. 6, nº 1, pp. 158-161, 2014.

[55] S. I. Khan, «Efficient Partitioning of Large Databases without Query Statistics,» *Database Systems Journal,* vol. 7, nº 2, pp. 34-53, 2016.