DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

OPCIÓN 1.- TESIS

TRABAJO PROFESIONAL

"DESARROLLO DE NUEVOS MÉTODOS DE FRAGMENTACIÓN DINÁMICA PARA BASES DE DATOS MULTIMEDIA"

QUE PARA OBTENER EL GRADO DE:
DOCTOR EN CIENCIAS
DE LA INGENIERÍA

PRESENTA:

M. S. C. FELIPE CASTRO MEDINA

DIRECTOR DE TESIS:
DRA. LISBETH RODRÍGUEZ MAZAHUA

CODIRECTOR DE TESIS:
DR. ASDRÚBAL LÓPEZ CHAU

ORIZABA, VER. MÉXICO DICIEMBRE 2023

Orizaba, Veracruz, 08/02/2024
Dependencia: **División de Estudios de Posgrado e Investigación**
Asunto: **Autorización de Impresión**
OPCION: I

C. FELIPE CASTRO MEDINA
Candidato a Grado de Doctor en:
CIENCIAS DE LA INGENIERÍA
P R E S E N T E.-

De acuerdo con el Reglamento de Titulación vigente de los Centros e Institutos Tecnológicos Federales del Tecnológico Nacional de México, 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 NUEVOS MÉTODOS DE FRAGMENTACIÓN 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®

DRA. OFELIA LANDETA ESCAMILLA
ENCARGADA DE LA JEFATRUA DE LA DIVISIÓN
DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

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

ÍNDICE

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La fragmentación y la asignación de datos son temas de gran interés en la industria y la academia. La automatización y adaptación de los esquemas de bases de datos a patrones de consultas en constante cambio hacen de la fragmentación dinámica un tópico para el análisis y desarrollo de nuevos enfoques que mejoren el desempeño de las consultas en sistemas distribuidos.

El almacenamiento de datos en los SGBD (Sistemas Gestores de Bases de Datos) presenta en la actualidad nuevos retos y oportunidades frente al contenido multimedia. El objetivo de este trabajo es proponer tres nuevos métodos que desempeñen la fragmentación horizontal, vertical e híbrida, dinámica contemplando CBIR (*Content-Based Image Retrieval*, Consulta de Imágenes Basada en Contenido). Cada método se enfoca en un tipo de fragmentación, incluyendo flujos de trabajo para CBIR y consultas tradicionales. De igual forma, la fragmentación estática se incluye como opción al decidir no aplicar la fragmentación dinámica. Finalmente, la validación de los métodos se lleva a cabo utilizando bases de datos multimedia.

Para alcanzar los objetivos de esta investigación se presenta XAMANA, una aplicación Web que integra los métodos desarrollados. Las tecnologías que se utilizaron para la creación de XAMANA son como marco de trabajo *JavaServer Faces* con la biblioteca *PrimeFaces*, como gestor de bases de datos MySQL y como IDE (*Integrated Development Environment*, entorno de desarrollo integrado) NetBeans, ya que este es el entorno que mejor se adecua al desarrollo con estas tecnologías. Los métodos de fragmentación permiten que el usuario utilice tres SGBDs diferentes a MySQL: PostgreSQL, Postgres-XL y MongoDB.

Palabras Clave: Fragmentación dinámica, CBIR, Sistemas distribuidos

ABSTRACT

Data fragmentation and allocation are issues of great importance in industry and academia. The automation and adaptation of database schemes to constantly changing access patterns make dynamic fragmentation a topic for the analysis and development of new approaches that improve query performance in distributed systems.

Data storage in DBMS (Database Management Systems) currently presents new challenges and opportunities in the face of multimedia content. The objective of this work is to propose three new methods that perform dynamic horizontal, vertical and hybrid fragmentation contemplating CBIR (Content-Based Image Retrieval). Each method focuses on one type of fragmentation, including workflows for CBIR and traditional queries. In the same way, static fragmentation is included as an option when deciding not to apply dynamic fragmentation. Finally, the validation of the methods is carried out using multimedia databases.

To achieve the objectives of this research, XAMANA is presented, a Web application that integrates the developed methods. The technologies that were used for the creation of XAMANA are JavaServer Faces as framework with the PrimeFaces library, MySQL as DBMS and NetBeans as IDE (Integrated Development Environment), because this is the environment that best adapts to the development with these technologies. The fragmentation methods allow the user to utilize three DBMS different from MySQL: PostgreSQL, Postgres-XL and MongoDB.

Keywords: Dynamic Fragmentation, CBIR, Distributed Systems

INTRODUCCIÓN

La fragmentación es un tema esencial en las bases de datos distribuidas, ya que actualmente las demandas de información crecen rápidamente y mejorar el desempeño de los esquemas se convierte en una necesidad. La asignación de datos, de igual manera, es un tema crucial para reducir los tiempos de respuesta de las consultas.

La fragmentación de bases de datos multimedia es un tema de gran interés. Mediante la fragmentación se logra mejorar el rendimiento de la ejecución de consultas complejas que obtienen objetos multimedia. Sin una estrategia de almacenamiento, los objetos multimedia complejos se almacenan de manera secuencial y las consultas se procesan de la misma forma. El costo de la ejecución de consultas complejas de objetos multimedia usando búsqueda secuencial es muy alto, especialmente cuando los datos multimedia son grandes, lo cual es usual (Fung, Leung & Li, et al., 2003).

Por otro lado, en la fragmentación estática, los datos o elementos de una tabla (atributos y/o tuplas) se asignan a un fragmento solo una vez en el momento de la creación, luego sus ubicaciones nunca cambian. Este enfoque tiene los siguientes problemas:

- El administrador de la base de datos (DBA) debe observar el sistema durante un período de tiempo significativo antes de que la operación de fragmentación pueda llevarse a cabo, hasta que se descubran las probabilidades de que las consultas accedan a ciertos elementos de la base de datos, así como las frecuencias de dichas consultas.
- Después de que se complete el proceso de fragmentación, nada garantiza que se hayan descubierto las tendencias reales en las consultas y los datos. Por lo tanto, el esquema de fragmentación puede no ser el adecuado. En este caso, los usuarios de la base de datos experimentarán un tiempo de respuesta de consultas muy largo.
- En algunas aplicaciones dinámicas, las consultas tienden a cambiar con el tiempo y se implementa un esquema de fragmentación para optimizar el tiempo de respuesta para un conjunto particular de consultas. Por lo tanto, si las consultas o sus frecuencias relativas cambian, el resultado de la fragmentación puede no ser adecuado.

- En los métodos de fragmentación estática, la refragmentación es una tarea pesada y solo se realiza manualmente cuando el sistema está inactivo.

Por el contrario, en la fragmentación dinámica, los atributos se reubican si se detecta que el esquema actual de fragmentación se ha vuelto inadecuado debido a cambios en la información de las consultas (Rodríguez et al., 2013).

Esta investigación propone un enfoque para mejorar el desempeño de bases de datos multimedia. Las características de este trabajo destacan frente a diferentes propuestas presentes hasta ahora en la literatura, ya que ninguno las incluye todas. Este estudio beneficia principalmente a los DBA (*Database Administrator*, Administrador de bases de datos), pero también a todos los usuarios que buscan mejorar el desempeño de bases de datos en un entorno distribuido.

Este trabajo aborda un análisis profundo del estado del arte, comparando las técnicas de fragmentación y analizando diferentes aspectos que este trabajo considera relevantes. El enfoque propuesto se compara con las características presentes en los demás trabajos para describir detalladamente el alcance de este.

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 contiene el diseño de los diagramas sugeridos por la metodología de desarrollo UWE (*UML-based Web Engineering*, Ingeniería Web Basada en UML), la presentación de la aplicación Web y los módulos para la fragmentación dinámica; el capítulo cuatro describe los resultados obtenidos por los enfoques de fragmentación propuestos y la descripción de los casos de estudio para validarlos; por último, en el capítulo cinco se presentan las conclusiones y recomendaciones.

PLANTEAMIENTO DEL PROBLEMA

El problema principal observado en esta investigación es que hasta ahora se reportan en la literatura escasos estudios que presenten métodos de fragmentación dinámica para bases de datos multimedia y que tomen en cuenta consultas basadas en contenido. Los métodos de recuperación de datos multimedia basados en contenido mejoran la precisión de las búsquedas que se realizan en la base de datos. La extracción, clasificación y manipulación automática del contenido multimedia son de importancia crítica para una gestión eficiente de datos multimedia (Nam et al., 2005).

Los métodos que incluyen CBIR son necesarios cuando las anotaciones textuales son inexistentes o incompletas (Springmann, Kopp & Schuldt, 2010). Además, los métodos basados en contenido mejoran potencialmente la precisión en la recuperación incluso cuando las anotaciones textuales están presentes, ya que dan un conocimiento adicional a las colecciones de datos multimedia (Rojas Ruiz, 2018).

Sin embargo, los métodos basados en contenido deben tratar con problemas de eficiencia y escalabilidad. Para resolver estos problemas recientemente se han aplicado métodos de fragmentación en bases de datos multimedia, no obstante se presentan dos problemas en la utilización de técnicas de fragmentación en bases de datos multimedia:

- La mayoría de los algoritmos existentes no toman en cuenta consultas basadas en contenido (Fung et al., 2002), (Fung, Leung & Li, 2003), (Fung, Karlapalem y Li, 2003), (Rodríguez y Li, 2011a), (Rodríguez-Mazahua et al., 2014), (Rodríguez-Mazahua et al., 2016), (Rodríguez-Mazahua et al., 2017).
- Los algoritmos que sí consideran consultas basadas en contenido, llevan a cabo una fragmentación estática (Saad et al., 2006), (Getahun et al., 2007a), (Getahun et al., 2007b), (Chbeir y Laurent, 2010).

Para resolver estos problemas, este trabajo utiliza los tres tipos de fragmentación de forma dinámica para determinar cuándo realizar nuevas fragmentaciones a lo largo del tiempo en una base de datos distribuida, evitando así la importante cantidad de tiempo que dedica el administrador de la base de datos a observar el sistema para recolectar las frecuencias de las

operaciones realizadas, garantizando el uso de tendencias reales, para reducir los tiempos de ejecución de las consultas, adaptando el esquema a los cambios en los patrones de acceso y haciendo de la refragmentación una tarea trivial, autónoma y ejecutada sobre la marcha.

JUSTIFICACIÓN

Los datos multimedia son de importancia clave en muchas áreas de aplicación como medicina, cartografía, meteorología, seguridad, entre otras (Lew et al., 2006). Las bases de datos multimedia en un ambiente distribuido necesitan un diseño adecuado para desempeñarse adecuadamente. La fragmentación de bases de datos es una tarea difícil que el administrador realiza. Elegir una técnica adecuada de fragmentación (horizontal, vertical o híbrida) y saber cuándo aplicarla nuevamente es un trabajo que debe ser automatizado para que el sesgo de error se minimice y el desempeño del sistema no se comprometa. Mediante este trabajo se beneficiarán los investigadores del área de bases de datos, ya que ahora cuentan con métodos de fragmentación dinámica para bases de datos multimedia que consideran consultas basadas en contenido, así como los diseñadores y los administradores de bases de datos, ya que las tareas de fragmentación se delegarán a la aplicación propuesta en este trabajo.

La justificación de esta investigación es la falta de métodos de fragmentación de bases de datos multimedia dinámicos que estén orientados a las consultas basadas en contenido. Diferentes autores proponen métodos de fragmentación para bases de datos multimedia, sin embargo, sólo un trabajo aborda la fragmentación dinámica de bases de datos multimedia (Rodríguez-Mazahua et al., 2017), pero no contempla consultas basadas en contenido. Los métodos propuestos producen esquemas de fragmentación que reducen el tiempo de respuesta y el costo de ejecución de consultas en bases de datos multimedia.

HIPÓTESIS

Los métodos de fragmentación dinámica para bases de datos multimedia desarrollados adaptan el esquema de fragmentación de acuerdo al cambio en los patrones de acceso para reducir el tiempo de respuesta y el costo de ejecución de las consultas basadas en contenido.

OBJETIVOS

Objetivo general

Desarrollar un conjunto de nuevos métodos para la fragmentación vertical, horizontal e híbrida dinámica de bases de datos multimedia que optimicen consultas basadas en contenido.

Objetivos específicos

- Estudiar y analizar el estado del arte de los métodos de fragmentación horizontal, vertical e híbridos, 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 seleccionar los algoritmos que se utilizarán para compararlos con el método propuesto.
- Diseñar los algoritmos y modelos de costo para la fragmentación vertical, horizontal e híbrida dinámica de bases de datos multimedia que consideren consultas basadas en contenido.
- Seleccionar las tecnologías que se utilizarán para la implementación de los algoritmos.
- Implementar los algoritmos y modelos de costo en los SGBDs utilizando bases de datos ampliamente empleadas en la literatura.
- Validación de los algoritmos y modelos de costo desarrollados por medio de casos de estudio.
- Comparar los algoritmos y modelos de costo con los métodos y modelos seleccionados del estado del arte.

CONTRIBUCIÓN AL CONOCIMIENTO

Un nuevo método de cada tipo de fragmentación (horizontal, vertical e híbrida) para bases de datos multimedia que adaptan los esquemas de fragmentación de acuerdo a los cambios en los patrones de acceso a la base de datos para reducir el tiempo de respuesta y el costo de ejecución de las consultas basadas en contenido.

CAPÍTULO 1. ANTECEDENTES

1.1. Marco teórico

A continuación, se definen algunos términos relevantes para el trabajo de investigación.

1.1.1. Fragmentación de datos

La fragmentación de datos surge como una estrategia para proporcionar una unidad de distribución nueva y adecuada en términos de desempeño. Los fragmentos en un DDBMS (*Distributed Database Management System*, Sistema de Administración de Bases de Datos Distribuidas) permiten que los datos se almacenen cerca de sus puntos de uso y esto proporciona dos beneficios potenciales (Özsu & Valduriez, 2020):

- Cada sitio maneja solo una parte de la base de datos. Debido a esto, el desempeño de las solicitudes de entrada/salida y el trabajo de la CPU mejora frente a las bases de datos centralizadas.
- La ubicación de los datos reduce las demoras de los accesos remotos.

Las tablas se dividen de forma horizontal, vertical o híbrida. La fragmentación horizontal utiliza el operador de selección donde los predicados determinan la fragmentación, la fragmentación vertical se realiza por medio del operador de proyección. En algunos casos, una fragmentación horizontal o vertical aplicada a un esquema de bases de datos no es suficiente para satisfacer los requisitos de las aplicaciones de los usuarios. En este caso, una fragmentación vertical se combina con una horizontal, o viceversa, produciendo una fragmentación híbrida (Özsu & Valduriez, 2020).

La fragmentación se realiza de forma estática o dinámica (Rodríguez et al, 2013). Rodríguez et al. (2013) muestran diferentes desventajas del enfoque estático y mencionan el enfoque dinámico como una solución para todos ellos.

Se aplican las siguientes tres reglas para garantizan que la base de datos no sufra cambios semánticos durante la fragmentación:

1. Completitud: Si una relación R se descompone en fragmentos $F_R = \{R_1, R_2, \ldots, R_n\}$, cada elemento de datos que se encuentre en R también se encuentra en uno o más de

R_i. En el caso de la fragmentación horizontal, el "elemento" se refiere a una tupla, mientras que en el caso de la fragmentación vertical, se refiere a un atributo.

2. Reconstrucción: Si una relación R se descompone en fragmentos $F_R = \{R_1, R_2, \ldots, R_n\}$, debería ser posible la reconstrucción de la relación a partir de sus fragmentos. Esto garantiza que se preserven las restricciones definidas en los datos en forma de dependencias.
3. Disyunción: Si una relación R se descompone en fragmentos $F_R = \{R_1, R_2, \ldots, R_n\}$ y el elemento d_i está en R_j, no debe estar en ningún otro fragmento. Este criterio asegura que los fragmentos sean disjuntos. Si la relación R se descompone verticalmente, sus atributos de clave primaria normalmente se repiten en todos sus fragmentos (para la reconstrucción). Por lo tanto, en el caso de fragmentación vertical, la disyunción se define solo en los atributos que no forman parte de la clave primaria de una relación.

1.1.2. Recuperación de imágenes basada en contenido

CBIR es una técnica que permite al usuario extraer una imagen con base en una consulta, desde una base de datos con una gran cantidad de imágenes. Usualmente el procedimiento se lleva a cabo realizando una comparación de diferentes características de bajo nivel, tales como: color, textura y forma, extraídas desde las imágenes mismas (Jenni, Mandala & Sunar, 2015).

1.1.3. Descriptor visual

Se define como la representación matemática del contenido de la imagen. Así, el descriptor visual más básico es un histograma de color, donde un conjunto limitado de valores representa de manera estadística el número de píxeles contabilizados de cada color en una imagen. Además de éste, existen múltiples descriptores que tratan de representar una información de mayor nivel semántico como los bordes presentes o las formas básicas que aparecen en la imagen (Rodríguez-Vaamonde, Torre-Bastida & Garrote, 2014).

1.1.4. Consultas kNN

En un conjunto de datos P y un punto de consulta q, una consulta kNN (*k-Nearest Neighbors*, k vecinos más cercanos) obtiene los puntos de datos k más cercanos a q (Papadias, 2009).

1.1.5. SURF

SURF (*Speeded-Up Robust Features*, Características Robustas Aceleradas) es un algoritmo para obtener conjuntos detector-descriptor. El detector se basa en una matriz *hessiana* e imágenes integrales para reducir el tiempo de cálculo. El descriptor, por otro lado, describe una distribución de respuestas *Haar-wavelet* dentro de los puntos de interés colindantes.

Para la extracción del descriptor, el primer paso consiste en construir una región cuadrada centrada alrededor del punto de interés y orientada a lo largo de la orientación seleccionada con base en información de una región circular alrededor del punto de interés. La región se divide en subregiones más pequeñas para calcular características simples en puntos de muestra. Para aumentar la robustez frente a deformaciones geométricas y errores de localización, los cálculos se ponderan con una función gaussiana centrada en el punto de interés. Luego, las respuestas *Haar-wavelet* se resumen en cada subregión y forman un primer conjunto de entradas al vector de características. Para obtener información sobre la polaridad de los cambios de intensidad se suman los valores absolutos de las respuestas. De esta manera, cada subregión posee un vector de cuatro dimensiones para su descriptor de estructura de intensidad subyacente. Esto da como resultado un vector descriptor. Las respuestas de las ondas son invariantes a un sesgo en la iluminación. La invariancia al contraste se logra convirtiendo el descriptor en un vector unitario (Bay, Tuytelaars & Van Gool, 2006).

1.1.6. BoofCV

BoofCV es una biblioteca de código abierto dedicada al procesamiento de imágenes de bajo nivel, calibración de camaras, detección / seguimiento de características, estructura a partir del movimiento y reconocimiento. BoofCV se encuentra bajo una licencia Apache 2.0 para uso académico y comercial.

BoofCV está organizado en varios paquetes: procesamiento de imágenes, características, visión geométrica, calibración, reconocimiento, visualización e IO (*Input/Output*, Entrada/Salida). El procesamiento de imágenes contiene funciones de procesamiento de imágenes de uso común que operan directamente en píxeles. El paquete de características contiene algoritmos de extracción de características para usar en operaciones de nivel

superior (Abeles, 2021).

1.1.7. UWE

UWE 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*, Lenguaje de Modelado Unificado) para sus modelos y sus métodos. En su implementación se deben contemplar 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.

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 (Guerrero, Domínguez & Pech, 2014).

1.1.8. Bases de datos multimedia

Actualmente, los datos multimedia se utilizan ampliamente, como audio, imagen y video. Se incluyen en bibliotecas digitales, catálogos electrónicos, entre otros. El rápido desarrollo de las aplicaciones multimedia ha creado un enorme volumen de datos multimedia que crece exponencialmente. Una base de datos multimedia es crucial en estas aplicaciones para proporcionar una recuperación de datos eficiente, ya que la principal característica de estos es su gran tamaño. Los administradores de bases de datos enfrentan un desafío de gran importancia, ya que deben contemplar las diferentes características de este tipo de bases de datos. Al igual que las bases de datos tradicionales, estas bases de datos sufren cambios en los

patrones de acceso y una estrategia que solucione la adaptación de los esquemas a estos cambios es de gran utilidad (Castro-Medina et al., 2020).

CAPÍTULO 2. ESTADO DEL ARTE

Los temas de fragmentación estática y dinámica son ampliamente abordados por trabajos en los que se busca una mejora de los sistemas distribuidos. Para dar un enfoque que contribuya a estos temas 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 estática y dinámica en bases de datos multimedia, posteriormente se hace una síntesis por medio de una tabla y por último se concluye con los hallazgos obtenidos después de haber analizado este número considerable de trabajos.

Una forma de entender los diferentes enfoques de los trabajos obtenidos es clasificarlos utilizando las categorías que se muestran en la Figura 2.1.

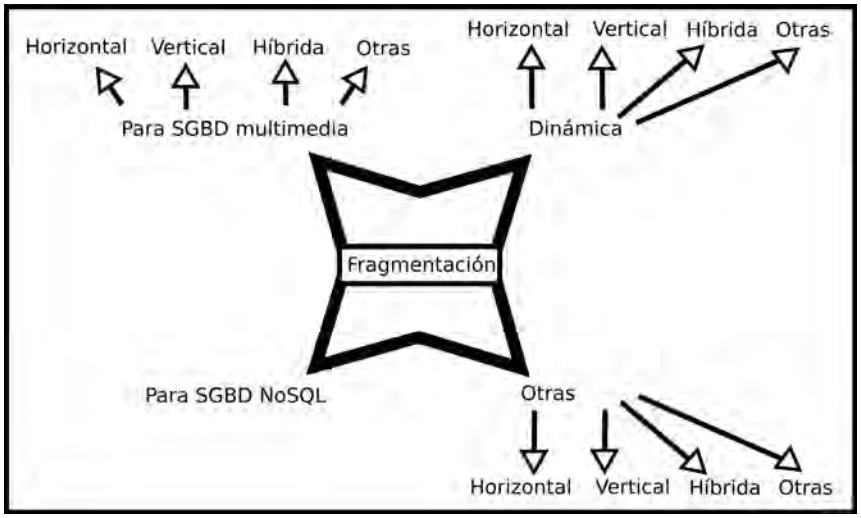


Figura 2.1. Clasificación de los artículos obtenidos en la etapa de búsqueda

Se proponen cuatro categorías para presentar los trabajos obtenidos. La primera categoría contiene los métodos que se centran en bases de datos multimedia; la segunda categoría cubre todos los enfoques que realizan una fragmentación dinámica y que no se enfocan en datos multimedia; la tercera categoría contiene todos los trabajos que consideran bases de datos NoSQL (No solo SQL), que no realizan fragmentación dinámica y que no toman en cuenta elementos multimedia; la última categoría muestra los trabajos que no están incluidos en ninguna de las categorías anteriores. Cada categoría se subdivide por tipo de fragmentación, es decir, horizontal, vertical e híbrida. La tercera categoría no se subdivide debido a la naturaleza de las bases de datos NoSQL.

La tecnología NoSQL tiene alta disponibilidad y alta escalabilidad, lo que proporciona nuevos métodos para el almacenamiento y la gestión de datos no estructurados. La tecnología NoSQL abandona el modelo relacional de su paradigma, puede almacenar datos con diferentes tipos y tiene características de alta escalabilidad (Wu, Chen & Jiang, 2016). Estas características hacen que las bases de datos NoSQL sean atractivas para almacenar y administrar datos multimedia.

2.1. Fragmentación de bases de datos multimedia

Actualmente, las aplicaciones multimedia se usan en gran medida, como audio/video, bibliotecas digitales, catálogos electrónicos, entre otros. El desarrollo acelerado de aplicaciones multimedia ha creado un gran volumen de datos multimedia y su uso se incrementa exponencialmente. Una base de datos multimedia es crucial en estas aplicaciones para proporcionar una recuperación de datos eficiente (Rodríguez-Mazahua et al., 2014) . En esta sección se agrupan los trabajos que abordan la fragmentación e incluyen bases de datos multimedia.

2.1.1. Fragmentación horizontal de bases de datos multimedia

El trabajo descrito por Saad et al. (2006) se enfocó en presentar la fragmentación horizontal primaria multimedia y propuso una estrategia de fragmentación basada en características multimedia de bajo nivel. Saad et al. enfatizaron la importancia de las implicaciones de los predicados multimedia en la optimización de fragmentos multimedia. Para validar el enfoque utilizado se presentó la implementación de un prototipo computacional de implicaciones de predicados multimedia. Se identificó la necesidad de clasificar los objetos multimedia que corresponden al mismo tipo para lograr un criterio consistente de fragmentación horizontal. Se mencionó que la utilización de nuevos operadores multimedia permitió que los procedimientos existentes de fragmentación fueran capaces de fragmentar datos multimedia. Esta investigación se enfocó en la fragmentación horizontal primaria de datos multimedia no estructurados enfatizando el impacto de las implicaciones de los predicados multimedia en la optimización de los fragmentos multimedia. Como trabajo futuro se incorporará la generación de un esquema conceptual multimedia, incluyendo el proceso de la fragmentación horizontal derivada y la optimización de los métodos de fragmentación usados.

La calidad del diseño de la distribución de bases de datos, la cual involucra las técnicas de fragmentación y asignación, debería ser juzgada por el desempeño de un sistema. Ma, Schewe & Wang (2006) analizaron la fragmentación y la asignación en el contexto de bases de datos con valores complejos. La fragmentación y la asignación de fragmentos a sitios se realizan de manera simultánea. Además, se presenta un modelo de costos del procesamiento de consultas para evaluar el desempeño del sistema. El principal aporte de los autores fue un enfoque heurístico para la fragmentación y la asignación. La evaluación se llevó a cabo mediante un esquema de bases de datos el cual fue poblado, posteriormente se consideraron cuatro sitios y 30 consultas, de las cuales 20% de ellas fueron frecuentemente usadas por las transacciones más importantes. El resultado del experimento validó el enfoque propuesto, ya que se minimizó el costo de transporte y de esta manera el desempeño de los esquemas de bases de datos mejoró.

Getahun et al. (2007a) trataron la fragmentación de bases de datos multimedia para tomar en cuenta las múltiples características de los objetos multimedia. Se discutió principalmente la fragmentación horizontal primaria multimedia y la investigación se centró en la implicación de los predicados textuales basada en semántica requerida como un preprocesamiento en los algoritmos actuales para fragmentar eficientemente datos multimedia. El núcleo de la propuesta se constituye por el uso del concepto del vecino más cercano en las bases de conocimiento para identificar implicaciones semánticas. Los autores mostraron el desarrollo y la implementación de un prototipo para evaluar el desempeño del enfoque presentado. Como trabajo futuro se mencionó que se evaluará a fondo la eficiencia del enfoque propuesto mediante un estudio comparativo para mostrar el progreso en la calidad de la fragmentación.

Getahun et al. (2007b) abordaron la fragmentación horizontal primaria de datos multimedia textualmente anotados. En este trabajo se discutió el problema de la identificación de las implicaciones semánticas entre predicados multimedia basados textualmente y se propuso integrar bases de conocimiento como marco de trabajo para evaluar la semántica de afinidad entre valores de predicados y operadores. En esta investigación también se presentó un prototipo que implementa varios aspectos de las implicaciones de predicados multimedia, mostrando los resultados experimentales, los cuales indican que el método propuesto es relevante. Se enfatizó la implicación de predicados basados en semántica, la cual se requiere

en los algoritmos de fragmentación actuales para la eficiente partición de datos multimedia. Se propuso un conjunto de algoritmos para identificar las implicaciones entre predicados semánticos, basados en implicaciones de operador y valor. Las implicaciones de operador se identifican utilizando un operador específico basado en conocimiento desarrollado en esta misma investigación. Además, se presentó un prototipo para probar el enfoque utilizado el cual demostró que el método propuesto tiene una complejidad polinomial. Como trabajo futuro se mencionó que incluirán la fragmentación horizontal derivada y la fragmentación vertical de datos multimedia tomando en cuenta la semántica y características multimedia de bajo nivel.

Grandes cantidades de datos complejos como imagen, sonido y video se recolectan y organizan en las bases de datos cada día. Estos datos complejos se presentan en diferentes formatos y no pueden ordenarse por sus contenidos de la misma manera que los datos convencionales. Fasolin et al. (2013) propusieron un enfoque eficiente para ejecutar consultas conjuntivas en grandes datos complejos junto con datos convencionales. Se realizó una fragmentación horizontal de acuerdo al criterio frecuentemente usado en los predicados de las consultas. La colección de fragmentos se indexó para encontrar eficientemente los fragmentos cuyo contenido satisface algunos predicados de consulta. Esta estrategia se aplicó a una colección de más de 106 millones de imágenes junto con sus datos convencionales relacionados. Este enfoque tuvo tres pasos: 1) Encontrar fragmentos con datos satisfaciendo algunos predicados de consultas; 2) Filtrar los datos en los fragmentos elegidos de acuerdo a otros predicados conectados de forma conjunta con los primeros; 3) Componer los resultados obtenidos de cada fragmento. Los resultados experimentales mostraron un aumento considerable en el rendimiento con el enfoque propuesto para consultas con predicados convencionales y basados en similitudes, en comparación con el uso de un índice métrico único para todo el contenido de la base de datos.

Rodríguez-Mazahua et al. (2014) propusieron un método de fragmentación horizontal para bases de datos multimedia el cual está basado en el algoritmo de agrupación jerárquica aglomerativa. La principal ventaja de este método es que no usa afinidad para crear el esquema de fragmentación horizontal. 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 de fragmentación horizontal multimedia 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 futuro, se mencionó que se desarrollará un algoritmo de fragmentación híbrida para bases de datos multimedia tomando en cuenta consultas basadas en contenido.

La cultura Hakka es una parte importante de la cultura del sur de China. Hay muchas características y un gran volumen de datos de esta cultura recopilados a través de la tecnología digital como datos no estructurados, lo que conlleva una gran dificultad para administrarlos y usarlos. La tecnología NoSQL tiene alta disponibilidad y alta escalabilidad, esto proporciona nuevos métodos para el almacenamiento y la gestión de datos de la cultura Hakka. Wu, Chen & Jiang (2016) gestionaron datos no estructurados de la cultura Hakka mediante MongoDB, una base de datos NoSQL orientada a documentos. El estudio se centró en la estrategia de gestión de datos heterogéneos de las múltiples fuentes de información, además, se exploró el mecanismo de fragmentación, indexación y consulta para la base de datos. Se tomaron como ejemplo los datos de la cultura Hakka de Fujian occidental y se construyó el prototipo del sistema de gestión de datos. La fragmentación y replicación se llevan a cabo en una arquitectura de bases de datos distribuidas. Los datos de la cultura Hakka se administran por tres tipos de servidores: servidor de datos, servidor de configuración y servidor de ruta. Los fragmentos se localizan en el servidor de datos y las consultas se realizan mediante llaves de fragmento. Los autores concluyeron mencionando que se presentó un nuevo método para mejorar la flexibilidad y la eficiencia de la gestión de datos no estructurados y heterogéneos de múltiples fuentes.

2.1.2. Fragmentación vertical para bases de datos multimedia

Fung, Leung & Li (2003) describieron el desarrollo de un sistema de base de datos de video *elearning*. La base de datos de video *elearning* proporciona un marco de trabajo para la modelación temporal para describir datos de video *elearning* y soporta la distribución de datos aplicando técnicas de fragmentación vertical de clases. En el trabajo previo se

construyó un sistema de información de cuatro dimensiones (4DIS, *Four Dimensional Information System*), el cual es un marco de trabajo para la modelación temporal orientada a objetos. Se aplicaron las técnicas de fragmentación vertical de clases sobre 4DIS como una medida para la ejecución eficiente de consultas. En este trabajo se observó la descripción del sistema administrador de base de datos de video *elearning* basado en el marco de trabajo para la modelación temporal y la aplicación de técnicas de fragmentación vertical de clases. Se presentó la obtención dinámica de video multimedia de *elearning* en internet y se desarrolló un modelo de costos detallado para la ejecución de consultas a través de la fragmentación vertical de clases. La efectividad de este enfoque se demostró en el contexto del sistema de base de datos de video *elearning* que usa el marco de modelado 4DIS. Como trabajo a futuro se mencionó que se harán estudios empíricos con consultas de *benchmark* sobre el sistema prototipo *elearning* experimental.

Rodríguez & Li (2011a) presentaron un algoritmo de fragmentación vertical para bases de datos multimedia distribuidas (MAVP, *Multimedia Adaptable Vertical Partitioning*) que toma en cuenta el tamaño de los objetos multimedia para generar un esquema de fragmentación vertical óptimo. MAVP minimiza la cantidad de accesos a datos irrelevantes y el costo de transporte de las consultas en bases de datos multimedia distribuidas para lograr una recuperación eficiente de objetos multimedia. MAVP requiere como entrada la matriz de uso de atributos y el tamaño de los mismos. El algoritmo soporta la fragmentación vertical de *n* maneras y la fragmentación vertical con el mejor ajuste. La primera genera el número específico de fragmentos requeridos por el usuario y la segunda genera una partición óptima general que minimiza el costo de procesamiento de consultas sin restricción en el número de fragmentos generados. En este trabajo se presentó el desarrollo de un modelo de costos, el cual considera el costo de procesamiento general de consultas en un ambiente distribuido multimedia. Como trabajo a futuro se mencionó la fragmentación vertical dinámica en bases de datos distribuidas multimedia basada en los cambios de las consultas.

La gran mayoría de algoritmos de fragmentación vertical son estáticos, es decir, que optimizan esquemas de fragmentación vertical mediante una carga de trabajo, pero si esta sufre cambios, también los sufrirá el esquema, lo que resultará en tiempos largos de respuesta de consultas. Rodríguez-Mazahua et al. (2017) propusieron un conjunto de reglas activas para

realizar la fragmentación vertical dinámica sobre bases de datos multimedia. Las reglas activas se implementaron en DYMOND (*DYnamic Multimedia ONline Distribution*, Distribución dinámica multimedia en línea), el cual es un sistema basado en reglas activas para la fragmentación vertical dinámica de bases de datos multimedia. DYMOND monitorea las consultas realizadas para determinar cuándo es necesaria una nueva fragmentación. El desempeño de la base de datos se mide en términos de la cantidad de accesos locales irrelevantes y la cantidad de accesos remotos relevantes. Estas medidas se utilizan para determinar el desempeño límite de una base de datos multimedia y determinar cuándo desencadenar una nueva fragmentación. En este trabajo se describieron todas las reglas activas aplicadas en DYMOND y se presentó un caso de estudio de una compañía de ventas de maquinaria, la cual posee una base de datos multimedia, esta base de datos contiene una tabla llamada equipo y contiene 100 tuplas con atributos alfanuméricos y multimedia. Se realizó la fragmentación vertical aplicando las reglas activas sobre esta base de datos y se obtuvo un menor tiempo de respuesta cuando se realizó la primera fragmentación y también cuando se refragmentó.

2.1.3. Fragmentación híbrida para bases de datos multimedia

Chbeir y Laurent (2010) presentaron un enfoque formal para identificar implicaciones de predicados y consultas para fragmentar eficientemente datos multimedia. Los autores en este trabajo se enfocaron en la fragmentación de bases de datos bajo el objetivo de reducir el acceso a datos irrelevantes mediante la agrupación de datos frecuentemente utilizados. Este enfoque es capaz de considerar comparaciones multimedia al igual que semánticas, por medio de una noción generalizada de dependencias funcionales las cuales son llamadas dependencias funcionales multimedia. Se mencionó que el enfoque propuesto utiliza la fragmentación combinada de datos multimedia basada en la comparación de consultas y la equivalencia de consultas. Se enfatizó que la principal característica del enfoque es la forma en la que se comparan las consultas, ya que, por medio de esta, hace explícito el uso de la semántica de los datos a fragmentar. Las dependencias funcionales multimedia toman en cuenta características específicas de los datos multimedia y son axiomatizadas de la misma manera que las dependencias funcionales estándar. Como trabajo futuro se planeó implementar el método de fragmentación, estudiar específicamente la prueba de vinculación de predicados atómicos y calcular las respuestas a las consultas en las que los predicados de

selección atómica se distribuyen en varias consultas.

Debido al incremento de las aplicaciones multimedia, el uso de técnicas de fragmentación para reducir el número de páginas requeridas para responder una consulta es de gran utilidad. Rodríguez-Mazahua et al. (2016) presentaron un método de fragmentación híbrida para bases de datos multimedia que toma en cuenta el tamaño de los atributos y la selectividad de los predicados para crear esquemas de fragmentación híbridos. Se presentaron los dos tipos de fragmentación horizontal y se menciona que para este trabajo se utiliza la fragmentación horizontal primaria. Existen tres maneras para desempeñar la fragmentación híbrida, la primera consiste en desempeñar primero la fragmentación vertical y después la horizontal en cada fragmento vertical, la segunda consiste en realizar la fragmentación horizontal y aplicar posteriormente la vertical en cada fragmento horizontal, por último, la tercera manera consiste en realizar la fragmentación híbrida tomando en cuenta directamente la semántica de las operaciones. En este trabajo se desempeñó primero la fragmentación horizontal y posteriormente la vertical. Para llevar a cabo la fragmentación horizontal se determinó el conjunto de predicados usados en la tabla a fragmentar y se construyó la matriz de uso de predicados, la cual relaciona consultas y predicados, colocando uno dentro de la tabla si la consulta usa al predicado y cero cuando no lo ocupa. Posteriormente se crea el árbol de partición basado en el enfoque *bottom-up*, comenzando por un solo predicado y creando nuevos esquemas de fragmentación mezclando repetidamente pares de predicados. Para realizar la fragmentación vertical se construye una matriz de uso de atributos, la cual se considera como un conjunto de atributos multimedia y relaciona atributos con consultas. Dentro de esta tabla se coloca en cada celda un uno si la consulta utiliza al atributo con el cual está relacionado y un cero si no. El algoritmo propuesto busca el esquema de partición vertical más adecuado para posteriormente obtener el esquema híbrido tomando en cuenta el modelo de costos descrito en el mismo trabajo. Se mencionó como conclusión que el método de partición híbrida para bases de datos multimedia reduce el acceso a datos irrelevantes tomando en cuenta el tamaño y la selectividad de cada atributo, además, como conclusión también se mencionó que se propuso un modelo de costos para bases de datos distribuidas multimedia.

La partición de índices multimedia es clave para una búsqueda eficiente de kNN (*k-Nearest Neighbors*, k Vecinos más Cercanos), pero los algoritmos existentes se basan en la similitud de documentos, sin importar el tamaño de la partición o las restricciones de redundancia. Mourão & Magalhães (2017) crearon un algoritmo de partición de índice que abordó las propiedades específicas de un sistema distribuido: equilibrio de carga entre nodos, redundancia en la falla de los nodos y uso eficiente de los nodos bajo consultas concurrentes. Los autores propusieron la representación de datos con libros de códigos. Cada documento se cuantifica en un pequeño conjunto de claves y se indexa en particiones por clave. Se propuso el algoritmo B-KSVD (*Balanced* KSVD), el cual distribuye los datos de manera uniforme entre las claves mediante la distribución en el espacio original. Los experimentos se centraron en medir la efectividad del equilibrio del tamaño de partición y la calidad de recuperación. Los resultados mostraron que B-KSVD equilibra mejor los tamaños de partición, en comparación con *k-means* y KSVD. Como conclusión se mencionó que se formalizaron los requisitos para crear representaciones completas con indexación de documentos redundantes, donde las particiones contienen subconjuntos de datos traslapados. Se propusieron representaciones basadas en codificación dispersa y modelos de agrupamiento, y una adaptación al algoritmo KSVD, que distribuye valores *hash* entre las posiciones, de acuerdo con la distribución global.

Los proveedores de la nube se enfrentan cada vez más a requisitos muy diversos y heterogéneos que sus clientes imponen en la gestión de datos. El problema de los requisitos heterogéneos en la gestión de datos en la nube se aborda a nivel global replicando y dividiendo datos en centros de datos o a nivel local al proporcionar sistemas de *polystore* en un centro de datos en la nube, sin embargo, no existe una solución integrada que aproveche los beneficios de ambos enfoques. Vogt, Stiemer & Schuldt (2018) presentaron la visión de Polypheny-DB de un sistema de *polystore* distribuido que combina a la perfección la replicación y la partición con *polystore* locales y que es capaz de adaptar dinámicamente todas las partes del sistema cuando cambia la carga de trabajo. Se presentaron los componentes básicos para ambas partes del sistema y se mostraron los desafíos abiertos hacia la implementación de la visión Polypheny-DB.

Diferentes dominios se ocupan de la gestión de volúmenes de datos masivos y miles de transacciones OLTP (*OnLine Transaction Processing*, Procesamiento de Transacciones En Línea) por segundo. Las bases de datos relacionales tradicionales no pueden hacer frente a estos requisitos. NewSQL es una nueva generación de bases de datos que proporciona alta escalabilidad, disponibilidad y soporte de propiedades ACID (*Atomicity, Consistency, Isolation, Durability*; Atomicidad, Consistencia, Aislamiento, Durabilidad). Aunque la fragmentación de datos es una característica esencial para administrar bases de datos relacionales, sigue siendo un problema abierto para los sistemas NewSQL. Schreiner et al. (2019) propusieron un enfoque de fragmentación híbrida para bases de datos NewSQL que permite al usuario definir las particiones de datos verticales y horizontales. Para determinar qué sitio almacenará cada fragmento de datos, se realizó una función *hash* que considera la información del esquema y las estadísticas de acceso a datos. La evaluación experimental comparó la versión híbrida de VoltDB con VoltDB estándar. Los resultados destacan que la estrategia mostrada aumentó el número de transacciones en un solo sitio del 37% al 76% conservando el mismo tiempo de respuesta. Como trabajo a futuro mencionaron que incluirán la evaluación de este enfoque con otros puntos de referencia y la ejecución de pruebas en un *cluster* distribuido físicamente.

2.1.4. Otros tipos de fragmentación para bases de datos multimedia

Torjmen, Pinel-Sauvagnat & Boughanem (2008) se enfocaron en la recuperación de documentos XML (*Extensible Markup Language*, lenguaje de marcado extensible) multimedia, cuyo objetivo es encontrar los componentes de un documento relevante que atienda las necesidades del usuario. Se propuso para representar elementos multimedia usar no solo información textual, sino también una estructura jerárquica. Gracias al poder representar documentos XML de forma arbórea se utiliza una analogía entre estos tipos de documentos y ontologías. Adicionalmente, el modelo propuesto también define la mejor ventana de fragmentos multimedia para retornarse al usuario. Una medida extendida de Wu-Palmer se define para calcular el grado de participación de cada nodo no multimedia en la representación. Los resultados mostraron que la información textual de los elementos debe ser diferente, retornar fragmentos multimedia basados en las relaciones jerárquicas entre los elementos y los elementos multimedia permite obtener buenos resultados.

El objetivo de Torjmen-Khemakhem, Pinel-Sauvagnat & Boughanem (2013) fue estudiar el impacto, en términos de efectividad, de la posición del texto en relación con los objetos multimedia buscados. Los objetos multimedia que se consideraron se describen en documentos estructurados como XML. Por lo tanto, la estructura del documento se explota para proporcionar esta posición de texto en los documentos. Aunque se ha demostrado que la información estructural es una fuente efectiva de evidencia en la recuperación de información textual, solo unos pocos trabajos investigaron su interés en la recuperación multimedia. El enfoque general de los autores se basó en dos pasos: primero recuperar elementos XML que contienen objetos multimedia, y luego explorar la información circundante para recuperar fragmentos multimedia relevantes. Se llevaron a cabo diferentes experimentos en el contexto del marco de trabajo INEX (*Initiative for the Evaluation of XML Retrieval*, Iniciativa para la evaluación de la respuesta XML), el cual contiene más de 660,000 documentos XML. Los resultados mostraron que las evidencias estructurales son de gran interés para ajustar la importancia del contexto textual para la recuperación multimedia. Como conclusión se mencionó que el enfoque propuesto superó a los del estado del arte.

Mettes et al. (2015) propusieron la detección y recuento de eventos usando una representación de puntajes de detector de concepto. Se propuso codificar videos usando fragmentos que se aprenden de manera discriminatoria por evento. La "bolsa de fragmentos" divide un video en propuestas de fragmentos semánticamente coherentes. Los experimentos llevados a cabo mostraron que los fragmentos propuestos por este enfoque ofrecen un mejor desempeño en las llamadas a sub-eventos. La "bolsa de fragmentos" se utiliza como una medida complementaria al proceso principal y proporciona una mejor detección de eventos. Ademas, se observó que la "bolsa de fragmentos" con el filtrado de conceptos produce un recuento de eventos deseable. Los experimentos se llevaron a cabo en un conjunto de datos llamado THUMOS'14, el cual contenía 1010 videos de validación. La evaluación experimental mostró la efectividad de la codificación de la detección de eventos como también la complementación natural de la agregación global de conceptos semánticos.

La búsqueda de similitud en datos ricos en características a gran escala (por ejemplo, imagen, video o texto) es un problema fundamental y se ha vuelto cada vez más importante en la investigación de minería de datos. Lu et al. (2018) diseñaron una nueva estructura de índice

llamada Dynamic Partition Forest (DPF) para dividir jerárquicamente las áreas de alta colisión con *hashing* dinámico, esto provoca una autoadaptación de varias distribuciones de datos. Se integró una estrategia de búsqueda de múltiples pasos con DPF para mitigar la pérdida de precisión con un esquema distribuido. Los resultados del experimento mostraron que DPF aumenta la precisión en un 3% a 5% dentro del mismo período de tiempo en comparación con DPF sin búsqueda de pasos múltiples. Además, DPF con el esquema de partición propuesto es 1.4 veces más rápido que DPF sin partición. Las comparaciones experimentales con otros dos métodos de vanguardia en tres conjuntos de datos populares muestran que DPF es de 3.2 a 9 veces más rápido para lograr la misma precisión con una disminución del espacio de índice del 17% al 78%.

Santos & Masala (2018) investigaron el uso de la fragmentación utilizando un patrón para dividir los datos en fragmentos antes de almacenarlos en la nube. Se comparó el rendimiento en dos proveedores diferentes de la nube. Además, se propuso un enfoque novedoso que combina una técnica de fragmentación de patrones con una base de datos NoSQL para organizar y administrar los fragmentos. Se presentó un análisis del desempeño usando la fragmentación de datos propuesta en diferentes proveedores en la nube. Para realizar los experimentos se utilizaron tres tipos de archivos, docx, jpg y pdf, todos con 100 Kb de tamaño. El resultado mostró que la fragmentación con patrones aleatorios es más rápida que otros enfoques. Mencionaron que como trabajo a futuro usarán bases de datos orientadas a columnas y guardarán los archivos divididos en diferentes ambientes y en objetos con formatos binarios en lugar de usar bases de datos orientadas a documentos, las cuales guardan la información en formato JSON (*JavaScript Object Notation*, Notación de Objetos de JavaScript).

Un estándar internacional entre varios proveedores para transmitir, almacenar, recuperar, imprimir, procesar y mostrar información de imágenes médicas es el formato DICOM (*Digital Imaging and Communications in Medicine*, Imagen Digital y Comunicaciones en Medicina). La computación en la nube hace posible el manejo de grandes cantidades de datos médicos. Le, Kantere & Orazio (2019) afirmaron que el número de soluciones encontradas es grande, pero no proporcionan ningún método para buscar la configuración óptima de datos híbridos. Se propuso NSGA-G, un NSGA (*Non-dominated Sorting Genetic Algorithm*,

Algoritmo genético de clasificación no dominado) basado en la fragmentación de cuadrícula para mejorar la complejidad y la calidad de los actuales NSGAs y para obtener el almacenamiento eficiente de las consultas realizadas a datos híbridos DICOM. El algoritmo propuesto busca una solución pareto-óptima con una buena compensación entre la diversidad y el desempeño. Los experimentos sobre archivos DICOM en un almacenamiento híbrido probaron que NSGA-G provee el mejor tiempo de procesamiento con resultados interesantes.

La distribución de índices multimedia en múltiples nodos permite la búsqueda en conjuntos de datos muy grandes de imágenes y videos, pero presenta un conjunto de desafíos: la distribución de documentos y consultas de manera efectiva entre nodos para admitir consultas concurrentes y lidiar con el aumento potencial de la falta de respuesta de los nodos. Un índice donde las particiones se basan en la distribución de vectores de características en el espacio original es capaz de mejorar la redundancia y aumentar la eficiencia. Mourão & Magalhães (2019) describieron cómo los *hashes* dispersos ayudan a encontrar este equilibrio y crear mejores opciones de distribución para vectores de características de alta dimensión. La propuesta de los autores distribuye y equilibra documentos y consultas a un subconjunto de nodos, de acuerdo con sus similitudes ortogonales. Se realizaron diferentes pruebas exhaustivas del enfoque en un servicio comercial en la nube. Los experimentos en un conjunto de datos de mil millones de vectores mostraron que este enfoque tuvo una baja sobrecarga de fragmentación, logró una distribución equilibrada de documentos y consultas, manejó consultas concurrentes de manera efectiva y se tuvo poca degradación cuando los nodos fallan. Como conclusión se mencionó que se describe DISH (*Distributed Indexing by Sparse-Hashing*, Indexación distribuida por *hash* disperso), un algoritmo de fragmentación de índices de imágenes a gran escala mediante *hashing* disperso y se muestra el uso de *hashes* dispersos para fragmentar documentos y consultas de manera equilibrada y redundante entre nodos.

Los datos culturales y la información en la web aumentan, evolucionan y se modifican continuamente en forma de *Big Data* debido a la globalización, la digitalización y su vasta exploración. Debido a esto, su integración en forma de repositorios de *Big Data* es esencial. Sharma & Bawa (2019) analizaron la complejidad de los crecientes datos culturales y presentaron un repositorio de *Big Data* cultural como una forma eficiente de almacenar y

recuperar *Big Data* cultural. La fragmentación se realiza mediante GridFS, dividiendo archivos en documentos bajo el SGBD de MongoDB. Las consultas se administran por el *driver* de GridFS, el cual se encarga de reunir las partes de cada archivo para satisfacer las peticiones. GridFS se encarga de dividir los archivos mayores a 16 MB y la base de datos CBDR (*Cultural Big Data Repository*, Repositorio de *Big Data* Cultural) automáticamente distribuye en fragmentos los menores a esta cifra. El repositorio propuesto es altamente escalable y proporciona métodos integrados de alto rendimiento para el análisis de *Big Data* del patrimonio cultural. Los resultados mostraron que un entorno no fragmentado requiere menos tiempo para las operaciones de lectura y escritura, pero el tamaño de *ítem* no es óptimo si se considera el factor de espacio de los datos culturales. Se mencionó como conclusión que CBDR permite una gestión eficiente de la información cultural proporcionando una administración de datos con diferentes estructuras de información.

2.2. Fragmentación dinámica

La fragmentación dinámica mejora el rendimiento de las bases de datos, resolviendo diferentes problemas de un enfoque estático. En este apartado se muestran todos los trabajos que se centran en realizar una fragmentación dinámica sin utilizar un enfoque orientado a bases de datos multimedia.

2.2.1. Fragmentación horizontal dinámica

El tiempo que se consume durante la ejecución de consultas en un entorno paralelo y distribuido se ve muy afectado por la forma en que las tablas que comprenden una base de datos se han fragmentado. Los métodos clásicos de fragmentación en un sistema distribuido de bases de datos ayudan, en gran medida, a hacer que la recuperación de información sea más rápida y con menores esfuerzos de cálculo. Sin embargo, se presenta un problema al usar estos métodos en aplicaciones donde la administración distribuida no puede hacer inferencias que ayuden a saber en qué sitios se ubican los datos con las características que el usuario está buscando. Vazquez (2000) presentó el método de Fragmentación Virtual Dinámica, el cual funciona como una forma alternativa que permite disminuir el tiempo de respuesta consumido por las consultas utilizando tablas fragmentadas horizontalmente. El método propuesto se probó e implementó en un servidor paralelo de base de datos que se ejecuta en una supercomputadora con arquitectura *shared nothing*. En las pruebas se observó

que el tiempo consumido para la ejecución de las consultas se ve muy afectado por la forma en que se fragmentaron las tablas, sin embargo, con la implementación del método de Fragmentación Virtual Dinámica, la mejora del desempeño en la recuperación de información se logró en las consultas que no contenían en el criterio de selección algún atributo utilizado por la fragmentación horizontal de la tabla.

Pinto & Torres (2002) abordaron el problema de la reasignación dinámica de datos en una base de datos distribuida, fragmentada con patrones de acceso cambiantes. Tradicionalmente, la fragmentación en bases de datos distribuidas se determina mediante análisis y optimización fuera de operación, sin embargo, se encontraron algunas empresas que tienen usuarios que acceden a sus bases de datos bajo patrones de acceso cambiantes. La Comisión Federal de Electricidad (CFE) fue una de ellas, ya que requirió un enfoque para fragmentar dinámicamente sus bases de datos, es decir, un algoritmo que sea capaz de reasignar datos mientras la base de datos está en operación. Se presenta como conclusión el desarrollo obtenido, el cual fue el algoritmo descrito, el cual mueve información entre bases de datos y replica parte de los datos en bases de datos esclavas. Además, se mencionó que se obtuvo un algoritmo basado en dos técnicas: Búsqueda esclavo-maestro, para proveer rápido acceso a las consultas de los usuarios, y Replicación esclavo-maestro, para aumentar la disponibilidad. Se observó un excelente comportamiento en el desempeño de la base de datos bajo el enfoque propuesto.

En los sistemas de bases de datos distribuidas, las tablas se fragmentan y replican con frecuencia en varios sitios para reducir los costos de comunicación de la red. La fragmentación, asignación y replicación son problemas desafiantes que se han resuelto previamente mediante enfoques estáticos o basados en un análisis de consultas *a priori*. Hauglid, Ryeng & Nørvåg (2010) desarrollaron DYFRAM, un enfoque descentralizado para la fragmentación y asignación dinámica de tablas en sistemas de bases de datos distribuidas basado en la observación de los patrones de acceso de los sitios a las tablas. DYFRAM realiza la fragmentación, replicación y reasignación en función del historial de acceso reciente, con el objetivo de maximizar el número de accesos locales y minimizar el acceso a sitios remotos. Se mostró la evaluación del enfoque propuesto, la cual se dividió en tres partes. La primera consistió en examinar los resultados desde la ejecución de un simulador en

cuatro cargas de trabajo involucrando dos sitios. En la segunda parte de la evaluación se utilizó el mismo simulador con dos cargas de trabajo dinámicas involucrando más sitios. La tercera parte de la evaluación consistió en implementar los experimentos sobre un sistema de base de datos distribuida. Los resultados de las simulaciones probaron que, para las cargas de trabajo típicas, DYFRAM reduce significativamente los costos de comunicación. El enfoque también demostró bien su capacidad para adaptarse a los cambios en la carga de trabajo. Además de las simulaciones, también se implementó el desarrollo de los autores en el sistema de base de datos distribuido DASCOSA-DB, y se demostró su desempeño en aplicaciones reales.

El procesamiento distribuido es una forma efectiva de mejorar el desempeño de los sistemas de bases de datos. Por esta razón, la fragmentación y la asignación adecuada de fragmentos en diferentes sitios de la red se consideran un área de investigación clave en el entorno de bases de datos distribuidas. Abdalla & Amer (2012) propusieron un modelo sincronizado de fragmentación horizontal, replicación y asignación en el contexto de las bases de datos relacionales. Se desarrolló una técnica heurística para satisfacer la fragmentación horizontal y la asignación utilizando un modelo de costos. Los experimentos se realizaron mediante una tabla, la cual contenía información sobre diferentes empleados de una empresa. Se consideraron cuatro sitios sobre un sistema distribuido, los costos entre sitios y las restricciones de cada uno. Se mencionó que usando este enfoque de fragmentación no se agregó ningún tipo de complejidad en la asignación de fragmentos. Este trabajo mejoró significativamente el desempeño de los sistemas de bases de datos distribuidas reduciendo el acceso remoto y los altos costos de trasferencia de datos entre los sitios.

Liroz-Gistau et al. (2012) y Liroz-Gistau et al. (2013) propusieron DynPart y DynPartGroup, dos algoritmos de fragmentación dinámica para bases de datos en continuo crecimiento. Estos algoritmos adaptan eficientemente la partición de datos a la llegada de nuevos elementos de datos teniendo en cuenta la afinidad de los nuevos datos con consultas y fragmentos. A diferencia de los enfoques estáticos existentes, el enfoque mostrado ofrece un tiempo de ejecución constante sin importar el tamaño de la base de datos y al mismo tiempo obtiene una muy buena eficiencia de fragmentación. Se validó la solución a través de la experimentación sobre datos reales. El conjunto de datos se tomó de Sloan Digital Sky

Survey catalog, Data Release 8 (DR8). Los resultados mostraron que en el caso de conjuntos de datos en los que existe una alta correlación entre los nuevos elementos de datos, el algoritmo DynPartGroup mantiene un comportamiento muy bueno. También revelaron que este algoritmo no se ve muy afectado por el desequilibrio de los tamaños de los fragmentos. Como conclusión se mencionó que se propusieron dos algoritmos para lidiar con las estrictas restricciones del desequilibrio que se presenta en bases de datos grandes y crecientes.

Al observar el problema de fragmentar de manera efectiva y eficiente los almacenes de datos, la mayoría de los enfoques de vanguardia, que a menudo son de base heurística, son estáticos, ya que suponen la existencia de un conjunto conocido de consultas *a priori*. Contrariamente a esto, en aplicaciones de la vida real, las consultas cambian dinámicamente y la heurística de fragmentación necesita integrar estos cambios. Siguiendo esta consideración principal, Bellatreche et al. (2013) propusieron y evaluaron experimentalmente un enfoque incremental para seleccionar esquemas de fragmentación de almacenamiento de datos utilizando algoritmos genéticos. El enfoque propuesto se evaluó utilizando el *benchmark* APB1 y un almacén de datos con un esquema en estrella, el cual se pobló con más de 24 millones de tuplas y tablas de 4 dimensiones. Las pruebas se realizaron en una escala pequeña y a gran escala, las cuales demostraron que de los tres algoritmos comparados ISGA (*Incremental Approach Based on Genetic Algorithms*, Enfoque Incremental Basado en Algoritmos Genéticos) superó a los otros dos. Como conclusión se mencionó que ISGA presentó una mejor optimización en el desempeño de las consultas mientras reduce los costos de mantenimiento.

El gran tamaño de un almacén de datos y la complejidad de las consultas OLAP (*OnLine Analytical Processing, P*rocesamiento Analítico en Línea) constituyen desafíos en el rendimiento de las consultas. La fragmentación de datos mejora significativamente la gestión de datos, la accesibilidad y el tiempo de ejecución de consultas. Un esquema de fragmentación óptimo está diseñado con base en la información recopilada de la carga de trabajo. Entonces, en el contexto de bases de datos relacionales y orientadas a objetos, estas técnicas permanecen adaptadas porque la carga de trabajo es muy estable. Sin embargo, las características específicas de los almacenes de datos y la naturaleza de las consultas OLAP hacen que el modelo de datos y la carga de trabajo sean muy dinámicos y, en consecuencia,

ciertos esquemas de fragmentación diseñados son ineficaces. Para resolver este problema, Derrar, Nacer & Boussaid (2013) propusieron un enfoque basado en el uso estadístico del acceso a datos para la fragmentación dinámica en los almacenes de datos. Se realizaron estudios experimentales utilizando Oracle 10G con el *benchmark* APB1 para verificar la adaptabilidad del enfoque propuesto. Se llevaron a cabo varias pruebas al considerar el criterio de evaluación como un umbral para el tiempo de respuesta de una consulta OLAP. Entonces, cuando la consulta se ejecuta más allá de este tiempo, se realizó un proceso de refragmentación de datos. Los resultados obtenidos fueron alentadores tanto en términos de espacio de memoria como de tiempo de ejecución de consultas.

Los datos estructurados de forma irregular se caracterizan por una variedad de entidades de rápida evolución sin un conjunto común de atributos. Estas entidades no muestran suficiente regularidad para capturarse en un esquema de base de datos tradicional. Una posible solución es usar una partición de tablas, que permita eliminar las particiones de entidades irrelevantes antes de tocarlas. Crear y mantener una partición de este tipo de forma manual es muy difícil incluso inviable, debido a la enorme complejidad. Herrmann, Voigt & Lehner (2014) resolvieron el problema de la fragmentación en línea para datos estructurados de manera irregular y presentaron a Cinderella, un algoritmo autónomo en línea para la fragmentación horizontal de entidades estructuradas irregularmente en tablas universales. Cinderella está diseñado para mantener su sobrecarga baja mediante la asignación gradual de entidades a particiones. Las particiones obtenidas permiten que solo se obtengan entidades con un subconjunto de atributos que eliminen fácilmente particiones de entidades irrelevantes. Cinderella aumenta la ejecución local y reduce el costo de ejecución de las consultas.

El rendimiento de cualquier base de datos distribuida depende en gran medida de la fragmentación de las relaciones globales y la asignación de esos fragmentos en diferentes sitios de la red. La asignación de datos o fragmentos se realiza de forma estática o dinámica. Al asignar eficientemente fragmentos de forma dinámica se mejora el rendimiento de las organizaciones comerciales distribuidas. Kumar & Gupta (2014) propusieron un enfoque extendido para la asignación dinámica de fragmentos en sistemas de bases de datos distribuidas no replicadas que incluye variables como el umbral de acceso, restricciones de tiempo de acceso, volumen de transmisión de datos y factor de distancia. Los cálculos

realizados sobre la base de datos hipotetica respaldaron que el algoritmo extendido propuesto, llamado TTVDCA (*Threshold, Time, Volume, and Distance Constraint Algorithm,* Algoritmo de Umbral, Tiempo, Volumen y Restricción de Distancia), es mejor que otros algoritmos desarrollados previamente en esta categoría y mostró una mejora en el rendimiento general del sistema.

Para las organizaciones que se expanden globalmente, las aplicaciones generan flujos de trabajo dinámicos con cambios frecuentes en los modelos de acceso a las bases de datos en diferentes sitios. En esas situaciones es adecuado un proceso dinámico para resolver las solicitudes en el sitio donde se generaron. La innovación presentada por Baron & Iacob (2014) consistió en la posibilidad de integrar los tres conceptos fundamentales específicos de las bases de datos distribuidas: fragmentación, replicación y asignación de fragmentos en un sistema dinámico desequilibrado, completamente descentralizado y totalmente automatizado que permitió lecturas y escrituras remotas a la réplica maestra y ofreció alta disponibilidad y un aumento en el rendimiento general del sistema. Como conclusión, los autores mencionaron que no solo innovaron, sino que también mejoraron el desempeño con el enfoque propuesto el cual es configurable y fácil de administrar. El modelo también es aplicable a bases de datos paralelas. Como trabajo a futuro se plantea desarrollar un sistema para detectar, basado en el análisis de consultas, el modelo más recurrente para aplicarse posteriormente.

Los entornos en la nube generalmente cuentan con varios centros de datos distribuidos geográficamente. Para aumentar la escalabilidad de las aplicaciones, muchos proveedores de la nube dividen los datos y distribuyen estas particiones en los centros de datos para equilibrar la carga. Sin embargo, si las particiones no se eligen cuidadosamente, puede dar lugar a transacciones distribuidas. Fetai, Murezzan & Schuldt (2015) presentaron Cumulus, un enfoque de fragmentación de datos adaptativo que identifica patrones de acceso característicos de mezclas de transacciones, determina particiones de datos basadas en estos patrones y vuelve a fragmentar datos dinámicamente si los patrones de acceso cambian. Se realizó la evaluación del enfoque con el *benchmark* TPC-C y se mostró que Cumulus aumentó significativamente el rendimiento general del sistema en una configuración OLTP en comparación con los enfoques de partición de datos estáticos. Además, se mostró que

Cumulus logra adaptarse a los cambios de carga de trabajo en tiempo de ejecución al generar particiones que coinciden con la carga de trabajo real y reconfigurar el sistema.

Abdel Raouf, Badr & Tolba (2016) desarrollaron un sistema dinámico mejorado de bases de datos distribuidas sobre un ambiente en la nube, el cual permite tomar dinámicamente decisiones de fragmentación, asignación y replicación en tiempo de ejecución. Además, se presentó una técnica mejorada de asignación y replicación que se aplica en una fase inicial del diseño de bases de datos distribuidas cuando no se tiene información de la ejecución de consultas. Mediante la información de las operaciones realizadas a una base de datos se crea una matriz llamada MCRUD (*Modified Create, Read, Update and Delete Matrix*, Matriz Modificada de Creación, Lectura, Actualización y Eliminación) que relaciona los tipos de operaciones que se realizaron con sus predicados y los sitios en los que fueron ejecutadas. Por medio de la MCRUD se obtiene la tabla ALP (*Attribute Locality Precedence*, Precedencia de la Localidad del Atributo), la cual muestra los costos de uso de cada atributo. De esta manera se crea un esquema de fragmentación teniendo en cuenta el atributo más costoso.

Un factor clave que afecta el rendimiento de los SGBD distribuidos es cómo se divide la base de datos. Si la base de datos está particionada incorrectamente, el número de transacciones distribuidas puede ser alto. Estas transacciones tienen que sincronizar sus operaciones a través de la red, que es considerablemente más lenta y conduce a un bajo rendimiento. Serafini et al. (2016) presentaron un nuevo enfoque de fragmentación en línea, llamado Clay, que admitió tanto esquemas basados en árboles como esquemas generales más complejos con relaciones arbitrarias de claves foráneas. Clay creó dinámicamente bloques de tuplas para migrar datos entre servidores durante el reparticionamiento sin imponer restricciones en el esquema, prestando especial atención al equilibrio de la carga y reducción de la cantidad de datos migrados. Clay logró este objetivo al incluir en cada bloque un conjunto de tuplas calientes (accedidas frecuentemente) y otras tuplas co-accesadas con estas tuplas calientes. Para evaluar el enfoque propuesto, se integró Clay en un SGBD distribuido de memoria principal y se demostró que generó esquemas de partición que permiten que el sistema logre un rendimiento hasta 15 veces mejor y una latencia 99% menor que los enfoques existentes.

El rendimiento de los sistemas de bases de datos distribuidas aumenta mediante una mejor

forma de fragmentación, asignación y replicación. La técnica de asignación dinámica de fragmentos proporciona una solución adecuada para los cambios de patrones de acceso a sitios desde múltiples ubicaciones a lo largo del tiempo. Zar Lwin & Naing (2018) propusieron un enfoque para la asignación dinámica de fragmentos no redundantes en un sistema de bases de datos distribuidas. El enfoque propuesto reasigna fragmentos mediante los patrones de acceso hechos a cada fragmento con la cantidad de datos, hasta la restricción de tiempo y el valor umbral deseados. La implementación se realizó sobre cuatro sitios en la nube totalmente conectados, en los cuales se ejecutaron 10,000 consultas. Se utilizó el enfoque propuesto para fragmentar la base de datos y migrar cada fragmento al mejor sitio. El resultado del experimento mostró que el rendimiento de los cuatro sitios después de la migración es ligeramente mejor que el rendimiento de los cuatro sitios antes de la migración.

El tamaño creciente de los archivos de datos sin procesar hace que la carga de datos sea una operación costosa que retrasa el tiempo de respuesta de la información. Para aliviar el costo de carga, los sistemas de procesamiento de consultas operan directamente sobre los datos sin procesar y ofrecen acceso instantáneo a los datos. Olma et al. (2019) presentaron un esquema de partición e indexación en línea, junto con un sintonizador de partición e indexación diseñado para motores de consulta *in situ*. El diseño del sistema propuesto mejoró el tiempo de ejecución de la consulta teniendo en cuenta los patrones de consulta del usuario. El sistema divide lógicamente los archivos de datos brutos y crea índices específicos de partición livianos para cada fragmento. Se creó un motor de consulta *in situ* llamado Slalom para mostrar el impacto del diseño propuesto. Como resultado de su naturaleza liviana, Slalom logra un procesamiento eficiente de consultas sobre datos sin procesar con un consumo mínimo de memoria. Los autores mostraron en la etapa de implementación que mediante los *microbenchmarks* y cargas de trabajo reales, Slalom superó a los motores *in situ* de última generación y logró tiempos de respuesta de consulta comparables con SGBD's completamente indexados, ofreciendo tiempos de ejecución de consulta acumulados más bajos para cargas de trabajo de consulta con un tamaño creciente y patrones de acceso impredecibles.

2.2.2. Fragmentación vertical dinámica

La partición vertical se ha empleado ampliamente en bases de datos relacionales para mejorar

el tiempo de respuesta de las consultas. Sin embargo, la mayoría de los enfoques de partición vertical son estáticos. Se crea un esquema de fragmentación vertical (VPS) de acuerdo con un conjunto de consultas previamente conocido. Si este conjunto sufre cambios, es posible que el VPS se degrade. Como resultado, el tiempo de respuesta de las consultas aumenta. Rodríguez et al. (2013) presentaron la mejora en DYVEP (*DYnamic VErtical Partitioning*, fragmentación vertical dinámica), que se desarrolló como un sistema activo con capacidad de partición dinámica. El sistema fragmenta y vuelve a fragmentar verticalmente una base de datos sin la intervención de un administrador de base de datos y mantiene un tiempo de respuesta de consulta aceptable. Se realizó la implementación sobre la base de datos PostgreSQL y se utilizó el *benchmark* TPC-H. Como conclusión se mencionó que DYVEP desempeña la fragmentación vertical dinámica y que sus principales ventajas son:

- DYVEP implementa un recopilador de estadísticas basado en reglas activas que acumula información sobre atributos, consultas y fragmentos sin intervención explícita del DBA (*Database Administrator,* Administrador de base de datos).
- En la fragmentación realizada por DYVEP, la configuración cambia dinámicamente de acuerdo con los cambios en la información de las consultas y es capaz de encontrar la mejor solución sin afectar el rendimiento de la base de datos.
- En DYVEP, todo el proceso de fragmentación vertical se implementa dentro de la base de datos utilizando *triggers*, la matriz de uso de atributos (AUM) utilizada por la mayoría de los algoritmos de fragmentación vertical se implementa como una tabla de la base de datos (AUT) y esta utiliza *triggers* para cambiar la configuración de los fragmentos automáticamente.
- DYVEP tiene un generador de particiones basado en reglas activas que crean automáticamente los fragmentos en el disco.
- Este enfoque implementa los fragmentos como índices, por lo tanto, el optimizador de consultas del sistema gestor de bases de datos es capaz de detectar los índices y reescribir las consultas para acceder a ellas en lugar de las tablas completas. Como resultado, los fragmentos son transparentes para los usuarios.

Pérez et al. (2000), presentaron una extensión del modelo de optimización matemática DFAR (*Dynamic Fragmentation, Allocation and Reallocation*, Fragmentación dinámica, asignación

y Reasignación), que unifica la fragmentación, asignación y migración dinámica de datos en sistemas de bases de datos distribuidas. La extensión consistió en la adición de una restricción que modela la capacidad de almacenamiento de los sitios de red. Este aspecto es particularmente importante en grandes bases de datos, que exceden la capacidad de uno o más sitios. El algoritmo de aceptación de umbral es una variación del método heurístico conocido como Simulated Annealing (Recocido Simulado), y se utiliza para resolver el modelo DFAR. Para obtener los resultados del enfoque propuesto se generaron las mejores estrategias para casos de prueba grandes, soluciones factibles y el desarrollo del algoritmo para la regulación de parámetros. Los experimentos iniciales revelaron que cuando la capacidad del sitio se usa casi hasta su límite, se producen puntos muertos de atributos (es decir, el intercambio de posiciones de atributos se bloquea), evitando que converja el algoritmo de aceptación de umbral. Los autores mostraron que la inclusión de un sitio artificial y la adición de un término de penalización a la función objetivo, previene puntos muertos de atributos y permite la convergencia del algoritmo. Como conclusión se mencionó que el trabajo mostró cómo expandir el modelo DFAR integrando reglas de capacidad para los sitios.

Li y Gruenwald (2013) introdujeron un nuevo algoritmo llamado Fragmentador en Línea Autoadministrado para Bases de Datos (SMOPD, *Self-Managing Online Partitioner for Databases*), el cual fragmenta verticalmente mediante la minería de conjuntos de elementos cerrados a partir de un conjunto de consultas e información estadística del sistema. Este algoritmo monitorea dinámicamente el desempeño de la base de datos usando parámetros configurados por el usuario y automáticamente detecta la tendencia en el desempeño para que se decida cuándo desempeñar una re-fragmentación sin la intervención del administrador de bases de datos. Este algoritmo libera al administrador de bases de datos de la difícil tarea de monitorear el sistema y esforzarse contra las largas tablas de estadística. El objetivo de SMOPD es localizar de las tablas de base de datos el tiempo promedio de respuesta que se hace más grande para un conjunto de consultas de acuerdo a la solución actual de fragmentación y proponer la nueva solución de la mejor fragmentación de todos los esquemas de fragmentación que se generan. Se menciona como trabajo a futuro que la investigación se extenderá para computadoras en *clusters*.

Las bases de datos distribuidas en *clusters* se usan ampliamente en muchas aplicaciones. Con la demanda creciente en el volumen de datos y la necesidad de mejorar la velocidad de respuesta en las consultas de los datos es importante desarrollar técnicas que mejoren el tiempo de respuesta de las consultas para satisfacer las necesidades de las aplicaciones. Una de ellas es la fragmentación vertical de bases de datos, la cual divide una tabla en tablas más pequeñas que contienen menos atributos, para que de esta manera se reduzcan las operaciones hechas a la memoria. Si bien se han desarrollado muchos algoritmos para la fragmentación vertical de bases de datos, ninguno de ellos está diseñado para fragmentar dinámicamente bases de datos almacenadas en *clusters*, es decir, sin interferencia humana y sin cargas de trabajo de consultas fijas. Para resolver este problema, Li & Gruenwald (2014) presentaron un algoritmo dinámico, SMOPD-C, que fragmenta de forma autónoma una base de datos distribuida verticalmente en *clusters*, determinar cuándo se necesita una nueva fragmentación y volver a dividir la base de datos en consecuencia. Luego, el trabajo presentado muestra diferentes experimentos que se llevaron a cabo para estudiar el rendimiento de SMOPD-C utilizando el *benchmark* TPC-H en un *cluster* de computadoras. Los resultados del experimento mostraron que SMOPD-C es capaz de realizar una fragmentación dinámica con alta precisión y de obtener un menor costo en la ejecución de las consultas frente a otros enfoques.

Los sistemas modernos de bases de datos están diseñados en torno a un único diseño de almacenamiento de datos. Sin embargo, los enfoques que implementan estos diseños no son universalmente eficientes; diferentes cargas de trabajo requieren diferentes diseños de almacenamiento y métodos de acceso a datos para lograr un rendimiento adecuado. Alagiannis, Idreos & Ailamaki (2014) presentaron el sistema H2O que permitió la flexibilidad para soportar múltiples diseños de almacenamiento y patrones de acceso a datos en un solo motor. Además, decide sobre la marcha, es decir, durante el procesamiento de consultas, elige el mejor diseño para la carga específica de trabajo. De esta manera, H2O no toma decisiones *a priori* y fijas sobre cómo se deben almacenar los datos, lo que permite que cada consulta disfrute de un patrón de almacenamiento y acceso que se adapta a sus propiedades específicas. Se presentó un análisis detallado de H2O utilizando el *benchmark* SDSS (Sloan Digital Sky Survey, *Encuesta Sloan del Cielo Digital*). Se demostró que, si bien los sistemas existentes no logran el máximo rendimiento en todas las cargas de trabajo, H2O

siempre iguala el rendimiento del mejor caso sin requerir ningún conocimiento de ajuste o carga de trabajo especifica.

Encontrar el esquema de partición vertical adecuado para una carga de trabajo es uno de los problemas esenciales de optimización de la base de datos. Con la partición adecuada, las consultas y las tareas de administración logran omitir datos innecesarios, mejorando su rendimiento. Durand et al. (2019) consideraron la viabilidad de una solución general de aprendizaje automático para superar los inconvenientes de los enfoques más comunes. Se amplió el trabajo en GridFormation (Durand et al, 2018), asignando la tarea de partición a una tarea de RL (*Reinforcement learning*, Aprendizaje de Refuerzo). Se validó la propuesta experimentalmente utilizando una base de datos y una carga de trabajo con el *benchmark* TPC-H y el marco de trabajo de Google Dopamine para RL profundo. Se presentaron tiempos de ejecución competitivos mientras aumenta el número de atributos en una tabla, superando a algunos algoritmos de vanguardia.

Los datos semiestructurados como JSON se han convertido en el estándar más usado para soportar el intercambio de datos en la Web. Al mismo tiempo, el soporte relacional para los datos JSON plantea nuevos desafíos debido a la gran cantidad de atributos, atributos dispersos y cambios dinámicos tanto en la carga de trabajo como en el conjunto de datos, que son típicos en dichos datos. Sharify et al. (2019) abordaron estos desafíos a través de un prototipo de motor de base de datos relacional ligero y un algoritmo de partición vertical flexible que utilizó heurística simple para adaptar el diseño de datos a la carga de trabajo. La evaluación experimental utilizando el conjunto de datos de Nobench para datos JSON, mostró que se superó a Argo, un modelo de datos de última generación que también asigna el formato de datos JSON en bases de datos relacionales. También se superó a Hyrise, un algoritmo de partición vertical de última generación, en un 24%. Además, el algoritmo propuesto fue capaz de lograr una utilización de caché alrededor de un 40% mejor y una utilización de TLB (*Translation Lookaside Buffer*, Búfer de Traducción Adelantada) un 35% mejor. Los experimentos llevados a cabo mostraron que el algoritmo de partición se adaptó a los cambios de carga de trabajo en unos pocos segundos.

Schroeder et al. (2020) presentó un método de distribución de datos RDF (*Resource*

Description Framework, Marco de Trabajo de Descripción de Recursos) que supera las deficiencias de los enfoques actuales para escalar el almacenamiento RDF tanto en el volumen de datos como en el procesamiento de consultas. Este enfoque se implementó en una vista resumida de datos para evitar un análisis exhaustivo de grandes conjuntos de datos. Como resultado, las plantillas de fragmentación se derivaron de elementos de datos en una estructura RDF. Además, se proporcionó un enfoque para insertar datos dinámicos, incluso si los nuevos datos no se ajustan a la estructura RDF original.

2.2.3. Fragmentación híbrida dinámica

Wang et al. (2014) solucionaron el problema de la distribución de datos mediante un modelo de triángulo general llamado DaWN (*Data Workload and Nodes*, Carga de Trabajo de Datos y Nodos). Con base en el análisis de datos y carga de trabajo, se presentó una estrategia novedosa llamada ADDS (*Automatic Data Distribution Solution*, Solución de Distribución Automática de Datos) para la distribución automática de datos en aplicaciones OLTP. La evaluación de este enfoque se llevó a cabo utilizando el *benchmark* TPC-C y la base de datos MySQL. Los autores compararon tres estrategias diferentes: Hashing, Round-Robin y ADDS. Según los resultados de una serie de experimentos sobre conjuntos de datos y transacciones de TPC-C, el enfoque propuesto mostró mejoras efectivas. Como trabajo a futuro mencionaron que hay diferentes aspectos a mejorar: el modelo DaWN necesita ser instanciado en diferentes tipos de aplicación *Big Data*, mientras que la estrategia ADDS necesita ajustarse a los requerimientos de datos reales y no solo simulados bajo el *benchmark*.

Chen et al. (2015) propusieron SOAP (*Scheduling Online Database Repartitioning*, Planificación en línea de Refragmentación de Bases de Datos), un marco de trabajo de un sistema para programar la refragmentación de bases de datos en línea para cargas de trabajo OLTP. SOAP cumple el objetivo de minimizar el período de tiempo de ejecución de las operaciones de refragmentación al mismo tiempo que garantiza la corrección y el rendimiento del procesamiento concurrente de las transacciones normales. SOAP empaqueta las operaciones de refragmentación en transacciones, y luego las mezcla con las transacciones normales para la optimización de la programación holística. SOAP utiliza un enfoque basado en costos para clasificar las prioridades de programación de las transacciones de refragmentación, y aprovecha un modelo de retroalimentación en la teoría de control para

determinar en qué orden y con qué frecuencia las transacciones de refragmentación deben programarse para su ejecución. Se creó un prototipo sobre PostgreSQL y se realizó un estudio experimental exhaustivo sobre Amazon EC2 (*Elastic Compute Cloud*, Computación Elástica en la Nube) para validar las importantes ventajas de rendimiento de SOAP.

Durand et al. (2018) mostraron el desarrollo de un marco de trabajo que soportó el ajuste autónomo en línea de la fragmentación de datos y presentó capas con una formulación de aprendizaje de refuerzo. Se establecieron los elementos centrales del enfoque propuesto: agente, entorno, espacio de acción y componentes de soporte. El entorno facilitó el proceso de búsqueda al generar recompensas inmediatas basadas en nuevos cálculos de costos, ya sea para la totalidad o una muestra de consultas de la carga de trabajo. Estos conjuntos de acciones se configuran, permitiendo la representación de diferentes problemas de partición. Para el uso de este enfoque en un entorno en línea, el agente es capaz de aprender una secuencia de longitud fija de *n* acciones que maximizan la recompensa temporal por la carga de trabajo prevista. A través de la implementación inicial, se afirmó la viabilidad del enfoque presentado.

Las bases de datos relacionales (RDB, *Relational Databases*) tradicionales en general no son adecuadas para los requisitos que exigen los grandes volúmenes de información que se manejan en los miles de transacciones OLTP por segundo. NewSQL es una nueva generación de bases de datos que combina la alta escalabilidad y disponibilidad con el soporte ACID. NewSQL es una solución prometedora para manejar estos requisitos de aplicación. Schreiner et al. (2018) propusieron un enfoque automatizado para la fragmentación de datos híbridos que reorganiza automáticamente los datos en función de la carga de trabajo actual de las bases de datos NewSQL. El módulo de fragmentación se encarga de monitorear el desempeño del enfoque y planes óptimos de fragmentación para las demandas de la aplicación. Se mencionó que el módulo se organiza en tres componentes: monitor, administrador y el generador de planes de fragmentación. Como conclusión los autores mencionaron que el trabajo propuesto no tiene precedentes en la literatura, ya que es la única investigación que propone un enfoque de fragmentación híbrido, ofreciendo la optimización y el almacenamiento de datos basado en la carga de trabajo.

Kulba & Somov (2020) analizaron la asignación dinámica de fragmentos en sistemas de procesamiento de datos distribuidos. Se presentó un algoritmo heurístico para colocar fragmentos en función de los parámetros de cada sistema a lo largo del tiempo. Se utilizaron dos métodos principales de fragmentación de datos: fragmentación horizontal y vertical. Los autores presentaron un método para la redistribución dinámica de fragmentos de tablas entre los nodos de un sistema distribuido, teniendo en cuenta los valores actuales de los parámetros del sistema, que cambian con el tiempo.

2.2.4. Otras fragmentaciones dinámicas

Las particiones verticales y horizontales permiten a los administradores de bases de datos (DBA) mejorar considerablemente el rendimiento de las aplicaciones de inteligencia empresarial. Sin embargo, encontrar y definir particiones horizontales y verticales adecuadas es una tarea desalentadora incluso para los DBA experimentados. Esto se debe a que el DBA tiene que comprender muy bien los planes de ejecución de consultas físicas para cada consulta en la carga de trabajo para tomar decisiones de diseño apropiadas. Jindal & Dittrich (2012) presentaron AutoStore: un almacén de datos de autoajuste que monitorea la carga de trabajo actual y divide los datos automáticamente en intervalos de tiempo de control, sin intervención humana. Esto permitió que AutoStore se adapte a las cargas de trabajo sin detener el sistema. Los resultados experimentales se obtuvieron mediante el conjunto de datos del *benchmark* TPC-H y mostraron que AutoStore superó los diseños de filas y columnas hasta en un factor de 2.

Sleit et al. (2007) propusieron una mejora para el algoritmo ADRW (*Adaptive Distributed Request Window*, Ventana de Solicitud Distribuida Adaptable) (Lin & Veeravalli, 2003) para lograr la fragmentación dinámica y la asignación de objetos en bases de datos distribuidas. El algoritmo se adaptó a los patrones cambiantes de las solicitudes de objetos con el objetivo de ajustar dinámicamente los esquemas de asignación de objetos para minimizar el costo total de servicio de todas las solicitudes. Los objetos pueden ser replicados o fragmentados dependiendo de los patrones de lecturas y escrituras. El trabajo presentado por los autores contempla los siguientes costos:

- Costo de enviar la consulta del objeto del procesador solicitante al procesador principal.

- Costo de recuperar y/o actualizar el objeto desde la memoria local del procesador o los procesadores que alojan al objeto.
- Costo de transferir el objeto desde la memoria principal del procesador de alojamiento al procesador solicitante.

Como conclusión se mencionó que la principal ventaja del enfoque propuesto, E-ADRW (*Enhanced Adaptive Distributed Request Window*, Ventana de Solicitud Distribuida Adaptable Mejorada) fue que requiere menos almacenamiento en comparación con los algoritmos óptimo y umbral mostrados en la etapa de evaluación.

El sistema de *cluster* MMDB (*Main Memory Database*, Base de Datos de Memoria Principal) es una base de datos relacional optimizada de memoria que se implementa en el entorno de *cluster* computacional, brinda a las aplicaciones un tiempo de respuesta extremadamente rápido y un rendimiento muy alto, como lo requieren muchas aplicaciones en una amplia gama de industrias. Hung & Huang (2012) propusieron un nuevo algoritmo de asignación dinámica de fragmentos (DFAPR, *Dynamic Fragment Allocation Algorithm in Partially Replicated Allocation Scenario*, Algoritmo de Asignación Dinámica de Fragmentos en un Escenario de Asignación Parcialmente Replicado). Este algoritmo reasigna datos con respecto al cambio en los patrones de acceso a cada fragmento. El enfoque propuesto migra o crea nuevas réplicas en sitios remotos, dependiendo de la frecuencia de acceso y el tiempo de respuesta promedio. Los resultados que se mostraron en la simulación demostraron que DFAPR es adecuado para el *cluster* MMDB debido a que proporciona un mejor tiempo de respuesta y maximiza el procesamiento local. Como conclusión se mencionó la importancia del trabajo realizado por los autores, ya que representó un esfuerzo significativo para minimizar la cantidad de datos transferidos durante el procesamiento y proporcionar el procesamiento local más alto para maximizar el número de aplicaciones paralelas.

El objetivo final de los desarrolladores de bases de datos es implementar un DBMS (*Database Management System*, Sistema Gestor de Bases de datos) totalmente autogestionado que no requiera la intervención del administrador. Aunque hay algunos avances en este campo, la tecnología autogestionada no está lista para su uso industrial y sigue siendo un área activa de investigación. Una de las tareas más cruciales para un sistema

de este tipo es el ajuste automático del diseño físico. Un enfoque autogestionado para esta tarea implica que el diseño físico de una base de datos debe adaptarse automáticamente a las cargas de trabajo cambiantes. Chernishev (2017) presentó un análisis profundo de las perspectivas de un almacén de columnas relacional distribuido adaptativo. Se demostró que el enfoque del almacenamiento de columnas es un gran avance para la construcción de una base de datos eficiente autogestionada. Se presentó una breve encuesta de los estudios de diseño físico existentes y se ofrece una clasificación de estos enfoques. Luego, se proporcionaron algunos puntos de vista sobre la organización de un sistema de almacén de columnas distribuido autogestionado. Como conclusión se mencionó que se presentaron diferentes opciones de diseño físico para crear el enfoque deseado, así como tres alternativas para diseñar una alerta y ejecutar la reorganización de la base de datos.

2.3. Fragmentación para SGBD NoSQL

Para optimizar el tiempo de respuesta de una aplicación web utilizando una base de datos basada en la nube, es posible dividir la base de datos para almacenarla en servidores locales. Sauer & Wei Hao (2015) utilizaron la minería de datos y el análisis de conglomerados en los registros de la base de datos para aplicar la fragmentación de datos. Los autores compararon los tiempos de respuesta promedio de tres algoritmos relacionados en una aplicación web simple usando un sistema de gestión de bases de datos NoSQL basado en la nube. El estudio experimental mostró que las técnicas presentadas mejoran el rendimiento de las aplicaciones web. Se llevaron a cabo diferentes pruebas, las cuales utilizaron diferentes esquemas de navegación simulados que generan el tráfico deseado. Se obtuvo el tiempo de respuesta de las solicitudes desde PHP para analizar los aciertos de asignación. Como conclusión se mencionó que el trabajo propuesto superó el rendimiento de una base de datos sola en la nube o usando el algoritmo *round robin* para desempeñar la fragmentación.

Una de las técnicas clave para resolver los desafíos de *Big Data* es introducir sistemas de almacenamiento escalables. Las bases de datos NoSQL se consideran sistemas eficientes de gestión de almacenamiento de *Big Data* que proporcionan escalabilidad horizontal. Para garantizar la escalabilidad del sistema, las estrategias de fragmentación de datos deben implementarse en estas bases de datos. Elghamrawy (2016) propuso un Módulo de Fragmentación Adaptable de *Hashing* de Encuentro (ARHPM, *Adaptive Rendezvous Hashing*

Partitioning Module) para las bases de datos Cassandra NoSQL. El objetivo principal de este módulo fue fragmentar los datos en Cassandra utilizando el *hashing* de encuentro y proponer un algoritmo de *hashing* de encuentro basado en el balanceo de cargas para garantizar un óptimo desempeño en el proceso de partición. Para evaluar el módulo propuesto, Cassandra se modificó incorporando el módulo de partición y se llevó a cabo una serie de experimentos para validar el equilibrio de carga del módulo propuesto mediante el uso de *Yahoo! Cloud Serving Benchmark* (*Benchmark* de servicio en la nube de *Yahoo!*). Los dos experimentos mostraron que el algoritmo propuesto es más rápido al fragmentar y se obtiene un mejor esquema en términos de desempeño. Como un trabajo futuro se mencionó que se utilizará el método de filtro de floración para mejorar el tiempo de recuperación *hash*.

Elghamrawy & Hassanien (2017) propusieron un marco MR-RHVH (*MapReduce Rendezvous Hashing-Based Virtual Hierarchies*, Jerarquías Virtuales Basadas en *Hashing* de Encuentro *MapReduce*) para la partición escalable de la base de datos Cassandra NoSQL. El marco MapReduce se utilizó para implementar MR-RHVH en Cassandra para mejorar su rendimiento en entornos altamente distribuidos. MR-RHVH distribuye los nodos para reunir regiones mediante una estrategia de jerarquía virtual adoptada. Cada región es responsable de un conjunto de nodos. Además, se utiliza un evaluador de filtro de floración propuesto para garantizar la asignación precisa de claves a los nodos en cada región. Se llevaron a cabo diferentes experimentos usando el *benchmark* YCSB (*Yahoo! Cloud Serving Benchmark*, Benchmark de servicio en la nube de Yahoo!). Los resultados mostraron que el algoritmo es altamente escalable y se ejecuta en un tiempo menor frente a diferentes sistemas recientes. Como conclusión se mencionaron las ventajas del enfoque propuesto y se presentó como trabajo a futuro el cambio de Hadoop a Spark.

MongoDB es un NoSQL basado en documentos que se desarrolló para responder a las crecientes demandas de almacenamiento. Para lograr un buen rendimiento, el sistema debe diseñarse correctamente desde el principio. Diferentes investigaciones se centraron en aplicar técnicas de equilibrio de carga para resolver el problema del punto de acceso, el cual afecta el desempeño general del sistema. Sin embargo, estas técnicas no son efectivas para los datos de series de tiempo, como los registros del sistema en tiempo real. Oonhawat & Nupairoj (2017) desarrollaron un nuevo algoritmo de distribución de datos basado en la fragmentación

consciente de etiquetas para minimizar el efecto del problema del punto de acceso, especialmente en los sistemas con requisitos de escritura pesados. Como conclusión se mencionó que el sistema mejoró debido a que es menos probable que cierto fragmento se vea afectado durante las consultas por el problema del punto de acceso. Como trabajo a futuro mencionaron que estudiarán el impacto de una mala predicción causada por patrones de acceso cambiantes y tomarán en cuenta las capacidades de cada servidor para reducir su carga de trabajo.

La seguridad en *Big Data* es uno de los principales desafíos para los proveedores de información. De hecho, la falta de una solución metodológica sólida dedicada a la seguridad de *Big Data* hace que los temas de privacidad y protección de datos personales sean áreas de investigación importantes. Debido al gran volumen de datos que se propaga entre las redes sociales y las aplicaciones orientadas a la nube, se debe pensar en un enfoque que mejore la seguridad de los datos en las bases de datos, específicamente en el contexto de los entornos NoSQL. Heni & Gargouri (2015) presentaron un nuevo enfoque metodológico para la seguridad de *Big Data* basado en la fragmentación de datos. El desarrollo propuesto muestra cuatro fases. La primera fase se utiliza para agrupar automáticamente *Big Data* en la base de datos NoSQL. La segunda fase consiste en la identificación de datos sensibles mediante una red neuronal. La tercera fase permite proveer una capa de seguridad a través de la fragmentación. La última etapa tiene por objetivo la reconstrucción de los fragmentos. Como conclusión se mostró el objetivo principal de la investigación presentada, el cual fue superar los problemas en el entorno de *Big Data*, especialmente para las bases de datos NoSQL. En este contexto, utilizaron la fragmentación de datos combinada con un cifrado de datos aplicado en MongoDB.

La fragmentación de datos en las unidades flash SSD (*solid state drives*, unidades de estado sólido) es un problema común que conduce a la degradación del rendimiento, especialmente cuando los dispositivos de almacenamiento subyacentes envejecen al actualizar en gran medida las cargas de trabajo. Nguyen & Lee (2017) abordaron este problema en MongoDB. Presentaron un nuevo esquema de mapeo de flujo basado en características únicas de MongoDB. Los resultados que se mostraron al evaluar los enfoques, presentaron mejoras en el rendimiento de dos *benchmarks,* YCSB y Linkbench. El trabajo presenta detalladamente la

fragmentación en MongoDB y los beneficios del enfoque propuesto. Como conclusión se menciona que adicionalmente se minimizó el tamaño máximo de la página en los archivos recuperados y esto mejoró significativamente el rendimiento de los dos conjuntos de datos que se utilizaron para la evaluación.

El almacenamiento en la nube es muy popular entre los usuarios, ya que demostró ser una solución rentable para guardar y procesar datos. Sin embargo, en los entornos en la nube, los proveedores de servicios poseen todo el control de los datos de los usuarios. Por esta razón surgen dos problemas principales en este contexto, la seguridad y la integridad de los datos. Santos, Ghita & Masala (2019) introdujeron un enfoque para la seguridad de los datos en la nube utilizando un algoritmo de fragmentación de patrones aleatorios y combinándolo con una base de datos NoSQL distribuida. Esto no solo aumentó la seguridad de los datos al almacenarlos en diferentes nodos, sino que también permitió al usuario implementar esta nueva alternativa para asegurar los datos. Se realizaron los experimentos en un conjunto de datos con cuatro tipos de datos. Cada archivo contuvo 100 kB de tamaño y se almacenó en Cassandra. Cassandra se desplegó en la infraestructura de Microsoft Azure. Los resultados mostraron un mayor rendimiento en comparación con sus contrapartes, lo que implica la usabilidad del método propuesto en la computación en la nube, especialmente en escenarios con necesidades de alta velocidad y recursos limitados.

2.4. Otros tipos de fragmentación

En esta sección se agrupan todos los trabajos que no están incluidos en el resto de clasificaciones, es decir, que no se centran en bases de datos multimedia, que no realizan fragmentación dinámica y que no están desarrollados para bases de datos NoSQL.

2.4.1. Otros tipos de fragmentación horizontal

Castro-Medina et al. (2020) mostraron un análisis de diferentes métodos de fragmentación, asignación y replicación de bases de datos y una aplicación web denominada FRAGMENT que adoptó la técnica de trabajo que se seleccionó en la etapa de análisis. La selección contempló las siguientes características:

- Presenta un método de fragmentación y replicación.
- Es aplicado a un ambiente de la nube.

- Facilidad de implementación.
- Se enfoca en mejorar el desempeño de las operaciones ejecutadas en la base de datos.
- Muestra todo lo necesario para su implementación.
- Se basa en un modelo de costos.

Los experimentos, utilizando el *benchmark* TPC-E, demostraron un menor tiempo de respuesta de las consultas ejecutadas en la base de datos distribuida generada por FRAGMENT frente a una base de datos centralizada.

La fragmentación horizontal derivada es una de las principales técnicas de diseño de distribución de bases de datos. A diferencia de la fragmentación horizontal primaria, la decisión de la fragmentación horizontal derivada no es sencilla. Ma, Schewe & Wang (2007) observaron que la fragmentación y la asignación a menudo se consideran por separado, sin tener en cuenta que están utilizando la misma información de entrada para lograr el mismo objetivo, es decir, mejorar el rendimiento general del sistema. Los autores abordaron la fragmentación y asignación horizontal derivada simultáneamente en el contexto del modelo de datos complejos. El objetivo principal fue proporcionar un enfoque manejable para minimizar los costos de procesamiento de consultas realizando fragmentación horizontal y asignación de fragmentos simultáneamente. Los resultados demostraron que el enfoque heurístico presentado para la fragmentación horizontal derivada mejoró el rendimiento del sistema frente a otros enfoques tradicionales de fragmentación. Como conclusión se mencionó que se propuso un enfoque heurístico para minimizar el costo de procesamiento de las consultas.

Bellatreche et al. (2011) desarrollaron un algoritmo combinado que maneja el problema de la dependencia entre la fragmentación y la asignación. Se desarrolló una nueva solución genética para resolver este obstáculo. El problema principal de esta solución y las soluciones anteriores es la falta de validación de estos modelos en un entorno real. La investigación que presentan los autores abordó esta brecha verificando la efectividad de la solución genética que se desarrolló en trabajos anteriores en el sistema gestor de bases de datos Teradata. Teradata es un SGBD (Sistema Gestor de Bases de Datos) que provee escalabilidad y solidez en entornos de usuario reales de hasta 10 petabytes de datos relacionales. Se llevaron a cabo

experimentos para la solución genética y el trabajo previo utilizando el *benchmark* SSB (*Star Schema Benchmark, Benchmark* de esquema en estrella) aplicándolo en Teradata con el *software* TD 13.10. Los resultados mostraron que la solución genética es más rápida que el trabajo anterior en un 38%. Se presentó una descripción detallada de la implementación del enfoque propuesto en la que se especificó que se utilizaron 22 consultas de las cuales se obtuvieron 50 predicados seleccionados. Los experimentos se aplicaron a dos enfoques diferentes, *Joint* y *Sequential.* Como trabajo a futuro mencionaron que se incorporará a Teradata un modelo de costos en la solución *Joint.*

Una de las ventajas clave de la computación en la nube es la elasticidad en la cual los recursos informáticos, como las máquinas virtuales, pueden aumentarse o disminuirse. Para aprovechar la elasticidad que ofrece la computación en la nube, los sistemas de procesamiento de SQL basados en la nube necesitan la capacidad de repartir los datos fácilmente cuando el número de nodos de la base de datos aumenta o disminuye. Lim (2013) investigó el soporte a la fragmentación elástica de datos en sistemas de procesamiento de SQL paralelo basados en la nube. La autora propuso diferentes algoritmos y técnicas de organización de datos asociadas que minimizaron el reparto de tuplas y el movimiento de datos entre nodos. La evaluación experimental demostró la efectividad de los métodos propuestos. Como trabajo a futuro se mencionó que abordará el tema de la distribución de datos asimétricos.

Diferentes trabajos de investigación proporcionan métodos de fragmentación basados en datos empíricos como el tipo y la frecuencia de las consultas. Estas soluciones no son adecuadas en la etapa inicial de una base de datos distribuida, ya que las estadísticas de las consultas no están disponibles en ese momento. Islam Khan (2016) presentó una técnica llamada MMF (*Matrix Based Fragmentation*, Fragmentación Basada en Matrices), que se puede aplicar tanto en la etapa inicial como en las etapas posteriores del diseño de las bases de datos distribuidas. En lugar de utilizar datos empíricos, se desarrolló una matriz llamada MCRUD. Hacer una adecuada fragmentación de las relaciones y la asignación de los fragmentos es un área importante de investigación en sistemas distribuidos. Con esta técnica, no se agrega complejidad adicional para asignar los fragmentos a los sitios de una base de datos distribuida, ya que la fragmentación se sincroniza con la asignación. Por lo tanto, el

rendimiento de un DDBMS (Distributed Database Management System, Sistema Gestor de Bases de Datos Distribuidas) mejora significativamente al evitar el acceso remoto frecuente y la alta transferencia de datos entre los sitios.

2.4.2. Otros tipos de fragmentación vertical

El trabajo presentado por Fung, Karlapalem y Li (2002) mostró el desarrollo de un modelo de costos analítico y completo para el procesamiento de consultas en clases de bases de datos orientadas a objetos fragmentadas verticalmente. Se presentó un conjunto de resultados de evaluaciones analíticas para mostrar el efecto de la fragmentación vertical y para estudiar la relación entre la proporción de proyección y el factor de selectividad del acceso secuencial vis a vis el acceso por índice. Posteriormente se mostró la implementación de un prototipo experimental que permite la fragmentación vertical de clases en una base de datos comercial orientada a objetos para validar el modelo de costos. Después de dos experimentos empíricos se obtuvieron las siguientes conclusiones:

- La fragmentación vertical reduce el acceso a disco para el procesamiento de consultas.
- Los cálculos teóricos basados en el modelo de costos propuesto dan límites realistas bajos y altos para los accesos actuales de entrada y salida a disco hechos por el procesamiento de consultas para bases de datos orientadas a objetos.
- La fragmentación vertical favorece a objetos de gran tamaño.

Como trabajo a futuro se mencionó un nuevo modelo de costos para escenarios fragmentados y no fragmentados, así como el desarrollo de un nuevo algoritmo para generar una fragmentación vertical óptima para cierto conjunto de consultas.

El trabajo previo de Fung, Karlapalem y Li (2003) fue mostrar que la fragmentación vertical de clases produce un decaimiento sustancial en el número total de accesos a disco usados para la ejecución de un conjunto de aplicaciones, pero encontrar un esquema óptimo de fragmentación vertical de clases es un problema difícil. En el nuevo enfoque se presentaron dos algoritmos para derivar esquemas de fragmentación vertical de clases óptimos y casi óptimos. El algoritmo basado en costos produce esquemas de fragmentación vertical de clases óptimos, enumerando exhaustivamente todos los esquemas y calculando el número de

accesos a disco requeridos para ejecutar un conjunto dado de aplicaciones. Para esto se desarrolló un modelo de costos para la ejecución de un conjunto de métodos en un sistema de base de datos orientado a objetos. Debido a que la enumeración exhaustiva es costosa y sólo trabaja para clases con números pequeños de variables de instancia, se desarrolló un algoritmo heurístico de ascenso de colinas (HCHA, *Hill-Climbing Heuristic Algorithm*) que toma la solución dada por un algoritmo basado en afinidad y la mejora, reduciendo así el número total de accesos a disco. Como conclusión se mencionó que se desarrollaron dos algoritmos que explotan la ejecución de los métodos para producir esquemas de fragmentación. El primer algoritmo usa un modelo de costos para la ejecución de métodos para buscar exhaustivamente todos los esquemas de fragmentación y encontrar el esquema óptimo de fragmentación vertical. Este algoritmo es útil para la comparación de efectividad de otros algoritmos, pero no es práctico ya que es altamente costoso. El segundo algoritmo, HCHA, aplica un algoritmo basado en afinidad para obtener un esquema de fragmentación inicial, después aplica heurísticas escalables para mejorar la solución usando el modelo de costos para la ejecución de métodos. Se demostró que este último es significativamente más eficiente que el enfoque basado en costos. Como trabajo a futuro se incluirán aplicaciones de la técnica de fragmentación vertical de clases al problema de la optimización de consultas y al desarrollo de aplicaciones multimedia involucrando objetos de gran tamaño.

Hadoop se ha convertido en una arquitectura líder para el procesamiento de datos a gran escala. Una de las formas eficientes de acelerar el procesamiento de datos es la técnica de almacenamiento orientada a columnas que se ha integrado recientemente en la familia Hadoop. Sin embargo, diseñar un algoritmo de agrupación de atributos apropiado para lograr un rendimiento óptimo de procesamiento de datos en el entorno de Hadoop orientado a columnas fue un gran obstáculo. Gu et al. (2012), propusieron un nuevo algoritmo llamado CHAC (*Column-oriented Hadoop based Attribute Clustering*, Agrupamiento de Atributos basado en Hadoop Orientado a Columnas) para resolver este problema. Ambos casos de agrupación de atributos traslapados y no traslapados se consideran en CHAC. Además, también se tuvo en cuenta un parámetro ajustable para prohibir la redundancia excesiva de atributos limitando la sobrecarga de espacio. Para realizar las pruebas se utilizó el *benchmark* TPC-H y el algoritmo se evaluó en 16 nodos de los cuales uno de ellos fungió como maestro. La base de datos contuvo 30 atributos y 20 GB de tamaño. Se observó que los resultados

generados por el modelo de costos están estrechamente relacionados con el tiempo de ejecución de las consultas en las fases del mapeo, ya que su tendencia es consistente, lo que indica la efectividad del modelo de costo propuesto.

Las bases de datos tradicionales no están equipadas con la funcionalidad adecuada para manejar el volumen y la variedad de *Big Data*. Zhao, Cheng & Rusu (2015) investigaron el problema del procesamiento de datos sin procesar impulsado por cargas de trabajo con carga parcial. Se modeló la carga como fragmentación vertical binaria completamente replicada. Se proporcionó una formulación lineal de optimización de programación de enteros mixtos que demostró ser NP-hard. Se diseñó una heurística con dos etapas que encuentra una solución cercana a la óptima en una fracción del tiempo. Se extendió la formulación de optimización y la heurística para el procesamiento lineal de datos sin procesar, escenario en el que el acceso y la extracción de datos se ejecutan simultáneamente. Se proporcionaron tres casos de estudio sobre formatos de datos reales que confirmaron la precisión del modelo cuando es implementado en un operador lineal del estado del arte para el procesamiento de datos sin procesar.

Costa, Costa & Santos (2019) mostraron diferentes estudios para comprender las formas de optimizar el rendimiento de varios sistemas de almacenamiento para *Big Data Warehousing.* Se mencionó que pocos de ellos exploran el impacto de las estrategias de organización de datos en el rendimiento de las consultas cuando utilizan Hive como tecnología de almacenamiento para implementar sistemas de *Big Data Warehousing*. Por esta razón, los autores evaluaron el impacto de la partición y el almacenamiento de datos en sistemas basados en Hive, probando diferentes estrategias de organización de datos y verificando la eficiencia de esas estrategias en el rendimiento de las consultas. Como conclusión, se mencionó que la implementación de estrategias basadas en la fragmentación trae beneficios tanto en términos de almacenamiento como en términos del procesamiento de las consultas. Ademas se presentaron buenas prácticas que se infieren del análisis realizado, resaltando que se preste especial atención a una excesiva fragmentación, ya que los sistemas presentan un declive en términos de rendimiento.

En los sistemas de bases de datos distribuidos, los costos de comunicación y el tiempo de

respuesta han sido desafíos abiertos durante mucho tiempo. Sin embargo, cuando los DDBS (*Distributed Dabase System*, Sistemas de Bases de Datos Distribuidas) se diseñan cuidadosamente, se logrará la reducción deseada en los costos de comunicación. Amer (2020) introdujo un enfoque heurístico de *k-means* para la fragmentación vertical y la asignación. Este enfoque se centró principalmente en el diseño de DDBS en la etapa inicial. Se llevó a cabo un estudio experimental breve pero efectivo, tanto en conjuntos de datos creados artificialmente como reales, para demostrar la optimización del enfoque propuesto frente a sus homólogos. Los resultados obtenidos sustentaron que el trabajo mostrado por el autor superó a diferentes métodos en la etapa de experimentación.

2.4.3. Otros tipos de fragmentación híbrida

La base de datos Teradata ofrece una solución híbrida única de almacenamiento basado en el concepto fila-columna que combina de manera transparente los esquemas de fragmentación horizontal y vertical. La característica clave de esta solución híbrida es que la capa de almacenamiento del sistema de archivos subyacente almacene y maneje todas las filas, columnas o particiones combinadas de la misma manera internamente. Al-Kateb et al. (2016) presentaron las características principales de este enfoque y explicaron detalladamente un nuevo método de implementación del almacenamiento fila-columna de Teradata. Posteriormente se presentó un estudio de rendimiento que demostró cómo las diferentes opciones de fragmentación afectan el rendimiento de las consultas y se proponen distintas técnicas de optimización de consultas aplicables específicamente a tablas fragmentadas. El propósito principal de los experimentos llevados a cabo fue identificar el impacto de diferentes opciones de columnas. En este trabajo se presentó una solución híbrida única de almacenamiento fila-columna implementada en la base de datos Teradata. Como trabajo a futuro mencionaron que se examinarán otros atributos de rendimiento de diferentes combinaciones de tipos de fragmentación.

Se han desarrollado H2TAP (*Heterogeneous Hybrid Transactional Analytical Processing*, Sistemas de Bases de Datos de Procesamiento Analítico Transaccional Híbrido Heterogéneo) para que coincidan con los requisitos para el análisis de baja latencia de datos operativos en tiempo real. Debido a los desafíos técnicos, estos sistemas son difíciles de diseñar, no triviales y complejos de administrar. Pinnecke et al. (2020) propusieron diferentes soluciones

para muchos de esos desafíos de forma aislada: aún no se desarrolla un motor unificado que permita optimizar el rendimiento al combinar estas soluciones. Los autores sugirieron una estructura de datos altamente flexible y adaptable llamada GRIDTABLE para organizar físicamente registros escasos pero estructurados en el contexto de H2TAP. Se concluyó el trabajo exhibiendo diferentes líneas de investigación abiertas y desafíos que deben abordarse para aprovechar la flexibilidad de la solución mostrada.

2.4.4. Otros tipos de fragmentación

Actualmente el formato XML se ha convertido en un estándar para representar datos complejos de negocios, sin embargo, las fuentes de datos XML tienen especificidades que serían complejas de manejar en un entorno relacional. Por esta razón los sistemas de bases de datos basados en XML nativo sufren de un desempeño limitado en términos de administración de volúmenes de datos y el tiempo de procesamiento de las consultas. Los algoritmos de fragmentación clásica no son adecuados para controlar el número de fragmentos originados. Cuzzocrea, Darmont & Mahboubi (2009) introdujeron el uso de un algoritmo basado en el agrupamiento *K-means* para un efectivo y eficiente soporte de la fragmentación de grandes almacenes de datos XML, y al mismo tiempo, un control y determinación del número de fragmentos originados mediante la configuración adecuada del parámetro *K*. Para validar la efectividad y la eficiencia se comparó la estrategia de fragmentación con dos adaptaciones significativas de los dos métodos de fragmentación más comunes para almacenar datos relacionales, las técnicas de fragmentación PC (*Predicate Construtions,* Construcción de Predicados) y AB (*Affinity-based,* basado en afinidad). Los resultados experimentales mostraron que el enfoque propuesto supera ambas técnicas de comparación bajo varias perspectivas del análisis experimental.

Los sistemas de almacenamiento transaccional distribuido hoy en día requieren niveles de aislamiento crecientes, rendimiento escalable, tolerancia a fallas y un modelo de programación simple para integrarse fácilmente con aplicaciones transaccionales. El reciente crecimiento de las infraestructuras a gran escala con docenas o cientos de nodos necesita soporte transaccional listo para escalar con ellos. El trabajo presentado por Turcu et al. (2016) determinó esquemas óptimos de fragmentación, los cuales ayudan en gran medida al diseño de esquemas cuando se trata de cantidades de datos no triviales. Como conclusión se

mencionó que se desempeñó un análisis estático de byte-code para determinar clases transaccionales que pueden ejecutarse utilizando el modelo de transacciones independientes. Debido al enfoque DMT (*Distributed Transactional Memory*, Memoria Transaccional Distribuida) utilizado, los autores adoptaron una técnica de aprendizaje automático para enrutar las transacciones a las particiones apropiadas. Los esquemas de fragmentación se obtuvieron eligiendo la primitiva transaccional correcta y enrutando las transacciones de manera adecuada. Para validar el desarrollo se utilizaron cinco *benchmarks* y en la mayoría de los casos se observaron mejoras tanto en la proporción de transacciones distribuidas como en el rendimiento transaccional.

Khan et al. (2018) presentaron un marco de trabajo de deduplicación de *cluster* robusto, tolerante a fallas y escalable que fue capaz de eliminar copias duplicadas en todo el *cluster*. Se diseñó una fragmentación de metadatos de deduplicación distribuida que garantizó la escalabilidad del rendimiento y la preservación de las restricciones de diseño de los sistemas de almacenamiento compartidos. La colocación de fragmentos y metadatos de deduplicación se realizó en todo el *cluster* en función de la huella dactilar del contenido de los fragmentos. Para garantizar la consistencia transaccional y la identificación de basura, se empleó un mecanismo de consistencia asincrónica basado en banderas. La evaluación mostró un gran ahorro de espacio en disco con una degradación mínima del rendimiento, así como una gran robustez en caso de falla repentina del servidor. El enfoque se implementó en Ceph, un *cluster* con arquitectura *shared-nothing*. Como conclusión los autores mencionaron que se propuso el diseño e implementación de un enfoque de fragmentación de metadatos de deduplicación distribuida que utiliza el *hash* de contenido de fragmentos para evitar problemas de transmisión y reubicación de objetos dinámicos.

Wycislik (2015) presentó una implementación de fragmentación de datos en bases de datos Oracle dedicadas a LOB (*Large Objects*, Objetos Grandes). Estas estructuras de LOB encapsulan cualquiera de los datos binarios, incluidos los documentos multimedia o electrónicos en varios formatos. La solución, a diferencia de la mayoría de las propuestas de los autores, fue lo suficientemente amplia como para permitir definir una API (*Application Programming Interface*, Interfaz de Programación de Aplicaciones), un esquema de datos y una implementación que no necesita cambiar independientemente de los escenarios de uso.

Sin embargo, es lo suficientemente flexible como para que una instancia del subsistema propuesto permita la prestación de servicios de fragmentación para muchos otros sistemas. Como trabajo a futuro se menciona que se realizará una selección apropiada de los parámetros físicos del espacio de las tablas teniendo en cuenta los detalles de los archivos LOB que están almacenados.

La Tabla 2.1 muestra la clasificación de los artículos por tipo de fragmentación, si es dinámica, además se especifica si el método de fragmentación toma en cuenta datos multimedia y si considera consultas basadas en contenido.

Tabla 2.1. Clasificación de artículos del estado del arte[A]

1	2	3	4	5	6	7	8	9	10	11
(Pérez et al., 2000)	Vertical	El algoritmo para el modelo DFAR se escribió en Borland C y se ejecutó en una PC con un procesador Pentium de 120 MHz y el sistema operativo Windows´95.	Sí	No	Sí	Dinámica	Distribuida	No se menciona	No	Los experimentos revelaron que cuando la capacidad del sitio se usa casi a su límite, el intercambio de posiciones de atributos podría bloquearse, evitando que el algoritmo de umbral de aceptación llegue a su límite y pueda converger. El trabajo mostró que la inclusión de un sitio artificial y la adición de un término de penalización solucionan este problema.
(Fung et al., 2002)	Vertical	Se fragmentó una estructura de clases relacionadas con los empleados, las clases son “Emp”, “Dept” y “Proj”, con 1000, 200 y 1000 instancias, respectivamente.	Sí	No	Sí	Estática	Orientada a objetos	NeoAccess System	No	Se desarrolló un modelo de costos analítico mejorado para calcular el número de accesos a disco requeridos por el procesamiento de consultas. El modelo de costos se usó para observar la efectividad de la fragmentación vertical bajo diferentes parámetros. Se demostró que mediante el modelo de costos propuesto se redujo considerablemente el número de accesos a disco.
(Fung, Leung & Li, 2003)	Vertical	Se fragmentó un curso de videos *e-learning* con tres tablas, *COURSE*,	Sí	Sí	Sí	Estática	Multimedia	Se desarrolló un SGBD especial para video *e-*	No	Se desarrolló un sistema gestor de base de datos de video *e-learning*, se introdujo la recuperación eficiente de video *e-learning* multimedia de

A 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		TOPIC y *SECTION*.						*learning* basado en 4DIS.		Internet y se desarrolló un modelo de costos para la fragmentación vertical de clases. Se demostró la efectividad de la técnica de fragmentación en el contexto del marco de trabajo de modelado 4DIS.
[B](Fung, Karlapalem y Li, 2003)	Vertical	Se fragmentó una base de datos orientada a objetos que contenía datos relacionados con empleados.	Sí	Sí	Sí	Estática	Orientada a objetos	NeoAccess System	No	Se desarrollaron dos algoritmos para aplicar los esquemas de fragmentación vertical derivada óptima/casi-óptima. El primer algoritmo es útil para analizar la efectividad de otros algoritmos, pero no es práctico, ya que es altamente costoso. El segundo algoritmo está basado en afinidad y obtiene un esquema de fragmentación inicial, después aplica la heurística *hill-climbing* para mejorar la solución utilizando un modelo de costos.
(Rodríguez y Li, 2011a)	Vertical	Se fragmentó como ejemplo una base de datos multimedia con una relación llamada equipo.	Sí	Sí	Sí	Estática	Multimedia	No se menciona el gestor que se utilizó, ni alguna tecnología específica.	No	Se desarrolló un algoritmo de fragmentación vertical para bases de datos multimedia (MAVP) basado en el algoritmo AVP (*Adaptable Vertical Partitioning*, Fragmentación Vertical Adaptable) (Son & Kim, 2004) y se desarrolló un modelo de costos. La evaluación experimental mostró que superó a AVP.

B 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
(Gu et al., 2012)	Vertical	El enfoque propuesto se evaluó mediante el *benchmark* TPC-H y un diseño de 20 atributos con más de 20 GB de tamaño. Las pruebas se llevaron a cabo sobre 16 nodos donde uno de ellos fungió como maestro.	Sí	Sí	Sí	Estática	*Big Data*	HDFS	No	Los resultados generados por el modelo de costos están estrechamente relacionados con el tiempo de ejecución de las consultas en las fases del mapeo, ya que su tendencia es consistente, lo que indica la efectividad del modelo de costos propuesto. CHAC redujo el tiempo de ejecución de las consultas 9.8% en un contexto no traslapado y 19.5% en un contexto traslapado.
[C](Li & Gruenwald, 2013)	Vertical	Se realizaron las pruebas bajo el *benchmark* TPC-H y se ejecutaron las consultas 10,000 veces, de las cuales se seleccionó el 60% aleatoriamente para utilizar posteriormente Auto-Clust.	No	Sí	Sí	Dinámica	Distribuida en línea	Oracle 11g	No	Como resultado se mostró en porcentajes la mejora en el desempeño de cada tabla sobre el *benchmark* TPC-H. Se mencionó también que se obtuvo un enfoque que mejora el desempeño de las bases de datos y toma decisiones para refragmentar automáticamente, minimizando la intervención del administrador de bases de datos.
(Rodriguez et al., 2013)	Vertical	Se realizó la implementación sobre la base de datos PostgreSQL y se utilizó el *benchmark* TPC-H.	Sí	Sí	Sí	Dinámica	Relacional	PostgreSQL	No	Se mostró como resultado la mejora en el desempeño que DYVEP posee al aplicarse. Además se plantearon todas las ventajas del enfoque propuesto.

C 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
(Alagiannis, Idreos & Ailamaki, 2014)	Vertical	Para la evaluación del enfoque se utilizó el *benchmark* SDSS. Se utilizó AutoPart para llevar a cabo la fragmentación Vertical.	Sí	No	Sí	Dinámica	Big Data	H^2O	No	Utilizando tanto pruebas comparativas sintéticas como experimentos con datos reales, se demostró que H2O se adapta eficientemente a medida que cambian las cargas de trabajo y se mantiene cerca del rendimiento óptimo sin requerir ningún conocimiento de la carga de trabajo.
[D](Li & Gruenwald, 2014)	Vertical	Se evaluó el desempeño de SMOPD-C con el *cluster* computacional OSCER BOOMER de la universidad de Oklahoma. Se utilizó el *benchmark* TPC-H en Oracle.	Sí	Sí	Sí	Dinámica	*Cluster*	Oracle	No	Los resultados del experimento mostraron que SMOPD-C es capaz de realizar una fragmentación dinámica con alta precisión y de obtener un menor costo en la ejecución de las consultas frente a otros enfoques.
(Zhao, Cheng & Rusu, 2015)	Vertical	Se utilizó el *benchmark* SDSS para evaluar el sistema con datos reales y se seleccionaron las 100 consultas más populares de la tabla *photoPrimary*.	Sí	Sí	Sí	Estática	Big Data	No se menciona	No	Los resultados confirman el rendimiento superior de la heurística propuesta sobre los algoritmos de fragmentación vertical relacionados y la precisión de la formulación al capturar los detalles de ejecución de un operador real.

D 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
(Rodríguez-Mazahua et al., 2017)	Vertical	Se fragmentó una base de datos multimedia de una compañía de venta de maquinaria con una relación llamada equipo.	Sí	Sí	Sí	Dinámica	Multimedia	PostgreSQL, pgAdmin, pgAgent, disparadores.	No	La principal ventaja de DYMOND lograda usando reglas activas fue la mejora del desempeño general de las bases de datos multimedia. DYMOND reduce las actividades que el DBA debe desarrollar y además, provee una eficiente recuperación de objetos multimedia en todo momento.
(Campero Durand et al., 2019)	Vertical	Se utilizó para su evaluación el *benchmark* TPC-H. Los algoritmos se implementaron en Java. La arquitectura se diseñó en el *framework* Google Dopamine	No	Sí	Sí	Estática	Relacional	No se menciona	No	Se mostró como resultado que se descubrió que la solución propuesta supera (en tiempo de optimización) al algoritmo de fuerza bruta, y que esta es competitiva con algoritmos de vanguardia, ya que permanece en el mismo orden de magnitud y se vuelve más competitiva cuando el número de atributos aumenta.
[E](Costa, Costa & Santos, 2019)	Vertical	Se presentó un análisis de diferentes enfoques para la fragmentación de los *Big Data Warehouse* y se mostró una conclusión de las pruebas realizadas, mostrando buenas prácticas que fungen como guía para	No	No	No	Estática	*Big Data* Werehouse	No se menciona.	No	Como resultados se obtuvieron recomendaciones basadas en el análisis presentado. Las buenas prácticas mencionadas tuvieron cinco enfoques: general, para estudios posteriores, para la implementación de técnicas de fragmentación, para la implementación de técnicas de *bucketing* y para el desempeño.

E1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		todos los interesados en utilizar alguna técnica de fragmentación.								
(Sharify et al., 2019)	Vertical	El sistema se implementó en C++. Para su evaluación se utilizó el *benchmark* NoBench el cual se usó con una carga de trabajo de 1000 consultas. Se comparó la capa DVP con capas basadas en filas y columnas utilizadas en muchas bases de datos comerciales, Argo1, Argo3 y Hyrise.	Sí	Sí	Sí	Dinámica	Relacional	No se menciona	No	La evaluación llevada a cabo detalló la utilización de cargas de trabajo adaptadas mediante NoBench y muestra que DVP superó el soporte relacional no optimizado para JSON en un 40% y frente a TLB en un 35%. La adaptación de la fragmentación basada en el cambio dinámico de la carga de trabajo dio como resultado un tiempo de ejecución de consultas 10% mejor.
[F](Amer, 2020)	Vertical	El conjunto de datos ocupado para realizar los experimentos fue una "*Car*", generado artificialmente. Este conjunto de datos	Sí	Sí	Sí	Estática	Distribuida	No se menciona	No	Los resultados obtenidos sustentaron que el trabajo mostrado por el autor superó a diferentes métodos en la etapa de experimentación.

F 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		contuvo 6 atributos y 400 tuplas.								
[G](Schroeder, Penteado & Hara, 2020)	Vertical	Se comparó ClusterRDF con los dos enfoques más relacionados utilizando el *benchmark* SPARQL de Berlín (BSBM).	Sí	Sí	Sí	Dinámica	Distribuida	Datos almacenados en RDF	No	Los resultados revelaron una mejora en el rendimiento del 27% al 86% en la recuperación de datos frente a los enfoques utilizados como referencia en la etapa de evaluación.
(Vazquez, 2000)	Horizontal	Los experimentos se llevaron a cabo en un software llamado *Parallel Server of Relational and Complex Architecture* (PSRCA), el cual es un sistema gestor de bases de datos relacional que trabaja con fragmentos horizontales	No	Sí	No	Dinámica	Relacional	PSRCA	No	Los resultados mostraron que el método de fragmentación virtual dinámica mejoró substancialmente los tiempos de respuesta de las pruebas realizadas a una base de datos de estudiantes.
(Pinto & Torres, 2002)	Horizontal	Se implementó la técnica de fragmentación horizontal en la base de datos de la	No	Sí	Sí	Dinámica	Distribuida	No se menciona	No	Como resultado se mencionó que se obtuvo un algoritmo basado en dos técnicas: Búsqueda esclavo-maestro, para proveer rápido acceso a las consultas de los usuarios, y

G 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		dependencia de Comisión Federal de Electricidad, la cual se menciona que cuenta con más de 50 atributos en la tabla llamada Usuarios. Se probó el algoritmo bajo 16 computadoras.								Replicación esclavo-maestro, para aumentar la disponibilidad. Se observó un excelente comportamiento en el desempeño de la base de datos bajo el enfoque propuesto.
[H](Saad et al., 2006)	Horizontal	Se llevaron a cabo experimentos con relaciones que contenían imágenes multimedia de vehículos como autos o camiones.	Sí	No	Sí	Estática	Multimedia	Oracle 9i	Sí	Se propuso un enfoque de fragmentación horizontal para bases de datos multimedia. El desarrollo de los nuevos operadores multimedia permitieron la adaptación de los procedimientos de fragmentación existentes, para que de esta manera fragmentaran datos multimedia
(Ma, Schewe & Wang, 2006)	Horizontal	La evaluación se llevó a cabo mediante un esquema de bases de datos el cual fue poblado por el *benchmark OO7*, posteriormente se consideraron cuatro sitios y 30 consultas,	Sí	Sí	Sí	Estática	De valores complejos, los cuales incluyen elementos multimedia	No se menciona.	No	Se presentó un modelo de costos detallado y un conjunto de heurísticas para la fragmentación y la asignación. Los experimentos validaron el enfoque presentado y mostraron que se logró minimizar el costo de trasporte mediante el predicado más frecuente en las consultas.

H 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		de las cuales 20% de ellas se utilizaron frecuentemente por las transacciones más importantes.								
[l](Getahun et al., 2007a)	Horizontal	La implementación consistió en variar el número de predicados y el número de conceptos para obtener el tiempo de ejecución usando la base semántica de predicados.	Sí	No	No	Estática	Multimedia	Oracle 9i	Sí	Mediante este trabajo se provee un nuevo enfoque para usarse con las técnicas de fragmentación existentes. Se descubrieron implicaciones de valor siguiendo valores orientados al dominio o genéricos de bases de conocimiento tales como WordNet.
(Getahun et al., 2007b)	Horizontal	Se fragmentó una base de conocimiento de 100 nodos extraída de WordNet y se varió el número de predicados, el valor de las cardinalidades y el tamaño de los datos multimedia para obtener el	Sí	No	No	Estática	Multimedia	Oracle 9i	Sí	En este estudio se desarrolló la fragmentación horizontal primaria para bases de datos multimedia. Se propusieron un conjunto de algoritmos para identificar la implicación entre los predicados semánticos, basados en la implicación del operador y del valor. Los resultados exhibieron que el método desarrollado tiene una complejidad polinomial.

l 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		tiempo de respuesta de cada ejecución.								
[J](Ma, Schewe & Wang, 2007)	Horizontal	Los experimentos se llevaron a cabo bajo cuatro sitios y se utilizaron 30 consultas, de las cuales el 20% son las más usadas. Las 30 consultas se obtuvieron con el *benchmark* OO7. Se utilizaron dos casos de estudio, uno aplicando el algoritmo propuesto y otro utilizando un enfoque tradicional.	Sí	Sí	Sí	Estática	Base de datos de valores complejos	No se menciona.	No	Los resultados demostraron que el enfoque heurístico presentado para la fragmentación horizontal derivada mejoró el rendimiento del sistema frente a otros enfoques tradicionales de fragmentación.
(Hauglid, Ryeng & Nørvåg, 2010)	Horizontal	Se realizaron las evaluaciones sobre dos escenarios bajo un simulador y una en un entorno real, la cual involucró un sistema de bases de datos distribuido, DASCOSA-DB.	Sí	Sí	Sí	Dinámica	Distribuida en la nube	DASCOSA-DB	No	Los resultados de las simulaciones revelaron que, para las cargas de trabajo típicas, el enfoque propuesto de fragmentación dinámica reduce significativamente los costos de comunicación.

J 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
[K](Bellatreche et al., 2011)	Horizontal	Para realizar los experimentos se utilizaron 22 consultas, de las cuales se extrajeron 50 predicados para aplicar dos enfoques de fragmentación, *Joint* y *Sequential*. El entorno en el cual se aplicaron los dos enfoques fue el *benchmark* SSB.	No	No	Sí	Estática	Data Warehouse	Teradata	No	Como resultados de la aplicación de los dos enfoques de fragmentación se obtuvo una mejora en el rendimiento del sistema de un 38% del método *Joint* respecto al método *Sequential*.
(Abdalla & Amer, 2012)	Horizontal	Se presentó la implementación del modelo propuesto mediante una tabla que contenía información de los empleados de una empresa, la cual pertenecía a un sistema de bases de datos distribuidas. La fragmentación y asignación se realizó en 4 sitios diferentes.	No	Sí	Sí	Dinámica	Distribuida en la nube	No se menciona	No	Como resultados se obtuvo un nuevo enfoque de fragmentación que toma en cuenta el costo de transporte y el costo de acceso a datos remotos. Se mencionó que se mejoró el desempeño de las bases de datos distribuidas aplicando el desarrollo propuesto.

K 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
(Liroz-Gistau et al., 2012)	Horizontal	Para realizar la evaluación se utilizó el catálogo SDSS DR8. La base de datos tuvo 350 millones de tuplas que ocupan 1.2 TB. La carga de trabajo consistió en un total de 27000 consultas.	Sí	Sí	No	Dinámica	Relacional	SDSS SkyServer SQL	No	Los resultados del experimento mostraron que la estrategia de fragmentación dinámica propuesta es capaz de lidiar con los datos de la aplicación descrita, pero que los autores afirman que la estrategia no está limitada a esta aplicación.
[L](Bellatreche et al., 2013)	Horizontal	El enfoque propuesto se evaluó utilizando el *benchmark* APB1 y un almacén de datos con un esquema en estrella, el cual se pobló con más de 24 millones de tuplas y tablas de 4 dimensiones.	No	No	Sí	Dinámica	Data Warehouse	No se menciona	No	Como resultados se mencionó que ISGA presentó una mejor optimización en el desempeño de las consultas frente a los otros dos algoritmos propuestos, y además reduce los costos de mantenimiento.
(Derrar, Nacer & Boussaid, 2013)	Horizontal	Se realizaron estudios experimentales utilizando Oracle 10G con el *benchmark* APB1 para verificar la adaptabilidad del	Sí	No	Sí	Dinámica	Data Warehouse	Oracle 10 G	No	Se presentó un enfoque que consiste no solo en analizar las frecuencias de acceso a datos para diseñar un esquema de fragmentación óptimo, sino también en hacer que ese diseño sea dinámico y se adapte a la carga de trabajo cambiante. Los resultados obtenidos fueron alentadores tanto en

L 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		enfoque propuesto.								términos de espacio de memoria como de tiempo de ejecución de consultas.
(Fasolin et al., 2013)	Horizontal	Los experimentos se realizaron en CoPhIR, una base de datos multimedia que contenía 106 millones de imágenes obtenidas desde la red social FLICKR.	No	No	No	Estática	Multimedia	Oracle 11g, FMI-SiR O	Sí	Los experimentos mostraron que la propuesta mejora drásticamente el tiempo de ejecución de las consultas y revelaron que no es viable ejecutar predicados basados en la similitud sobre toda la base de datos CoPhIR incluso usando el índice métrico *Slim-tree* para aumentar la velocidad de ejecución de los predicados.
[M] (Lim, 2013)	Horizontal	Se evaluó el enfoque usando el *benchmark* TPC-H generado con 60 167 tuplas y 6 001 215 tuplas.	Sí	Sí	Sí	Estática	Distribuida en la nube	IBM DB2	No	Se propusieron diferentes métodos para escalar bases de datos paralelas de diferente cantidad de nodos. El prototipo implementado en la etapa de experimentación demostró su efectividad y eficiencia.
(Liroz-Gistau et al., 2013)	Horizontal	Las pruebas se realizaron utilizando el catálogo de Sloan Digital Sky y su archivo *log*. La evaluación se llevó a cabo en un procesador Intel de 3 GHz en Ubuntu 11.10 con 4 GB de RAM.	Sí	Sí	No	Dinámica	Relacional	SDSS SkyServer SQL	No	Los resultados revelaron que en el caso de conjuntos de datos en los que existe una alta correlación entre los nuevos elementos de datos, el algoritmo DynPartGroup mantiene un comportamiento muy bueno.

M 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
(Herrmann, Voigt & Lehner, 2014)	Horizontal	Se realizó la evaluación con el *benchmark* TPC-H y DBpedia. Se implementó Cinderella en PostgreSQL.	Sí	Sí	Sí	Estática	Distribuida	PostgreSQL	No	Cinderella crea particiones de bajo peso y separa las entidades de forma autónoma en particiones de tamaño fijo. Los resultados mostraron que Cinderella fue capaz de disminuir significativamente el tiempo de ejecución de consultas selectivas en comparación con una tabla universal no fragmentada.
(Kumar & Gupta, 2014)	Horizontal	Los experimentos se llevaron a cabo con tablas ficticias y se comparó el enfoque con otros algoritmos de fragmentación.	No	No	No	Dinámica	Distribuida	No se menciona	No	Como resultados se mencionó que los cálculos realizados sobre la base de datos hipotéticos respaldan que el algoritmo extendido propuesto TTVDCA es mejor que otros algoritmos desarrollados previamente en esta categoría y mejoró el rendimiento general del sistema.
[N](Baron & Iacob, 2014)	Horizontal	Para presentar el sistema se propuso un ejemplo en el cual se aplica el enfoque desarrollado, sin embargo, no se menciona un conjunto de datos específico para la evaluación del sistema.	No	Sí	Sí	Dinámica	Relacional	No se menciona	No	Como resultado, los autores mencionaron que no sólo innovaron, sino que también mejoraron el desempeño con el enfoque propuesto el cual es configurable y fácil de administrar.

N 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
(Rodríguez-Mazahua et al., 2014)	Horizontal	Se fragmentó un *benchmark* de una compañía de venta de maquinaria. Se comparó el esquema producido por MHPA (*Multimedia Horizontal Partitioning Algorithm*, Algoritmo de Fragmentación Horizontal Multimedia) y el enfoque de fragmentación basado en afinidad.	Sí	Sí	Sí	Estática	Multimedia	No se menciona el SGBD que se utilizó ni alguna tecnología especifica.	No	Se propuso un algoritmo de fragmentación horizontal para bases de datos multimedia basado en el agrupamiento aglomerativo para eliminar el problema de afinidad. La evaluación mostró que el algoritmo desarrollado superó al basado en afinidad en la mayoría de los casos.
[O](Fetai, Murezzan & Schuldt, 2015)	Horizontal	Se utilizó el *benchmark* TPC-C con más de 10 000 tuplas y se utilizó Amazon EC2 para realizar las transacciones en línea.	No	Sí	Sí	Dinámica	Distribuida en la nube	No se menciona	No	La evaluación mostró la ganancia general en el rendimiento en comparación con los sistemas en los que se producen transacciones distribuidas. Los resultados de la evaluación mostraron que Cumulus superó los enfoques de fragmentación estática y también los enfoques que no admiten la re-partición.

O 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
(Sauer & Wei Hao, 2015)	Horizontal	Los estudios experimentales se llevaron a cabo en la base de datos MongoDB simulando tráfico en la red de 100, 110 y 150 usuarios. Se compararon los tiempos de respuesta de diferentes algoritmos de fragmentación como *round robin*, *k-means* y *minimum spanning tree*.	No	Sí	No	Estática	NoSQL	MongoDB	No	Como resultados se mencionó que el trabajo propuesto superó el rendimiento de una base de datos sola en la nube o usando la fragmentación *round robin*.
[P](Abdel Raouf, Badr & Tolba, 2016)	Horizontal	Se realizaron los experimentos sobre una tabla que contiene datos sobre cuentas de usuarios de bancos. Se utilizaron 3 sitios para aplicar la distribución de fragmentos.	Sí	Sí	Sí	Dinámica	Distribuida en la nube	No se menciona	No	Se presentó un enfoque eficiente para la fragmentación, asignación y replicación basado en el historial de acceso. El objetivo de la técnica presentada es maximizar los accesos locales.

P 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
(Elghamrawy, 2016)	Horizontal	Se evaluó el algoritmo LBRH (*Load Balancing based Rendezvous Hashing*, *Hashing* de encuentro basado en el balanceo de carga) y el tiempo de respuesta de Cassandra con ARHPM. El *benchmark* utilizado fue YCSB.	Sí	No	No	Estática	NoSQL	Cassandra	No	Los resultados mostraron que la carga de trabajo con ARHPM es balanceada más uniformemente que utilizando los otros dos estándares de Cassandra.
[Q](Serafini et al., 2016)	Horizontal	Los experimentos se llevaron a cabo utilizando el *benchmark* TPC-C y un *cluster* de 10 nodos.	Sí	Sí	Sí	Dinámica	OLTP	H-Store	No	Se presentó Clay, un algoritmo de elasticidad DBMS en línea que no hace suposiciones sobre el esquema de la base de datos, y simultáneamente equilibra la carga y minimiza las transacciones distribuidas. Los experimentos revelaron que Clay superó sustancialmente las técnicas de elasticidad de la base de datos de última generación.
(Wu, Chen & Jiang, 2016)	Horizontal	Se construyó la fragmentación de base de datos de la cultura Hakka, la indexación, la	No	Sí	No	Estática	Multimedia	MongoDB	No	La implementación mejoró la flexibilidad y la eficiencia del manejo de las múltiples fuentes de datos heterogéneos y no estructurados.

Q 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		indexación espacial y consultas. Mediante lo construido se propuso un prototipo de manejo de datos de la cultura Hakka.								
(Islam Khan, 2016)	Horizontal	Se aplicó la fragmentación en un esquema propuesto relacionado con la administración de clientes.	No	Sí	Sí	Estática	Relacional	Oracle 10 g	No	Mediante los experimentos se mostró que la técnica propuesta logró una buena tasa de éxito. Por esta razón, el rendimiento de un sistema gestor de bases de datos distribuido mejoró significativamente al evitar el acceso remoto frecuente y la alta transferencia de datos entre los sitios.
[R](Elghamrawy & Hassanien, 2017)	Horizontal	Se modificó el SGBD Cassandra incrustando MR-RHVH. Se utilizó el *benchmark* YCSB para llevar a cabo las evaluaciones.	No	Sí	Sí	Estática	Relacional	Oracle 10 g	No	Los resultados mostraron que el marco de trabajo MR-RHVH fue altamente eficiente en términos de rendimiento y latencia cuando se compara con los sistemas recientes.
(Oonhawat & Nupairoj, 2017)	Horizontal	Se mencionó que los experimentos se llevaron a cabo	Sí	Sí	No	Estática	NoSQL	MongoDB	No	Como resultado se mostró que a diferencia de la fragmentación de otros autores ésta permitió que los

R 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		mediante 17 millones de tuplas obtenidas de la universidad Chulalongkom.								fragmentos reciban datos y sean fáciles de predecir. La sobrecarga de la creación de los fragmentos y la migración se minimizaron mediante la estrategia de la pre-asignación de la creación de los fragmentos.
(Zar Lwin & Naing, 2018)	Horizontal	La implementación se realizó sobre cuatro sitios en la nube totalmente conectados, en los cuales se ejecutaron 10,000 consultas. Se utilizó el enfoque propuesto para fragmentar la base de datos y migrar cada fragmento al mejor sitio.	Sí	No	Sí	Dinámica	Distribuida	No se menciona	No	El resultado del experimento mostró que el rendimiento de los cuatro sitios después de la migración es ligeramente mejor que el rendimiento de los cuatro sitios antes de la migración.
(Olma et al., 2019)	Horizontal	Slalom se construyó en C++. La evaluación comparó Slalom frente a DBMS-X y PostgreSQL.	Sí	No	Sí	Dinámica	Relacional	PostgreSQL, Slalom, DBMS-X	No	Los resultados revelaron que Slalom superó a los motores *in situ* de última generación y logró tiempos de respuesta de consulta comparables con SGBD's completamente indexados, ofreciendo tiempos de ejecución de consulta acumulados más bajos para cargas de trabajo de consultas con un tamaño creciente y patrones de acceso impredecibles.
[S](Castro-Medina	Horizontal	La implementación	Sí	Sí	Sí	Estática	Distribuida	MySQL	No	Las conclusiones mencionan que los

1	2	3	4	5	6	7	8	9	10	11
et al., 2020)		se llevó a cabo utilizando el *benchmark* TPC-E y los servicios Web de Amazon.								resultados mostraron mejoras en los tiempos de respuesta de las consultas. Bajo este nuevo enfoque, el número de operaciones remotas disminuye y aumentan las locales, reduciendo de esta manera los costos de trasporte.
(Chbeir y Laurent, 2010)	Híbrida	Se presentó como implementación un ejemplo de una base de datos multimedia la cual considera una tabla llamada "Albums" y contiene como atributos *nombre, nacimiento, lugar, genero, imagen, canción y clip*. Se aplica la fragmentación a dicha tabla.	Sí	No	No	Estática	Multimedia	No se menciona ningún SGBD ni tecnología especifica.	Sí	Se propuso un nuevo enfoque de fragmentación mixta para bases de datos multimedia, basado en la comparación y la equivalencia de consultas. Con este nuevo enfoque se fragmentan los datos multimedia eficientemente.
(Jindal & Dittrich, 2012)	Híbrida	Se utilizaron dos *benchmarks* para evaluar el sistema: TPC-H y SSB. Se utilizó BerkeleyDB para utilizar una carga de trabajo real.	Sí	Sí	Sí	Dinámica	OLTP, OLAP	BerkeleyDB	No	Las gráficas presentadas como resultados de la evaluación mostraron que AutoStore superó los enfoques sin fragmentación y con una fragmentación únicamente vertical bajo dos diferentes tablas de los conjuntos de datos mencionados.
[T](Wang et al.,	Híbrida	Los experimentos se	Sí	Sí	Sí	Dinámica	Relacional	MySQL	No	ADDS mostró 28.45% menos

S 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
2014)		realizaron con 8 nodos cada uno equipado con MySQL. El conjunto de datos se obtuvo mediante el *benchmark* TPC-C.								transacciones distribuidas frente a la estrategia *Hashing* y *Round-Robin*. Cada nodo, con las dos estrategias anteriores, necesita procesar más transacciones y ADDS reduce la carga de trabajo por nodo en un 10.42%.
(Chen et al., 2015)	Híbrida	La evaluación del enfoque se llevó a cabo en Amzon EC2 con 5 nodos utilizando PostgreSQL. Se creó un conjunto de datos de 500 000 tuplas.	Sí	No	Sí	Dinámica	Distribuida en la nube	PostgreSQL	No	Con base en los experimentos de ejecutar el prototipo propuesto en Amazon EC2, se obtuvo como resultado que *Hybrid* fue el mejor enfoque, ya que logró una gran mejora en el rendimiento en comparación con las dos soluciones básicas utilizadas en la mayoría de los sistemas existentes.
(Kulba & Somov, 2020)	Híbrida	La implementación del algoritmo se llevó a cabo en C++ dentro del entorno de desarrollo MS Visual Studio 2017.	No	No	Sí	Dinámica	Distribuida	No se menciona	No	No se mencionan las etapas de evaluación y resultados.
(Al-Kateb, Sinclair, Au & Ballinger, 2016)	Híbrida	Se desarrolló la implementación bajo *Teradata 6650 Enterprise Data Warehouse* (EDW). Se utilizó el *benchmark* TPC-H	No	No	Sí	Estática	Data Warehouse	Teradata	No	Los resultados mostraron que las tablas fragmentadas por columnas (con o sin fragmentación de filas) son 50% más pequeñas de tamaño en promedio. Además, se observa en los resultados de los experimentos llevados a cabo el impacto de diferentes opciones de

T 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		con un Terabyte de tamaño.								fragmentación.
[U](Rodríguez-Mazahua et al., 2016)	Híbrida	Se fragmentó una base de datos multimedia de una compañía de venta de maquinaria con una relación llamada equipo.	Sí	Sí	Sí	Estática	Multimedia	No se menciona el gestor que se utilizó, ni alguna tecnología específica.	No	La fragmentación híbrida desarrollada optimizó el costo de ejecución de consultas, ya que reduce los datos irrelevantes utilizados por las consultas. Se presentó el modelo de costos que toma en cuenta el costo de procesamiento general de consultas. La evaluación experimental mostró que, en la mayoría de los casos, el algoritmo propuesto superó al algoritmo de fragmentación horizontal y vertical.
(Mourão & Magalhães, 2017)	Híbrida	El método propuesto se evaluó mediante un conjunto de datos con 1 millón de descriptores desde dos tipos de características: GIST y SIFT.	No	No	No	Estática	Multimedia	No se menciona	Sí	El resultado de los experimentos mostró que procesar libros de código que eviten fragmentos grandes produce un esquema más balanceado y tiene un impacto positivo en el tiempo de respuesta del sistema.
(Durand et al., 2018)	Híbrida	Para la implementación se utilizó OpenAI Gym, un kit de herramientas de código abierto para	No	No	Sí	Estática	Distribuida en la nube	No se menciona	No	No se presentó la evaluación del sistema y debido a esto no se tienen resultados del enfoque propuesto.

U 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		el aprendizaje de refuerzo.								
[V](Schreiner et al. 2018)	Híbrida	No se mencionaron detalles de la implementación.	No	No	No	Estática	OLTP, NewSQL	No se menciona	No	Se obtuvo un enfoque para la fragmentación de datos considerando cargas de trabajo, cargas OLTP y soporte para el *streaming* de datos. El trabajo mostró que la técnica de fragmentación híbrida propuesta ofrece la optimización de los datos y el almacenamiento de datos con base en las cargas de trabajo.
(Vogt, Stiemer & Schuldt, 2018)	Híbrida	No se presentaron detalles de la implementación.	No	Sí	Sí	Dinámica	*Polystorage* distribuido	No se menciona	No	No se mostraron los resultados, ya que no se presentó una evaluación de la implementación.
(Schreiner et al., 2019)	Híbrida	Se consideró el *benchmark* OLTP-Bench. El trabajo se comparó con diferentes esquemas de fragmentación mediante VoltDB. Se utilizó OLTP-Bench para usar el *benchamrk* TPC-C.	Sí	Sí	No	Estática	OLTP	NewSQL	No	Lo resultados mostraron la eficiencia de Hybrid-VoltDB de dos maneras: en términos del desempeño de las transacciones distribuidas y el procesamiento de las operaciones generales de VoltDB nativo incluso sin fragmentación.
(Pinnecke et al., 2020)	Híbrida	Los experimentos se llevaron a cabo mediante las tablas CUSTOMER y	No	No	Sí	Estática	OLTP, OLAP	H^2TAP	No	Como resultado de los experimentos se observó que las cargas de trabajo mixtas afectan el rendimiento del sistema. Como conclusión se

V 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		LINEITEM del *benchmark* TPC-C								mencionan diferentes desafíos que presenta el enfoque propuesto.
[W](Sleit et al. 2007)	De objetos	Se presentó un ejemplo de la aplicación del algoritmo propuesto y se calcula el costo antes y después de la fragmentación y replicación demostrando de esta manera su efectividad reduciendo los costos.	Sí	Sí	Sí	Dinámica	Distribuida de objetos	No se menciona	No	Se mostró como resultado la principal ventaja de E-ADRW, la cual es reducir el tamaño del espacio ocupado de almacenamiento, ya que, comparado con el algoritmo óptimo y de umbral, determinados en la etapa de análisis, requiere menos espacio este enfoque.
(Hung & Huang, 2012)	De objetos	Para evaluar el algoritmo se utilizó el simulador OptorSim y se consideraron diferentes sitios en MMDB. OptorSim realiza de 100 a 5000 operaciones con 6 tipos de trabajo.	No	Sí	Sí	Dinámica	*Cluster*	MMDB	No	Se presentó como resultado que la simulación demuestra que DFAPR es adecuado para el *cluster* MMDB debido a que proporciona un mejor tiempo de respuesta y maximiza el procesamiento local.
[X](Torjmen, Pinel-	De documentos	La ejecución del	No	Sí	No	Estática	Multimedia	XML	Sí	Se propuso un nuevo enfoque que

W 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
Sauvagnat & Boughanem, 2008)		trabajo se realizó sobre una colección de Wikipedia en XML que contenía 660 mil documentos, 30 millones de elementos y más de 300 mil imágenes.								consiste en la representación de elementos multimedia por otros nodos contenedores de información relevante. Este enfoque estudia el uso de información textual y relaciones semánticas jerárquicas entre los elementos XML. Los experimentos mostraron la eficiencia del método propuesto.
(Cuzzocrea, Darmont & Mahboubi, 2009)	De documentos	Para realizar los experimentos presentados se utilizó el *benchmark* XWeb (*XML Data Warehouse Benchmark*, *Benchmark* de almacén de datos XML).	No	No	Sí	Estática	OLAP	XML	No	Se introdujo un enfoque para la fragmentación de almacenes de datos XML, el cual está basado en la minería de datos bajo el algoritmo de agrupamiento *K-means*. Los resultados obtenidos mostraron que el trabajo presentado superó el desempeño de los algoritmos utilizados para su comparación.
(Torjmen-Khemakhem, Pinel-Sauvagnat & Boughanem, 2013)	De documentos	La evaluación del trabajo presentado por los autores se llevó a cabo mediante INEX, con un conjunto que supera los 660,000 documentos.	Sí	Sí	No	Estática	Multimedia	XML	Sí	Como resultados se demostró que la estructura de los documentos ayuda a mejorar la efectividad de la recuperación de elementos multimedia y a evaluar la relevancia de los fragmentos. Además, se mencionó que usar el factor de distancia entre elementos multimedia tiene un alto impacto en el desempeño de la

X 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
										respuesta.
(Heni & Gargouri, 2015)	De documentos	Las pruebas se realizaron sobre la base de datos MongoDB, sin embargo, no se detalla qué conjuntos de datos se utilizaron.	No	No	No	Estática	NoSQL	MongoDB	No	Como resultado se mencionó que se obtuvo un enfoque para la protección de datos en *Big Data* para bases de datos NoSQL.
[Y](Santos & Masala, 2018)	De documentos	Los experimentos se llevaron a cabo en Amazon Web Services y Microsoft Azure comparando la ejecución del mismo algoritmo con diferentes tipos de archivos.	No	Sí	No	Estática	Multimedia	RPFNoSQL DB	No	Como resultado se obtuvo un nuevo método para combinar la fragmentación con patrones aleatorios en bases de datos NoSQL para facilitar la organización y el manejo de los datos. La investigación ha indicado que existe una compensación en el rendimiento cuando se utiliza una base de datos. Cualquier ligera diferencia en un entorno de *Big Data* es de gran importancia, sin embargo, este costo se compensa al organizar y administrar los datos con el enfoque presentado.
(Mourão & Magalhães, 2019)	De documentos	Los experimentos se llevaron a cabo en la nube de Microsoft Azure. El conjunto de datos utilizado fue *Billion Vectors,*	Sí	Sí	No	Estática	Multimedia	No menciona	No	Como resultados se mostraron diferentes aspectos en los que DISH presenta una mejora notable: fragmentación de índices multimedia, fundación estadística rigurosa, procesamiento de consultas paralelas

Y 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		el cual tiene más de 1 billón de vectores de características SIFT.								con baja latencia y resistencia a falla de nodos.
(Mettes et al. 2015)	De archivos de video	Los experimentos se llevaron a cabo en un conjunto de datos llamado THUMOS'14, el cual contiene 1010 videos de validación. Cada video tiene algunas diferencias semánticas en eventos con acciones relacionadas en intervalos de tiempo diferentes.	No	No	No	Estática	Multimedia	ImageNet	Sí	Mettes et al. mostraron la codificación de videos usando fragmentos. La evaluación mostró la efectividad de la codificación de detección de eventos, como también su naturaleza complementaria para una agregación global de conceptos semánticos.
[z](Wycislik, 2015)	De datos no estructurado	Todas las implementaciones requeridas se encapsularon en dos paquetes de código SQL, no se mencionó ninguna evaluación. Las implementaciones se desplegaron en un	Sí	Sí	No	Estática	Multimedia	Oracle	No	Como resultado se obtuvo un ejemplo de implementación de la fragmentación dedicada al almacenamiento y procesamiento de objetos LOB.

z 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		sistema de documentos clínicos.								
[AA](Turcu et al., 2016)	K-way graph partitioning	El desarrollo se implementó en el sistema DTM (*Distributed Transactional Memory*, Memoria Transaccional Distribuida) y las pruebas se realizaron bajo los *benchmark* TPC-C, TPC-W, AuctionMark, EPinions y ReTwis en un sistema distribuido de 20 computadoras físicas con el sistema gestor de bases de datos H-base.	Sí	No	Sí	Estática	OLTP	H-Store	No	La aportación final mencionada fue la metodología utilizada para la fragmentación de datos automática. Se desarrolló un sistema para fragmentar mediante el enfoque de aprendizaje automático, el cual mejoró el rendimiento 4.5 veces más comparado con otros desarrollos.
(Chernishev, 2017)	De columnas	No se presentó la implementación.	No	Sí	No	Dinámica	Relacional	No se menciona	No	Como resultados se mencionó que se presentaron diferentes opciones de diseño físico para crear el enfoque deseado, así como tres alternativas para crear una alerta y ejecutar la reorganización de la base de datos.
(Nguyen & Lee,	Interna	Se implementó el	Sí	Sí	No	Estática	NoSQL	MongoDB	No	Los resultados que se mostraron al

AA 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
2017)		enfoque de fragmentación sobre dos *benchmarks*, YCSB y Linkbench. Los experimentos se llevaron a cabo en servidores con WiredTiger como motor de almacenamiento. Se utilizaron 23 millones de documentos de 1 kB cada uno.								evaluar los enfoques, presentaron mejoras en el rendimiento de los dos *benchmarks*, YCSB y Linkbench. El trabajo que se presentó mejoró en un 44% el desempeño de YCSB y 43.73% Linkbench.
[BB](Khan, Lee at al., 2018)	De metadatos	Se implementó el enfoque propuesto en Cepth v10.2.3. Se usó el algoritmo SHA-1 para generar las huellas de los fragmentos. Se usó el *benchmark* FIO (*Flexible I/O Tester*, Probador de E/S Flexible), con 500 GB de carga de trabajo.	No	No	No	Estática	*Shared-nothing*	SQLite	Sí	La evaluación mostró que el enfoque propuesto soporta alta escalabilidad con mínima sobrecarga en el desempeño y una robusta tolerancia a fallas.
(Lu et al., 2018)	Basada en	Se evaluó el método	Sí	No	No	Dinámica	Multimedia	No se	Sí	El enfoque se comparó con otros dos

BB 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
	contenido	propuesto mediante tres conjuntos de datos ampliamente usados: Fashion-MNIST, SIFT y GloVe.						menciona		métodos y se demostró la eficiencia y exactitud del método propuesto.
[CC](Le, Kantere & Orazio, 2019)	De rejilla	El algoritmo se probó con un conjunto de datos DICOM con seis modalidades de imágenes diferentes. El conjunto de datos contenía dos terabytes, incluyendo 2.4 millones de imágenes de 20,080 estudios.	Sí	No	No	Estática	Multimedia	Oracle 11g	No	Los resultados mostraron que el algoritmo de mejor calidad, NSGA-III, tiene un tiempo de cómputo muy largo. Sin embargo NSGA-G es más pequeño que los otros algoritmos analizados.
(Santos, Ghita & Masala, 2019)	Fragmentación aleatoria de archivos	Se realizaron los experimentos en un conjunto de datos con 4 tipos de datos. Cada archivo contuvo 100 kB de tamaño y se almacenó en Cassandra. Cassandra se	No	Sí	No	Estática	NoSQL	Cassandra	No	Los resultados mostraron un mayor rendimiento en comparación con sus contrapartes, lo que implica la usabilidad del método propuesto en la computación en la nube, especialmente en escenarios con necesidades de alta velocidad y recursos limitados.

CC 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

1	2	3	4	5	6	7	8	9	10	11
		desplegó en las infraestructura de Microsoft Azure.								
(Sharma & Bawa, 2019)	Fragmentación de archivos	Se utilizó un conjunto de datos de los detalles arquitectónicos e imágenes de monumentos del patrimonio indio, videos del desempeño de rituales, archivos de audio de cantos y archivos escaneados [DD]de los libros santos.	No	Sí	No	Estática	Multimedia	MongoDB	No	Los resultados mostraron que el trabajo mejoró el tiempo de respuesta de las lecturas y escrituras de cada fragmento. El modelo CBDR da una solución en el campo de la tecnología y la cultura mediante el almacenamiento eficiente de datos culturales.

DD 1) Artículo, 2) Tipo de fragmentación, 3) Detalles de la implementación, 4) Completitud, 5) Facilidad de implementación, 6) Modelo de costo, 7) Dinámica o estática, 8) Tipo de base de datos, 9) Sistema gestor de base de datos (SGBD) y tecnologías, 10) Consultas basadas en contenido, 11) Resultados

Al concluir el análisis comparativo se obtuvo información valiosa sobre el proceso, diseño y herramientas para llevar a cabo el objetivo de este trabajo. Se observa que los trabajos Rodríguez-Mazahua et al. (2017), Fetai, Murezzan & Schuldt (2015) y Rodríguez-Mazahua et al. (2016) presentan modelos de costos fáciles de implementación. Rodríguez-Mazahua et al. (2017) y Fetai, Murezzan & Schuldt (2015) realizan una fragmentación dinámica vertical y horizontal, respectivamente. Rodríguez-Mazahua et al. (2016) presenta una fragmentación híbrida y destaca por contener un modelo de costos simple enfocado en bases de datos multimedia. Los tres trabajos mencionados presentan, para la fragmentación vertical, horizontal e híbrida, excelentes formas de realizar la fragmentación y son consideradas de manera especial en este trabajo, ya que destacan por diversas características.

Saad et al. (2006) es el trabajo más citado en la categoría “Fragmentación de bases de datos multimedia” (Rodríguez & Li, 2011; Zhao, Cheng & Rusu, 2015; Getahun et al., 2007; Tekli et al., 2007; Rodríguez-Mazahua et al., 2014; Chbeir & Laurent, 2010; Rodríguez-Mazahua et al., 2016), Jindal & Dittrich (2012) es el artículo con más citas en la categoría "Fragmentación dinámica" (Li & Gruenwald, 2013; Alagiannis, Idreos & Ailamaki, 2014; Li & Gruenwald, 2014; Zhao, Cheng & Rusu, 2015; Rodríguez-Mazahua et al., 2016; Sharify et al., 2019; Herrmann, Voigt & Lehner, 2014; Fetai, Murezzan & Schuldt, 2015; Serafini et al., 2016; Olma et al., 2019; Wang et al., 2014; Chen, Cao & Zhou, 2015; Durand et al., 2018; Vogt, Stiemer & Schuldt, 2018; Pinnecke et al., 2020; Chernishev, 2017), Karlapalem & Li (2003) es el trabajo más citado en la categoría "Otros tipos de fragmentación” (Fung, Karlapalem & Li, 2002; Rodriguez & X. Li, 2011; Gu et al., 2012; Zhao, Cheng & Rusu, 2015).

CAPÍTULO 3. METODOLOGÍA

Para llevar a cabo los objetivos de este trabajo, se proponen una serie de pasos que dividen el trabajo en etapas, las cuales están sujetas a la metodología de desarrollo elegida. En la Figura 3.1 se presenta la metodología general de este proyecto de investigación y se detalla en las siguientes secciones.

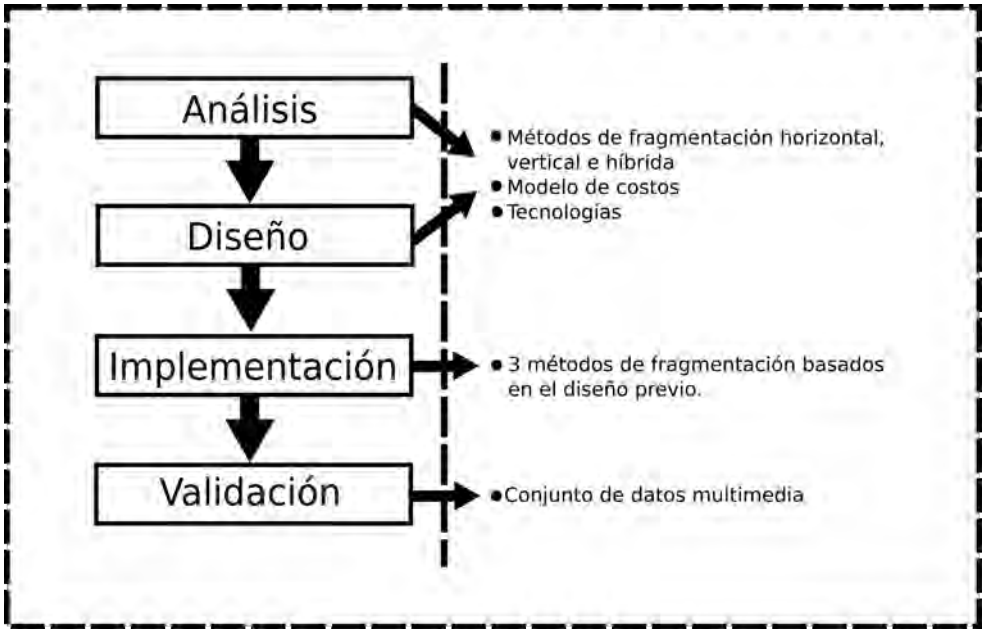


Figura 3.1. Metodología general del proyecto de investigación

3.1. Análisis

En esta etapa del trabajo se analizaron diferentes artículos científicos por medio de la búsqueda en las principales bibliotecas digitales como son ACM Digital Library, SpringerLink, ScienceDirect y IEEE Xplore, donde se encuentren métodos de fragmentación, modelos de costos y tecnologías empleadas para su desarrollo. La Figura 3.2 muestra la metodología de búsqueda de trabajos relacionados y su clasificación.

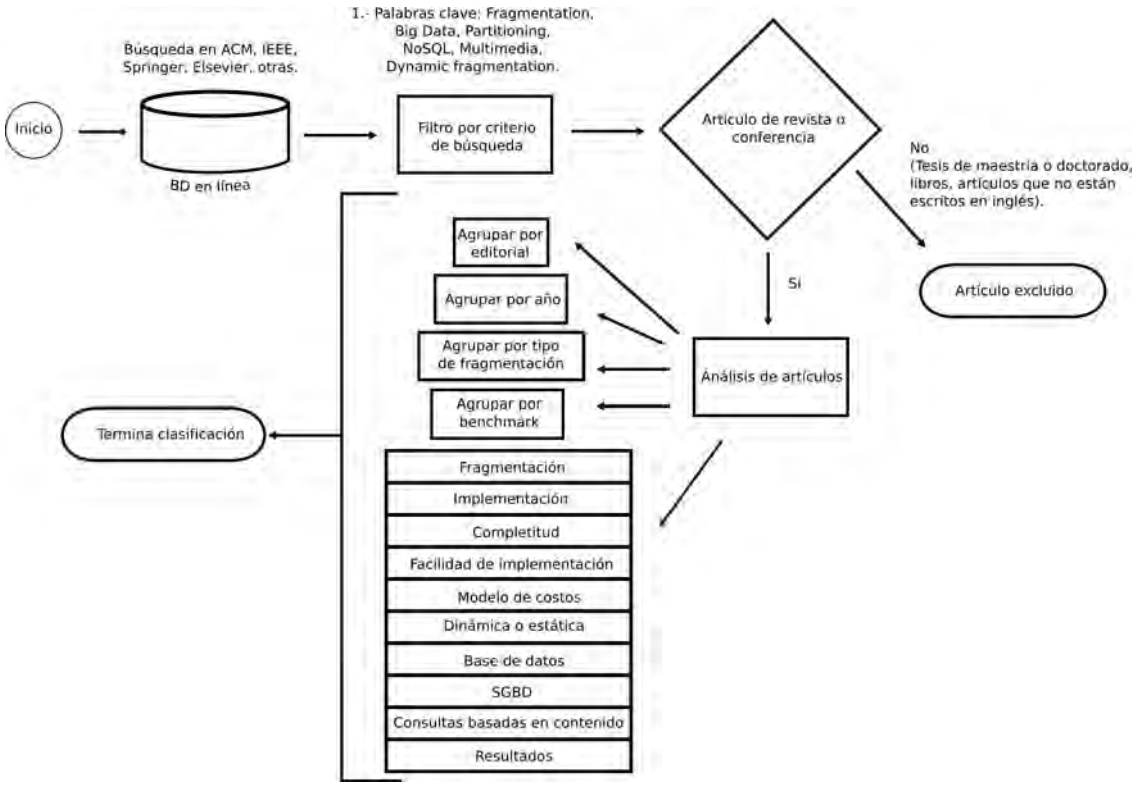


Figura 3.2. Diagrama de flujo de la metodología de búsqueda

La fragmentación dinámica es un tema muy abordado en la literatura de trabajos relacionados con la fragmentación aplicada para mejorar el desempeño de las bases de datos distribuidas. Para dar un enfoque que contribuya a las principales características de este trabajo, es necesario conocer los trabajos relacionados. En la Figura 3.2 se muestran todas las etapas de la metodología de búsqueda. La búsqueda de trabajos se realizó en las principales bibliotecas digitales de editoriales científicas, ACM Digital Library, IEEE Xplore Digital Library, SpringerLink y Science Direct (Elsevier). Los trabajos encontrados que no son publicados por dichas editoriales se clasifican en "Otros". La búsqueda consistió en encontrar todos los artículos que contienen las siguientes palabras clave: *Fragmentation*, *Big Data*, *Partitioning*, NoSQL, *Multimedia*, *Dynamic Fragmentation*. Una vez obtenidos todos los trabajos, se aplicó un filtro y se descartaron todos los trabajos que son tesis de maestría o doctorado, así como los libros y artículos que no estén escritos en inglés. Los artículos resultantes se clasificaron por editorial, por año, por tipo de fragmentación, por *benchmark* y se describieron los costos más utilizados. Cada artículo se analizó detalladamente indicando el tipo de fragmentación, si tiene todos los detalles para aplicar el método, si los elementos

utilizados son fáciles de implementar, si tiene un modelo de costos, también se detalla si la fragmentación es estática o dinámica, qué tipo de base de datos se manejó, qué SGBD se utilizó y, finalmente, si el método contempla CBIR. Los resultados del análisis comparativo se presentan en el Capítulo 4.

3.2. Diseño

Esta sección muestra el diseño de la estrategia de fragmentación siguiendo la metodología de desarrollo UWE. Se llevan a cabo todos los diagramas y pasos de la metodología elegida para el desarrollo del enfoque.

La Figura 3.3 representa la arquitectura "Modelo Vista Controlador" (MVC) de la aplicación.

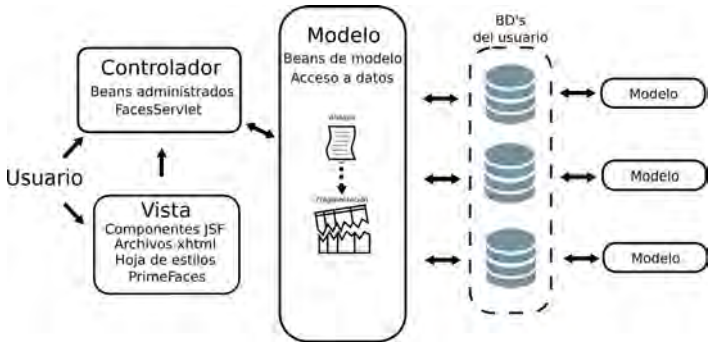


Figura 3.3. Arquitectura para la fragmentación y refragmentación

En la vista están las páginas que interactúan con el usuario para solicitar información sobre las bases de datos y cómo acceder a ellas. Los *beans* administrados de JSF (JavaServer Faces) que gestionan el flujo de información se encuentran en el controlador. En el modelo, se definen las reglas del negocio, que en este escenario particular será la aplicación de la técnica de fragmentación y refragmentación, contemplado el cálculo del modelo de costos para cada carga de trabajo a lo largo del tiempo. El modelo izquierdo se relaciona con la fragmentación y el derecho con la refragmentación en cada nodo de la red.

3.2.1. 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, es decir, muestra las actividades a las que está

relacionado cada actor. En la Tabla 3.1 se definen los actores de la aplicación y una breve descripción del papel que desempeñan.

Tabla 3.1. Actores de la aplicación Web

Actor	Descripción
Administrador de la base de datos (DBA)	El DBA es el único actor en el sistema y su única función es solicitar la fragmentación y poner en marcha la refragmentación proporcionando los archivos correspondientes para su proceso.

Una vez ya definido el único actor en la Tabla 3.1, en la Figura 3.2 se muestra el diagrama de casos de uso, relacionando al actor con sus actividades. Como se observa, el actor se relaciona con una actividad.

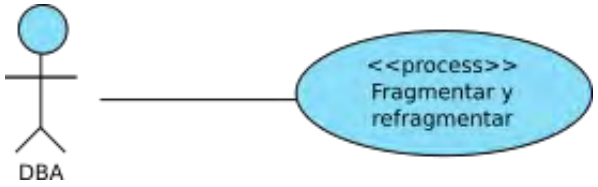


Figura 3.4. Diagrama de casos de uso de la aplicación.

En la Figura 3.4 se muestra el caso de uso "Fragmentar y refragmentar" con el estereotipo "Process" (Proceso), como lo sugiere la metodología UWE. Este caso de uso se describe con tres diagramas de actividades, uno para cada tipo de fragmentación, en las Figuras 3.5, 3.6 y 3.7.

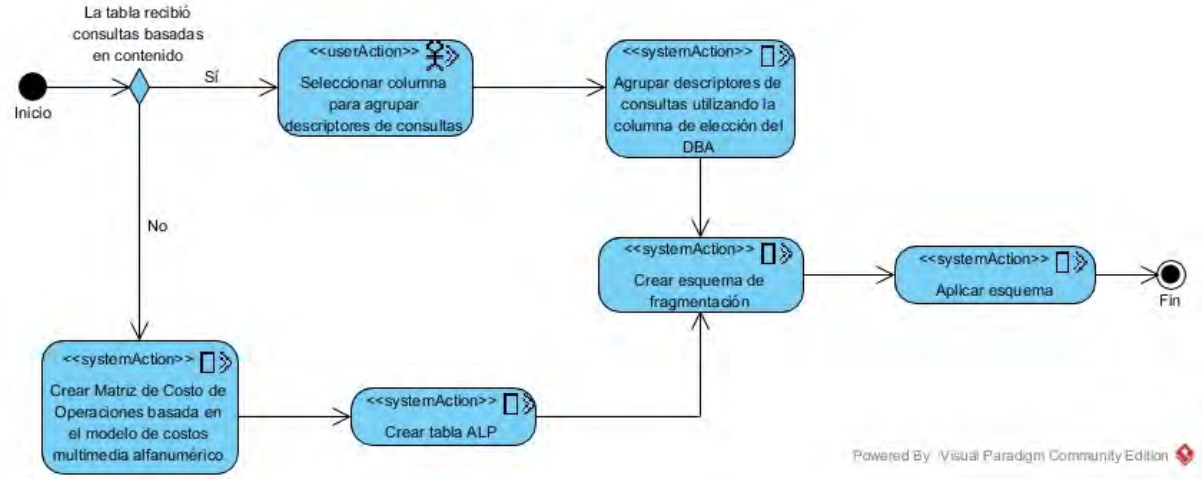


Figura 3.5. Diagrama de actividades de la fragmentación horizontal

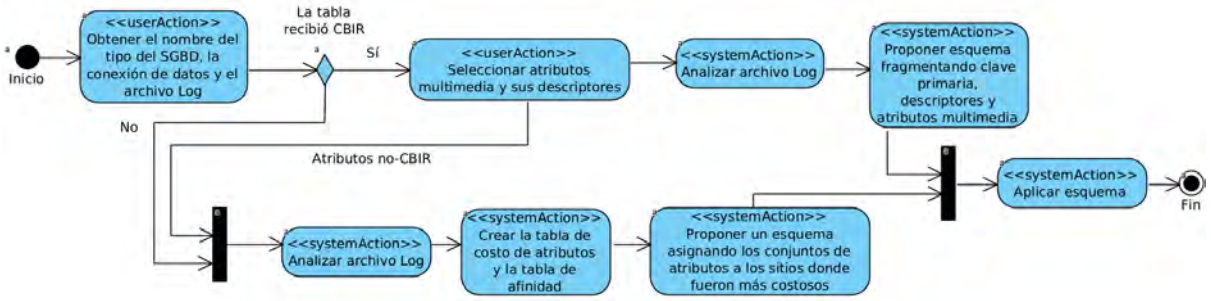


Figura 3.6. Diagrama de actividades de la fragmentación vertical

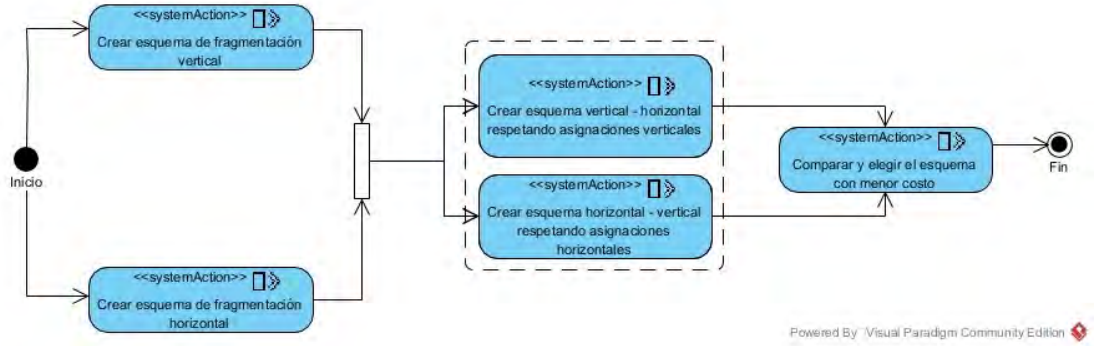


Figura 3.7. Diagrama de actividades de la fragmentación híbrida

El diagrama de actividades de la fragmentación horizontal (Figura 3.5) y el diagrama de actividades de la fragmentación vertical (Figura 3.6) identifican como primer decisión el tipo de consultas que recibe la tabla que se desea fragmentar. Ambos diagramas contemplan CBIR para tratar de diferente forma el flujo de la fragmentación. La fragmentación horizontal, al recibir consultas basadas en contenido, utiliza "Seleccionar columna para agrupar descriptores de consultas". La aplicación, al identificar el atributo multimedia y su columna de descripción, propone un esquema de fragmentación agrupando tuplas mediante su descripción. Como último paso, el usuario aplica físicamente el esquema propuesto. La fragmentación horizontal sin CBIR se basa en el flujo de trabajo propuesto por Castro-Medina et al. (2020), sin contemplar la replicación y el entorno en la nube.

La fragmentación vertical tomando en cuenta CBIR coloca en fragmentos separados el conjunto de atributos relacionados con CBIR y a los atributos comunes los trata con el flujo de fragmentación de la parte inferior del diagrama mostrado en la Figura 3.6. La fragmentación vertical sin CBIR crea un esquema mediante la tabla del costo de los atributos.

El costo de los atributos se determina por medio de las consultas desempeñadas en el archivo *log* de la base de datos.

Se crea una nueva tabla de afinidad de atributos, se determina el costo de cada conjunto de atributos y se asignan al sitio donde representan el costo más alto. La tabla de afinidad se crea mediante la aparición de conjuntos de atributos en las operaciones realizadas. Ambos flujos de trabajo muestran un esquema propuesto que aprueba el administrador de la base de datos. La última acción que se muestra en el diagrama es la aplicación del esquema en los diferentes sitios.

La fragmentación híbrida, mostrada en la Figura 3.7, crea esquemas de ambos tipos y los combina, produciendo esquemas vertical-horizontal y horizontal-vertical. Se determinan los costos mediante la simulación de la ejecución de las mismas consultas encontradas en el archivo *log*. El esquema con menor costo se elige para aplicarlo.

Las fragmentaciones dinámicas de cada tipo tienen en cuenta dos factores adicionales: 1) Se realizan nuevas fragmentaciones cuando se superan los umbrales de desempeño y operaciones, y 2) La inclusión de un ejecutable que observa no solo un fragmento sino todos los presentes en el servidor donde se encuentra. El flujo de trabajo de las fragmentaciones dinámicas de cada tipo se muestra en la Figura 3.8.

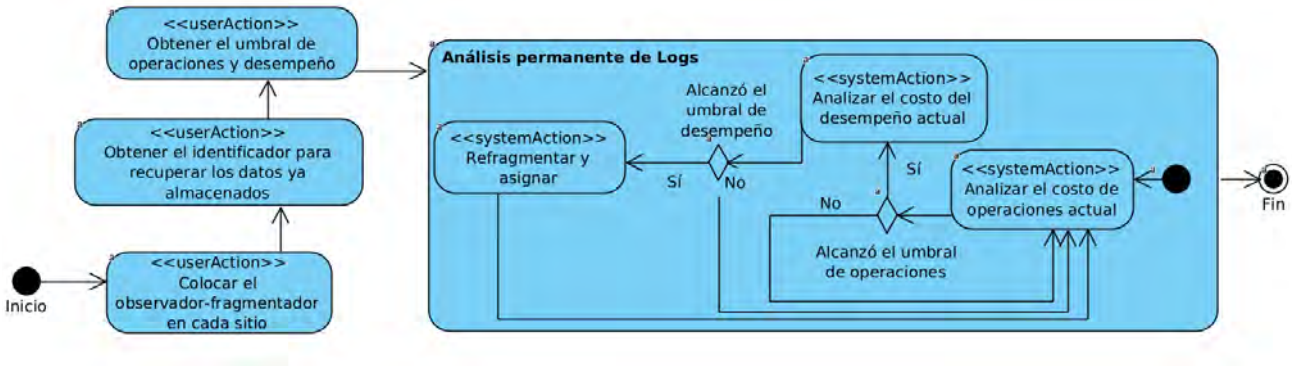


Figura 3.8. Diagrama de actividades de los tres tipos de fragmentaciones.

3.2.2. 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 representa por medio de tres diagramas principales: diagrama conceptual, diagrama lógico y diagrama físico de la base de datos. Se observan estos diagramas en las Figuras 3.9, 3.10 y 3.11.

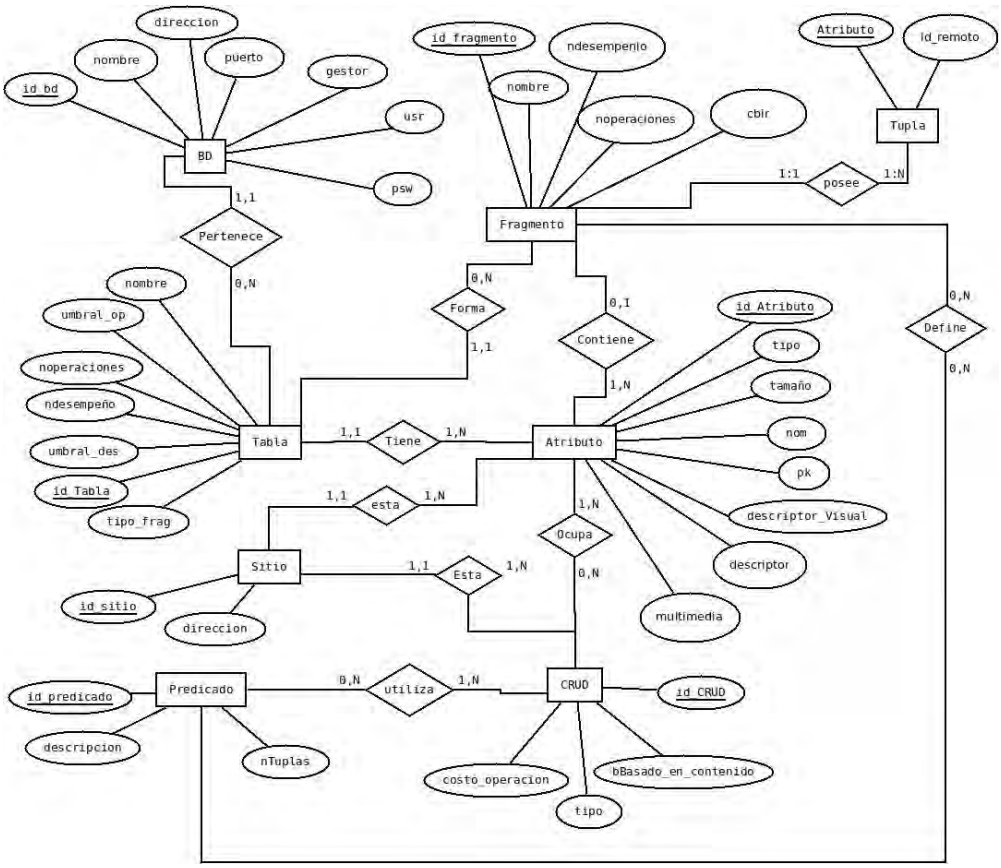


Figura 3.9. Diagrama conceptual de la aplicación

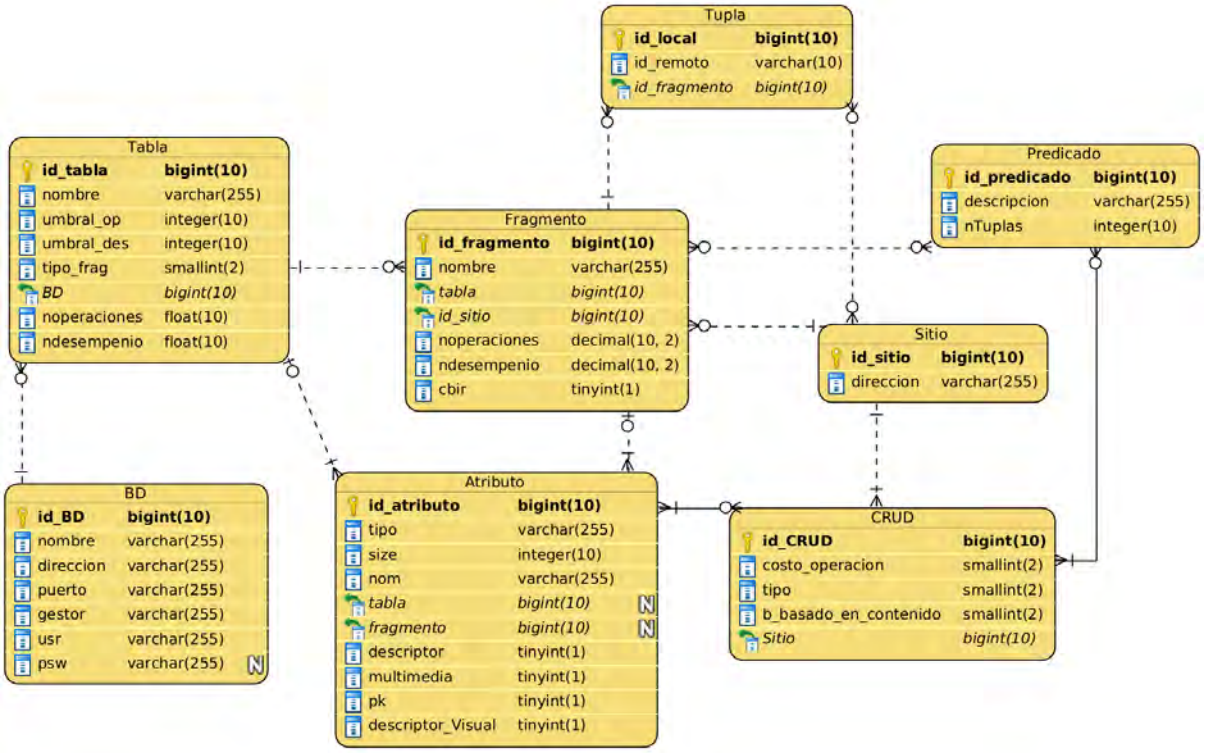


Figura 3.10. Diagrama lógico de la aplicación

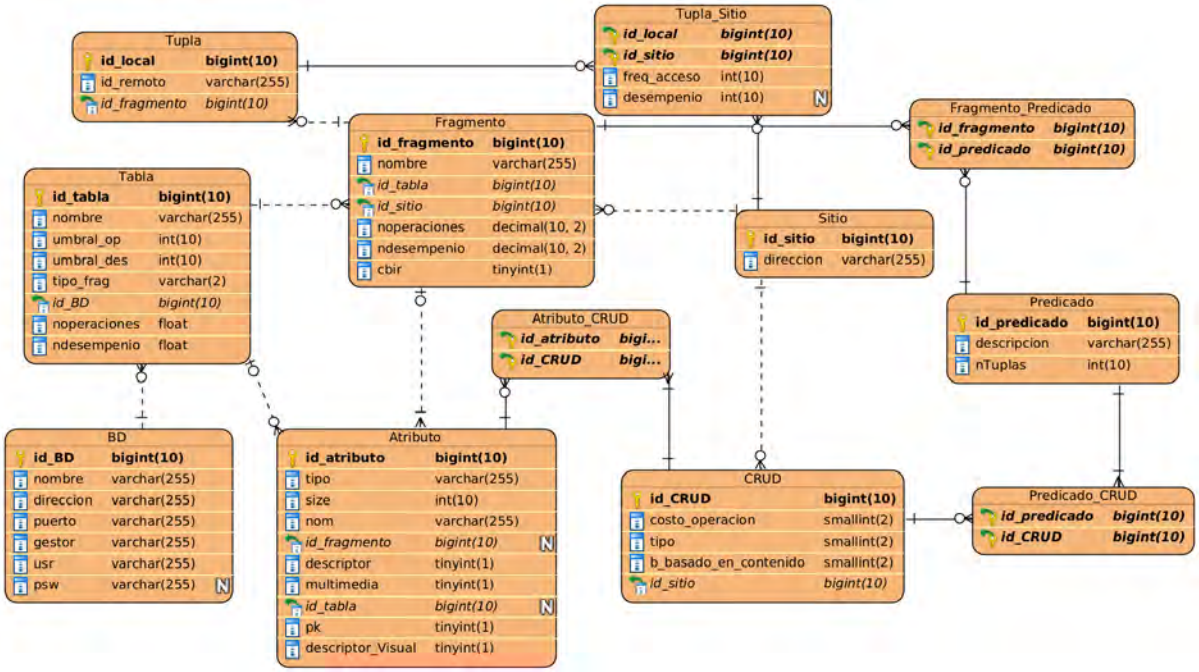


Figura 3.11. Diagrama físico de la aplicación

Tablas de la base de datos

BD: La tabla *BD* almacena la información relacionada con los datos de conexión a las bases de datos del usuario. Se considera que los datos de conexión sean los mismos para todos los sitios de la red, solo la dirección IP es diferente. Se contempla que la contraseña sea opcional si el SGBD permite la deshabilitación de la autenticación.

Tabla: La relación *Tabla* almacena la información relacionada con los datos de la tabla que se fragmenta y refragmenta. En esta relación se almacenan los umbrales que el usuario determina para desempeñar la refragmentación.

Atributo: Para la fragmentación vertical, la relación *Atributo* almacena todo lo relacionado con los atributos de la tabla y los fragmentos. Esta relación contiene los atributos *descriptor* y *multimedia* para identificarlos en el flujo de trabajo con CBIR.

Tupla: La tabla *Tupla* se utiliza para la fragmentación horizontal dinámica contemplando CBIR. El enfoque utiliza el uso de las tuplas para determinar nuevos fragmentos.

Fragmento: La relación *Fragmento* almacena todo lo relacionado con los fragmentos generados de manera estática y dinámica. La tabla toma en cuenta los valores de operaciones y desempeño iniciales. Estos valores se utilizan para determinar el momento en el que los valores actuales superan los umbrales.

Sitio: La relación *Sitio* contiene la IP de los sitios de todos los fragmentos y nodos de la red ocupados en el archivo *log* de cada SGBD.

CRUD: La relación *CRUD* almacena toda la información de las operaciones encontradas en el archivo *log*.

Predicado: La tabla *Predicado* se encarga de almacenar los predicados utilizados en las operaciones de la tabla CRUD. Los predicados se ocupan ampliamente en la fragmentación horizontal.

3.2.3. Modelo de navegación

Mediante 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 navega un usuario dentro de un sitio o de una aplicación. En las Figuras 3.12, 3.13 y 3.14 se presentan los modelos de navegación de cada tipo de fragmentación.

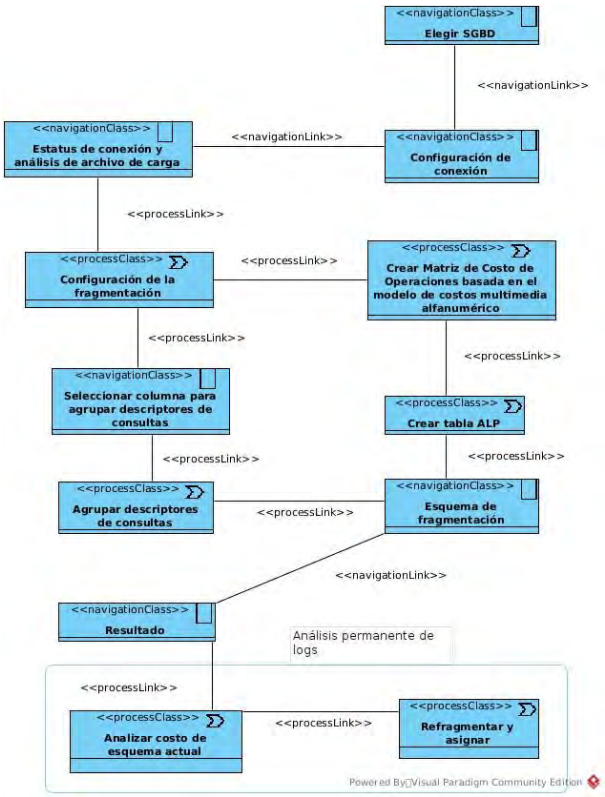


Figura 3.12. Modelo de navegación de la fragmentación horizontal

Los tres diagramas del modelo de navegación poseen una clase estereotipada *<<navigationClass>>* y nombrada *Elegir SGBD.* Este enfoque se diseña para funcionar con cuatro SGBD: MySQL, PostgreSQL, Postgres-XL y MongoDB. Al seleccionar el gestor se deben configurar los datos de conexión (IP, nombre de la base de datos, puerto, usuario, contraseña y nombre de la tabla a fragmentar). El siguiente paso es seleccionar el archivo *log* para someterlo al análisis para obtener el costo de operaciones y desempeño. La fragmentación se configura fijando los umbrales que debe alcanzar el flujo de trabajo dinámico para desempeñar la refragmentación. Además, en este paso se identifica si la tabla

que se quiere fragmentar recibe CBIR. En este punto el flujo de navegación se bifurca para tratar de diferente forma la fragmentación con CBIR. Si la tabla recibe CBIR, el usuario debe seleccionar el atributo que describe al atributo multimedia. Si la tabla no recibe CBIR, se crea una tabla que contiene los costos de cada predicado encontrado en el *log* para determinar el atributo más costoso. Al conocer el atributo más costoso mediante la tabla ALP (*Attribute Locality Precedence*, Precedencia Local de Atributos), se propone un esquema con fragmentos producido por los predicados del atributo más costoso. El resultado del esquema CBIR propone agrupar tuplas mediante el atributo que describe a la columna multimedia y de esta manera generar fragmentos. De la misma manera que *Elegir SGBD*, *Analizar costo del esquema actual* y *Refragmentar y asignar* están presentes en los tres diagramas para desempeñar la fragmentación dinámica.

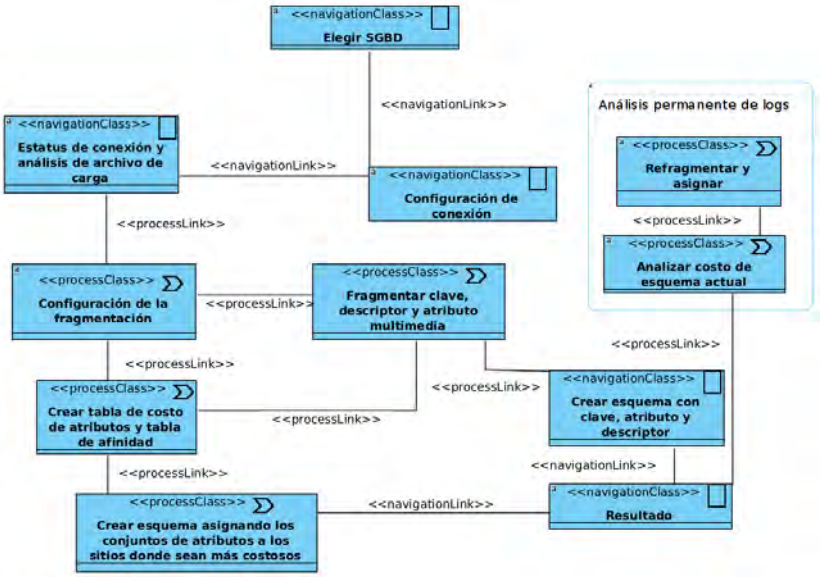


Figura 3.13. Modelo de navegación de la fragmentación vertical

La Figura 3.13 presenta, después de la configuración de la fragmentación, dos flujos de navegación. Si la tabla que se desea fragmentar recibe CBIR, los atributos relacionados con la columna multimedia se colocan en fragmentos separados. Si la tabla no recibe CBIR, se determina el costo de los atributos y se crea un primer esquema para identificar el nuevo costo.

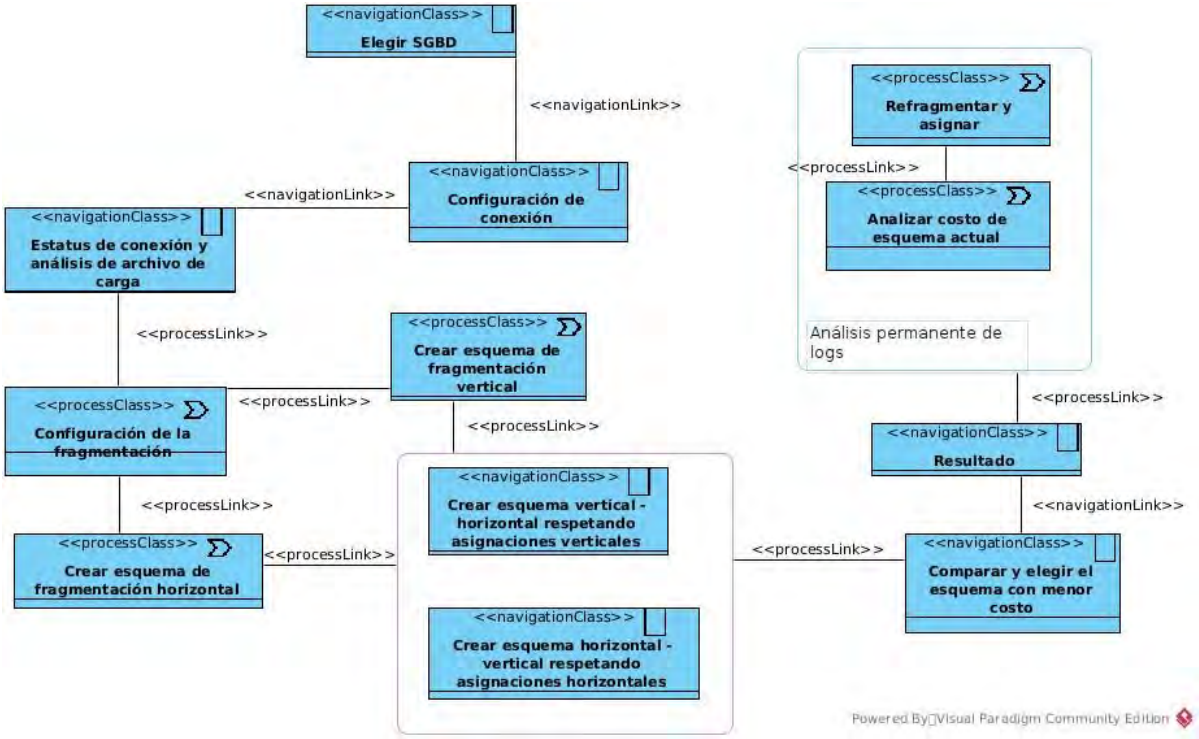


Figura 3.14. Modelo de navegación de la fragmentación híbrida

La Figura 3.14 presenta, después de la configuración de la fragmentación, dos clases de proceso que se llevan a cabo de manera simultánea y, de la misma manera, las siguientes dos. Este flujo de navegación combina los dos modelos de navegación anteriores, mezclando los costos y considerado alguno de los dos primero. Se comparan los costos de los esquemas vertical-horizontal (vertical como principal) y horizontal-vertical (horizontal como principal). El esquema combinado más barato se elige para su aplicación.

3.2.4. Modelo de presentación

El modelo de presentación muestra la simulación de las páginas que contiene el enfoque estático (aplicación Web) de la fragmentación. Cada página describe los componentes que la conforman de manera visual. La Figura 3.15 presenta la página *configuración de conexión y fragmentación*.

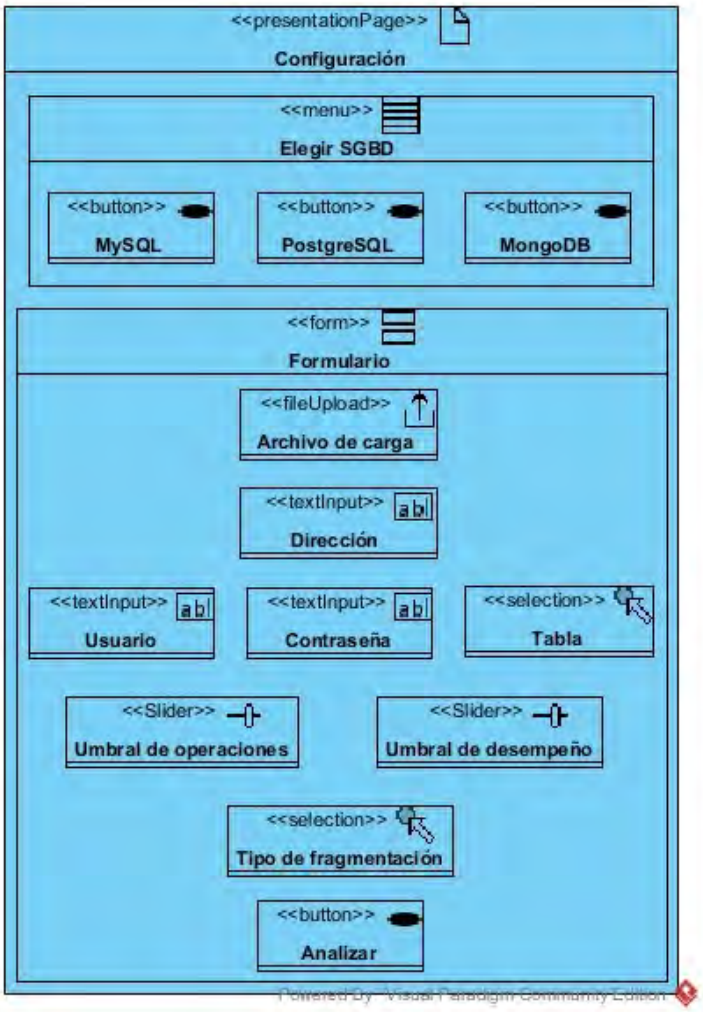


Figura 3.15. Página *Configuración de conexión y fragmentación* del modelo de presentación

Configuración de conexión y fragmentación es la página encargada de obtener los datos del tipo de SGBD, datos de la conexión y configurar la fragmentación. En esta vista se colocan los archivos *log* y se fijan los umbrales de operaciones y desempeño mediante *Sliders*. Por último, se elige el tipo de fragmentación que se desempeñará.

La Figura 3.16 presenta la página del *Análisis del archivo de carga y esquema final de la fragmentación horizontal.* Se muestran los diálogos cuando se incluye CBIR y cuando no. El recuadro *Columna* se usa solo en caso de utilizar CBIR. Los cuadros *MCRUD* y *ALP* solo se utilizan cuando la tabla que se desea fragmentar no recibe CBIR.

El esquema propuesto utiliza diferentes formatos cuando se selecciona CBIR y cuando no y se representa en el cuadro *Tabla final*. Si el usuario decide aplicar el esquema, el botón *Fragmentar y asignar* es el encargado de ejecutar la acción.

La Figura 3.17 muestra la página *Análisis del archivo de carga y esquema final de la fragmentación vertical.* Si la tabla no recibe CBIR, se utiliza el recuadro *Tabla costo de atributos.* En caso contrario se genera el esquema final agrupando los atributos relacionados con la columna multimedia (atributo multimedia y atributo que describe a la columna multimedia).

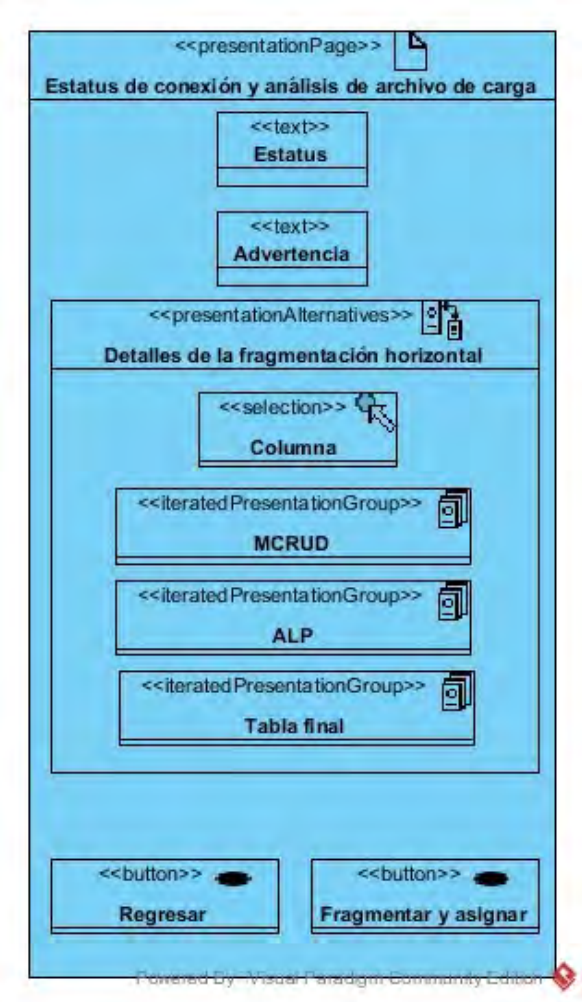


Figura 3.16. Página *Análisis del archivo de carga y esquema final de la fragmentación horizontal* del modelo de presentación

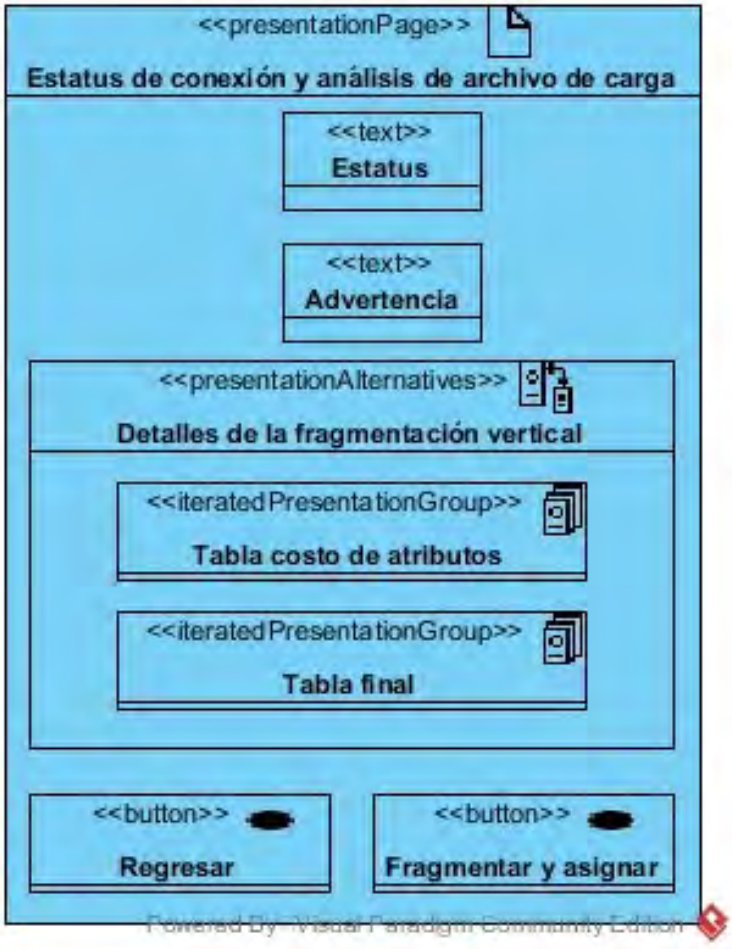


Figura 3.17. Página *Análisis del archivo de carga y esquema final de la fragmentación vertical* del modelo de presentación

La Figura 3.18 presenta la pantalla final de la fragmentación híbrida, combinando las dos anteriores. Al producir el recuadro *Esquema con menor costo,* ya se llevó a cabo la comparación de los costos horizontal-vertical y vertical-horizontal.

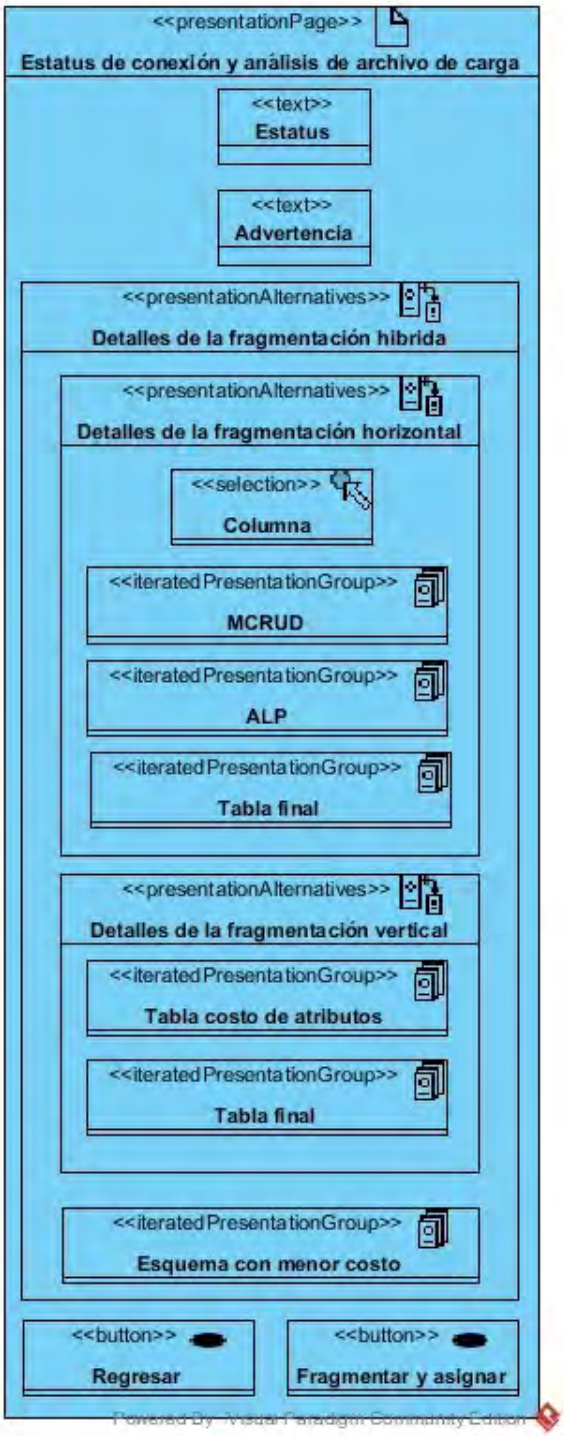


Figura 3.18. Página *Análisis del archivo de carga y esquema final de la fragmentación híbrida* del modelo de presentación

Al desempeñar la fragmentación estática se presenta un diálogo con el estatus de la acción realizada, mostrando una vista resumida de los fragmentos finales. La página relacionada con esta actividad se presenta en la Figura 3.19.

Figura 3.19. Página *Resultados* del modelo de presentación

3.2.5. Modelo de proceso

El diagrama de procesos se basa en el diagrama de actividades, agregando diversos controles que enriquecen la especificación de la aplicación. Las Figuras 3.20, 3.21 y 3.22 presentan el modelo de procesos. En los tres diagramas del modelo de procesos se representan las clases BD y Tabla. Además, se muestra el recuadro del análisis permanente de los *logs*. En la Figura 3.20 se presenta el diagrama de proceso de la fragmentación horizontal, agregando diversos componentes como los nodos de decisión y la representación de los objetos dentro del flujo de actividades.

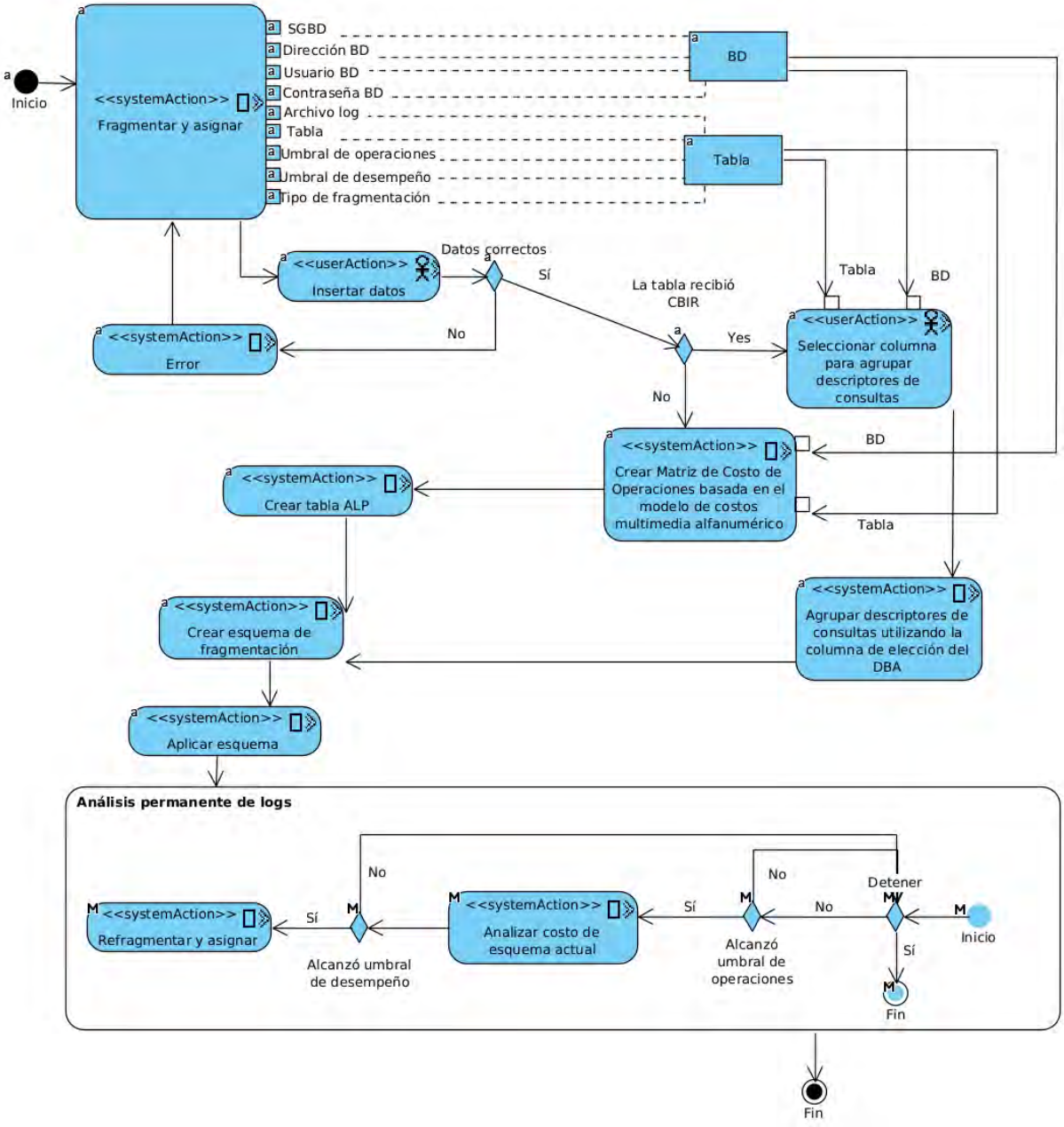


Figura 3.20. Diagrama de la fragmentación horizontal del modelo de procesos

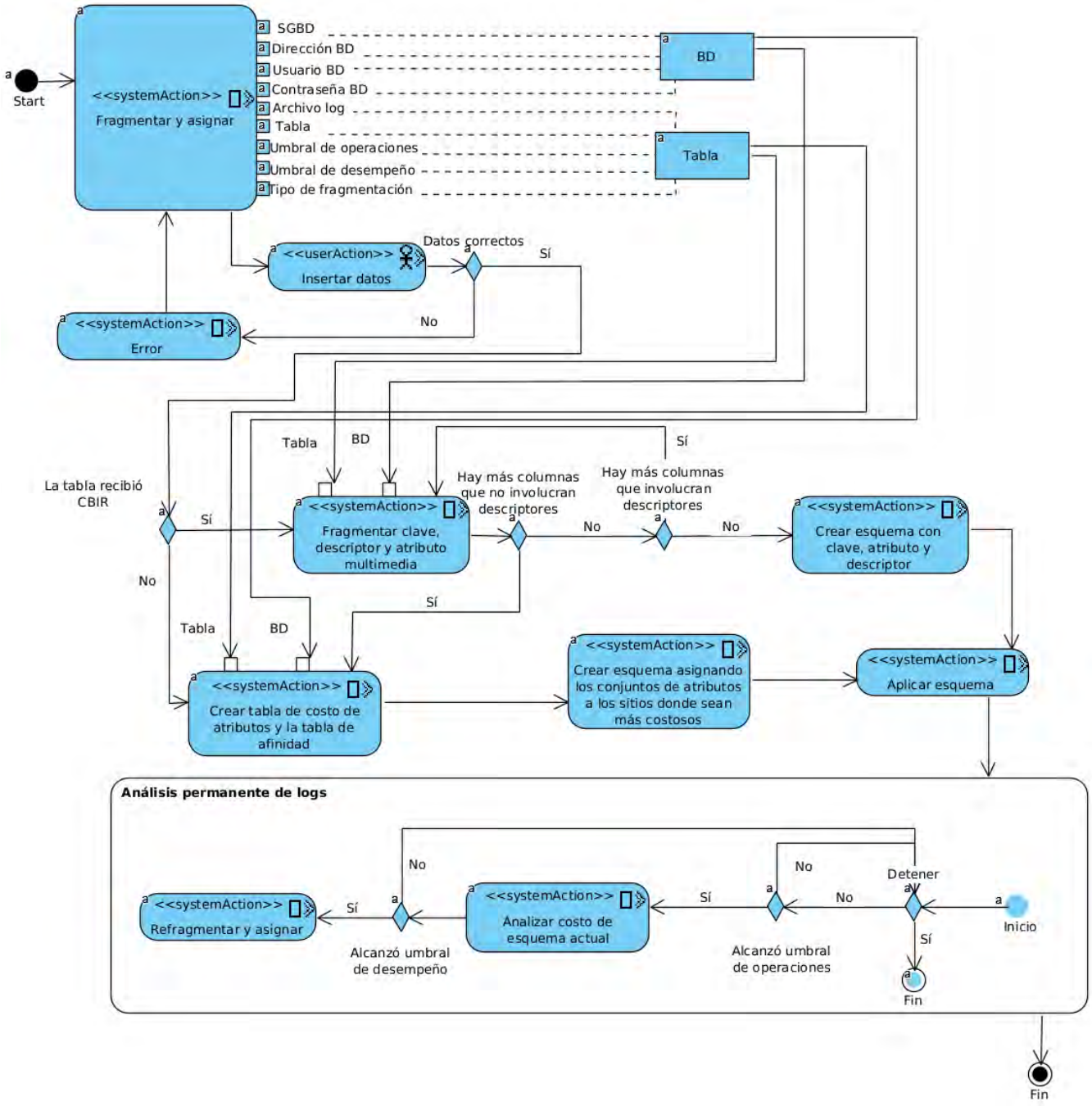


Figura 3.21. Diagrama de la fragmentación vertical del modelo de procesos

El modelo de procesos de la fragmentación vertical muestra las actividades relacionadas con las dos clases en la Figura 3.21. La representación de las clases en este diagrama presenta la relación entre ellas y la interacción con el usuario por medio de los recuadros estereotipados

como *<<userAction>>,* sin embargo, el diagrama no representa las clases que modelan los datos que se obtienen de la base de datos y su relación con los SGBD de los usuarios. Como ya se mencionó en el diagrama de actividades, la Figura 3.21 aborda la fragmentación de forma diferente cuando la tabla que se desea fragmentar recibe CBIR. En el diagrama de procesos se aprecia este nodo de decisión y los nodos de decisión relacionados con los umbrales de operaciones y desempeño.

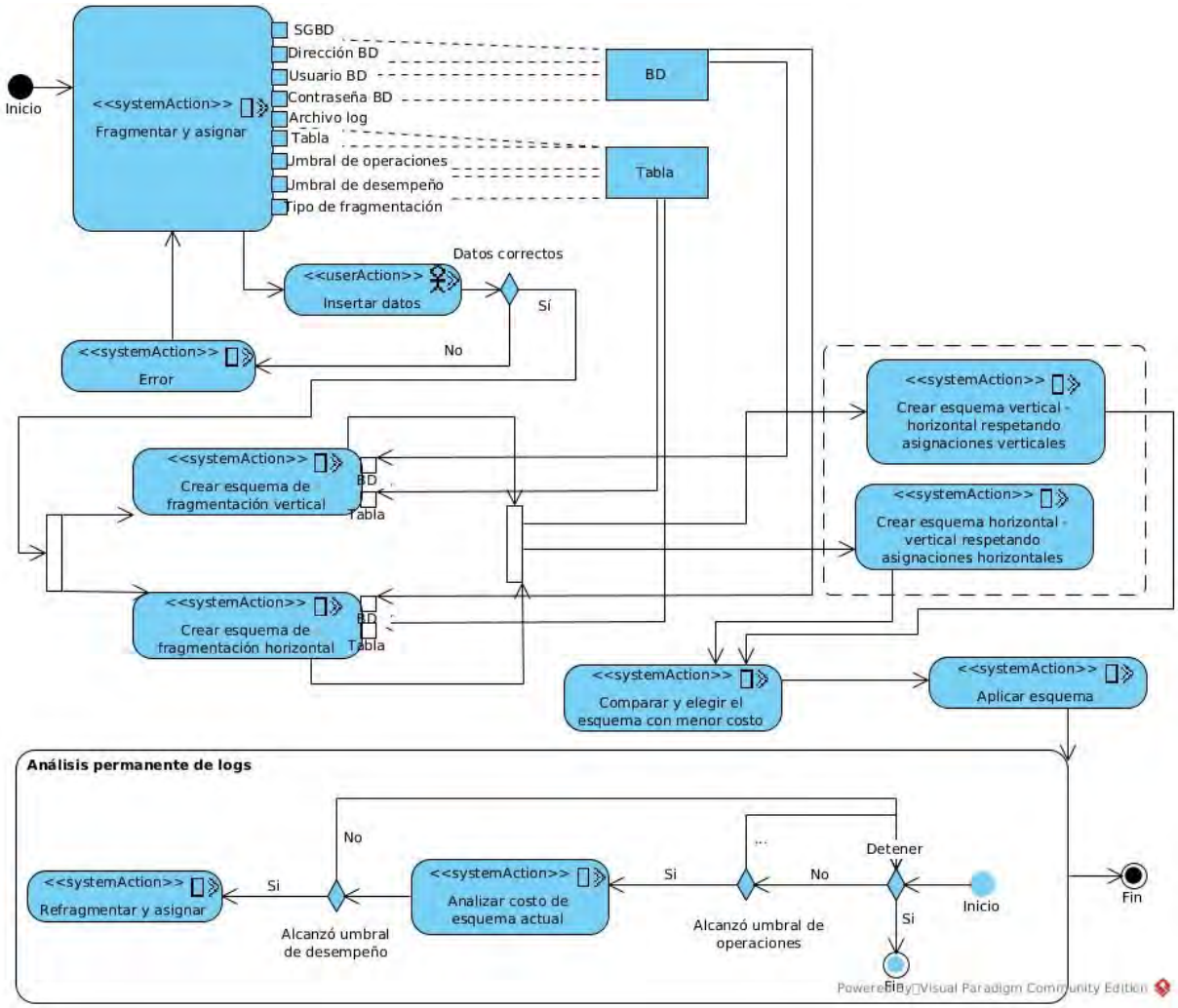


Figura 3.22. Diagrama de la fragmentación híbrida del modelo de procesos

La fragmentación híbrida, en la Figura 3.22, representa las acciones que se llevarán a cabo en la misma etapa. El análisis de *logs* se lleva a cabo de manera permanente en cada nodo del usuario en los tres modelos de procesos. La fragmentación híbrida utiliza y aplica el esquema combinado más barato bajo la carga de trabajo encontrada en el *log*.

3.3. Implementación

La etapa de implementación tiene la finalidad de desarrollar los tres métodos de fragmentación bajo el diseño mostrado. Esta etapa se lleva a cabo con las tecnologías ya mencionadas, siendo Java el lenguaje principal. JavaServer Faces tiene un rol principal en el desarrollo de la aplicación Web, facilitando la aplicación de la primera parte de la arquitectura al definir *beans* administrados, *beans* de modelo y páginas XHTML (*Extensible Hypertag Markup Language*, Lenguaje de Marcado de Hiperetiquetas extensible) para la vista. La biblioteca PrimeFaces amplía la capacidad del lenguaje para la vista y los componentes. Java en su versión estándar se utiliza para lograr la fragmentación dinámica, cumpliendo la última parte de la arquitectura, definiendo modelos en cada uno de los sitios. MySQL es el sistema gestor de bases de datos utilizado para ambas partes de la arquitectura. El Capítulo 4 describe detalladamente la implementación del diseño con las tecnologías ya descritas.

3.4. Validación

La validación de la implementación se realiza mediante bases de datos multimedia. Para los tres métodos se utilizan tablas que reciben CBIR y tablas que solo contienen elementos multimedia, de esta manera se evalúan los flujos de las bifurcaciones en los diagramas de procesos. Las cargas de trabajo se generan mediante un simulador de operaciones. La etapa de validación y el simulador de operaciones se detallan en el Capítulo 4.

3.5. Modelos de costos

Los modelos de costos para los tres tipos de fragmentación incluyen el cálculo del costo de operaciones y el costo de desempeño, siendo este último el que se describe bajo ecuaciones en esta sección. El costo de operaciones es la sumatoria del número de operaciones desempeñadas en cada fragmento (o tabla inicialmente) desde la última refragmentación (en el caso de la primera fragmentación, desde cero) hasta el momento del cálculo. Los umbrales de operaciones y desempeño se fijan mediante porcentajes que condicionan la refragmentación. Ambos umbrales deben superarse para llevar a cabo una nueva fragmentación.

La Ecuación (1) describe el costo de desempeño de la fragmentación horizontal contemplando CBIR.

$$HorCBIR_i = \sum_{j=1}^{N} (VDO_j * VR_j) \quad (1)$$

HorCBIR es el costo de desempeño de un fragmento o tabla *i*, *VDO* es el valor de la operación descrito en la Tabla 3.2 y *VR* es el valor remoto. *VR* toma el valor de uno cuando la operación *j* es local y dos cuando es remota, y *N* es el número de operaciones en el archivo *log* considerando el predicado del fragmento.

Tabla 3.2. Valores para cada tipo de operación

Tipo de operación	Valor
Lectura	1
Eliminación	2
Creación	2
Actualización	3

La Tabla 3.2 muestra los valores que se asignaron para cada tipo de operación. Los valores descritos se utilizan en todos los modelos de costos de este trabajo. Realizar una operación de forma remota es más caro que realizarla de manera local. El cálculo del desempeño tiene en cuenta el sitio donde se ejecutan las operaciones.

La refragmentación, al superarse ambos umbrales, en la fragmentación horizontal contemplando CBIR, se realiza mediante el gráfico de la curva Gaussiana para normalizar el uso de las tuplas de todos los fragmentos en cada sitio. El gráfico se divide en dos y las tuplas se asignan al sitio que represente un costo mayor para las tuplas de cada lado.

La Ecuación (2) describe el costo de desempeño de la fragmentación horizontal cuando no se recibe CBIR.

$$Hor_i = \sum_{j=1}^{N} (NTuplas_j * VDO_j * VR_j) \quad (2)$$

Hor es el costo de desempeño de la fragmentación horizontal cuando la tabla o fragmento *i* que se desea fragmentar o refragmentar no recibe CBIR. *Ntuplas* representa el número de tuplas de los predicados de cada operación *j. VDO* es el valor de la operación relacionado a la Tabla 3.2. *VR* es el valor remoto, descrito en la ecuación del costo HorCBIR, de cada operación *j* y N es el número de operaciones encontradas en el *log*.

La Ecuación (3) describe el costo de desempeño de la fragmentación vertical contemplando CBIR y cuando no se recibe CBIR.

$$Ver_i = \sum_{j=1}^{N} \left(\sum_{u=1}^{H} VDO_u * VR_u * CR_u \right)_j * TA_j \tag{3}$$

Ver es el costo de la fragmentación vertical cuando se contemplan operaciones CBIR y cuando la tabla no las recibe. *j,* a diferencia de las ecuaciones anteriores, representa cada atributo incluido en la tabla o fragmento. *u* es cada operación que incluye al atributo *j* en el mismo sitio, con los mismos atributos, representando de esta manera la frecuencia de *u. VDO* es el valor de la operación relacionado con la Tabla 3.2. *CR* es el costo de reuniones tomando valores *n+1*, donde *n* es el número de reuniones. *TA* es el tamaño de cada atributo en bytes. *N* es el número de atributos en la tabla o fragmento. *H* es el número de operaciones en el *log*.
Las ecuaciones del costo de la fragmentación híbrida se muestran a continuación. La Ecuación (4) se utiliza cuando la tabla recibe CBIR y la Ecuación (5) cuando la tabla no recibe este tipo de operaciones.

$$HibCBIR = \frac{\sum_{j=1}^{N} \left(VDO_j * VR_j * CR_j \right) + \sum_{k=1}^{M} \left(\sum_{l=1}^{P} \left(VDO_l * VR_l * CR_l \right)_k * TA_k \right)}{2} \tag{4}$$

$$Hib = \frac{\sum_{j=1}^{N} \left(NTuplas_j * VDO_j * VR_j * CR_j \right) + \sum_{k=1}^{M} \left(\sum_{l=1}^{P} \left(VDO_l * VR_l * CR_l \right)_k * TA_k \right)}{2} \tag{5}$$

En ambas ecuaciones de la fragmentación híbrida se realiza el promedio de las ecuaciones anteriores (horizontal y vertical). Al superar los umbrales definidos por el usuario en la fragmentación dinámica, *CR* se toma en cuenta para comparar el costo del esquema

respetando asignaciones horizontales y el costo respetando asignaciones verticales.

3.6. Algoritmo para la fragmentación horizontal contemplando CBIR

Para complementar la etapa de modelado de la fragmentación horizontal tomando en cuenta CBIR, se presenta el Algoritmo 1 y 2 para llevar a cabo la tarea descrita por los diagramas. El Algoritmo 1, Vigilante, representa el pseudocódigo para analizar los cambios en cada operación realizada y desencadenar una nueva fragmentación. El *token* que utiliza el Algoritmo 1 como parámetro lo proporciona la aplicación Web, que es un identificador para obtener el conjunto completo de fragmentos, tuplas, estadísticas, sitios y datos de la base de datos.

Algoritmo 1: Vigilante

```
1  input: Token
2  fragmentos[]← Token.obtener fragmentos
3  for i from 0 to size of fragmentos[] do
4      nOperaciones←(fragmentos[i].Porcentaje de umbral de operaciones*
5                    fragmentos[i].Número de las operaciones iniciales)/100
6      nOperaciones← nOperaciones+fragmentos[i].Número de las operaciones iniciales
7      if fragmentos[i].Número de operaciones actual >= nOperaciones then
8          nDesempeño←(fragmentos[i].Porcentaje de umbral de desempeño*
9                      fragmentos[i].Número del desempeño inicial)/100
10         nDesempeño ←nDesempeño+fragmentos[i].Número de desempeño inicial
11         if fragmentos[i].Número de desempeño actual >= nDesempeño then
12             x←fragmentador(fragmentos[i], Token)
13             escribir en el archivo de bitacora x
```

Figura 3.23. Algoritmo de fragmentación horizontal contemplando CBIR (parte estática)

Como se observa en las líneas 4 y 8, los umbrales de operación y desempeño se obtienen como un porcentaje y deben interpretarse bajo sus respectivos valores para determinar cuándo se alcanzan.

El Algoritmo 1 utiliza al Algoritmo 2, Fragmentador, que se encarga de aplicar la fragmentación. Como parámetros, el Algoritmo 2 recibe los datos del fragmento (o la tabla) a fragmentar y el *Token*. En ambos algoritmos se omiten diferentes detalles para resumirlos y hacerlos más fáciles de entender. Uno de ellos es el registro de las nuevas estadísticas en la base de datos de la aplicación Web una vez realizada la fragmentación. En el Algoritmo 2, la implementación de la fragmentación en MongoDB está altamente relacionada con el lenguaje

en el que se implementa el algoritmo, sin embargo, esta parte se describe de manera general usando instrucciones del SGBD NoSQL.

En la aplicación particular del Algoritmo 2 en el vigilante-fragmentador, implementado en Java, las líneas 3-11 se llevaron a cabo como consultas a la base de datos en la aplicación Web. El Algoritmo 2 tiene en cuenta las siguientes condiciones previas:

- Al usar la aplicación Web, los otros sitios del usuario son del mismo tipo de base de datos.
- En cada sitio, hay una base de datos con las mismas credenciales con permisos de lectura y escritura que se utilizaron para fragmentar la primera vez.
- Las direcciones IP de los sitios son públicas. Si son privados, la aplicación Web solo funcionará en una intranet.
- Las direcciones IP de los sitios son estáticas.

La variable *imparPar*, en la línea 13, permite la fragmentación para tuplas impares y luego para tuplas pares cambiando el vector a su vez y la dirección IP a la que se asignarán las tuplas. La disposición de las tuplas de menor a mayor frecuencia y la separación de pares e impares provoca una representación del gráfico de una curva Gaussiana para normalizar los datos. Separar tuplas pares e impares es similar a dividir el gráfico gaussiano en dos.

La línea 20 tiene en cuenta MongoDB, ya que al ser un SGBD NoSQL, las operaciones son diferentes. Las líneas 21 y 27 separan la formación de la instrucción *CREATE* para los SGBD de MySQL y PostgreSQL, ya que la sintaxis es similar, pero no la misma. Otra razón por la que el algoritmo separa los SGBD relacionales se debe a los tipos de datos, ya que algunos no están incluidos en ambos.

El proceso de fragmentación maneja cuatro direcciones IP. El Algoritmo 2 en las líneas 7, 8 y 9 muestra el propósito de tres de ellas. La cuarta dirección es la conexión al servidor de la aplicación Web propuesta en este enfoque, la cual se debe tener en cuenta en el flujo de trabajo de la fragmentación que realiza la aplicación Web, ya que los usuarios deben permitir la conexión remota desde esta IP a sus servidores.

Las líneas 38-54 representan la obtención de tuplas del servidor de origen al servidor de destino. Las líneas 57-62 muestran un proceso más compacto para MongoDB, ya que permite la copia de documentos sin definir una estructura en la colección de destino y fuente, incluso en nuevas versiones de MongoDB, al hacer una conexión a una colección inexistente, el SGBD la crea automáticamente.

Algoritmo 2: Fragmentador

```
1  input: fragmento, Token
2  output: status
3  tuplaIdFrecuencia[][]←obtener identificadores de tupla y la frecuencia de acceso de fragmento.Identificador
4  idImpar[]←ordenar tuplaIdFrecuencia[][] por frecuencia de menor a mayor, numerarlas y obtener impares
5  idPar[]←ordenar tuplaIdFrecuencia[][] por frecuencia de menor a mayor, numerarlas y obtener pares
6  conexión←obtener datos de conexión usando Token
7  IPimpar←obtener el sitio que ocupa más las tuplas impares
8  IPpar←obtener el sitio que ocupa más las tuplas pares
9  IPinicial←obtener el sitio de origen de los fragmentos
10 atributos[]←obtener atributos del fragmento
11 atributoLlave←obtener identificador de la tabla
12 for imparPar from 0 to 2 do
13     if imparPar = 0 then
14         vectorEnTurno[]←idImpar[]
15         ipEnTurno←IPimpar
16     else
17         vectorEnTurno[]←idPar[]
18         ipEnTurno←IPpar
19     sentencia←""
20     if Token.obtener tipo de SGBD != "MongoDB" then
21         if Token.obtener tipo de SGBD = "MySQL" then
22             foreach r←atributo en atributos[] do
23                 sentencia←sentencia + r.obtener nombre + " " + r.obtener tipo de dato + "(" + r.obtener tamaño + "), ";
24             sentencia←sentencia - últimos dos caracteres
25             crearSentencia←"create table "+fragmento.obtener nombre+"_"+(imparPar)+ " ("+sentencia+",
26                         PRIMARY KEY("+atributoLlave.obtener nombre+"))"
27         if Token.obtener tipo de SGBD = "PostgreSQL" o "Postgres-XL" then
28             foreach r←atributo in atributos[] do
29                 sentencia←sentencia + r.obtener nombre + " " + r.obtener tipo de dato + "(" + r.obtener tamaño + ")";
30                 if r.obtener nombre = atributoLlave.obtener nombre then
31                     sentencia←sentencia + " PRIMARY KEY, "
32                 else
33                     sentencia←sentencia + ", "
34             sentencia←sentencia-últimos dos caracteres
35             crearSentencia←"create table "+fragmento.obtener nombre+"_"+(imparPar)+ " ("+sentencia+");"
36         ejecutar crearSentencia en ipEnTurno usando conexión
37         u←""
38         foreach r←atributo in atributos[] do
39             u←u + r.obtener nombre + ", "
40         u←u-últimos dos caracteres
41         foreach y←item in vectorEnTurno[] do
42             consulta←"select " + u + " from " + fragmento.obtener nombre + " where "
43                         + atributoLlave.obtener nombre + "=" + y + ";"
44             t[]←ejecutar consulta en IPinicial usando conexión
45             valores←""
46             for w from 0 to size of t[] do
```

```
46          for w from 0 to size of t[] do
47              valores← valores + t[w] + ", "
48          valores← valores - últimos dos caracteres
49          consulta← "INSERT INTO " + fragmento.obtener nombre + "_" + (imparPar)
50                  + " VALUES (" + valores + ");"
51          eliminarConsulta← "DELETE FROM " +fragmento.obtener nombre + "_" + (imparPar) + " WHERE "
52                  + atributoLlave.obtener nombre + "=" + y + ";"
53          ejecutar consulta en ipEnTurno usando conexión
54          ejecutar eliminarConsulta en IPinicial usando conexión
55      status←"La fragmentación se llevó a cabo de manera satisfactoria"
56  else
57      ejecutar createCollection(fragmento.obtener nombre + "_" + (imparPar)) en ipEnTurno usando conexión
58      foreach y←item in vectorEnTurno[] do
59          where←crear un nuevo documento con _id:y
60          resultSet←buscar usando where en IPinicial usando conexión
61          insertOne con resultSet en una nueva colección llamada fragmento.obtener nombre + "_" + (imparPar)
62          deleteOne usando where en IPinicial
63      status←"La fragmentación se llevó a cabo de manera satisfactoria"
64  return status
```

Figura 3.24. Algoritmo de fragmentación horizontal contemplando CBIR (parte dinámica)

El Algoritmo 3, Fragmentador vertical, representa el pseudocódigo para llevar a cabo la fragmentación sin contemplar CBIR y cuando esta la incluye.

Algoritmo 3: Fragmentador vertical

```
1   input: fragmento
2   output: status
3   lConexión←obtener la conexión local del usuario desde la app de XAMANA
4   rConexión←obtener la conexión remota del parámetro fragmento (Sitio donde será asignado)
5   xConexión←obtener conexión a la base de datos de XAMANA
6   if lConexión.obtener tipo de base de datos = MySQL o PostgreSQL o Postgres-XL
7       atributosSQL←""
8       foreach r←atributo en fragmento.atributos[] do
9           atributosSQL← atributosSQL + r.obtener nombre + " " + r.obtener tipo de dato + "("
10                  + r.obtener tamaño + "), "
11      atributosSQL← atributosSQL-últimos dos caracteres
12      SentenciaCreación← "create table "+fragmento.obtener nombre+ " ("+atributosSQL
13                  +",PRIMARY KEY("
14                  +lConexión.obtener tabla original.obtener llave primaria.obtener
15                  nombre de atributo+"))"
16      ejecutar SentenciaCreación en rConexión
17      SentenciaConsulta← "Select count(*) from " + lConexión.obtener tabla original.obtener nombre
18      tupleNumber ←ejecutar SentenciaConsulta en lConexión
19      for i from 0 to tupleNumber do
20          SentenciaConsulta← "Select " + atributosSQL.obtener nombres + " from "
21          + lConexión.obtener tabla original.obtener nombre + " Limit 1 offset " + i
22          t[]←ejecutar SentenciaConsulta en lConexión
23          valores←""
24          for w from 0 to tamaño de t[] do
25              valores← valores + t[w] + ", "
```

```
26      valores←valores-últimos dos caracteres
27      SentenciaInserción←"insert into " + fragmento.obtener nombre + "("
28                         + fragmento.obtener nombre de atributos + ") values ("
29                         + valores + ")"
30      ejecutar SentenciaInserción en rConexión
31  status←"La fragmentación se llevó a cabo de manera exitosa"
32  else
33      ejecutar createCollection(fragmento.obtener nombre) en rConexión
34      documentos[]← ejecutar find(Projections.fields(Projections.include(fragmento.obtener
35                    nombre de atributos)) en lConexión
36      foreach documento en documentos[] do
37          ejecutar insertOne(documento) in rConexión
38      status←"La fragmentación se llevó a cabo de manera exitosa"
39  guardar la IP del sitio en xConexión (solo si no existen en xConexión)
40  guardar datos del fragmento en xConexión:
41      nombre, id de la tabla, id del sitio, número de operaciones (archivo log),
42      valor de desempeño (ecuación de costos), si es CBIR o no
43  if Fragmentador vertical es estático then
44      guardar información de los atributos en xConexión:
45          nombre, tipo,tamaño, id del fragmento(solo si pertenece a un fragmento),
46          id de la tabla(solo si pertenece a una tabla), descriptor(true o false), multimedia(true o false),
47          llave primaria(true o false)
48  else
49      solicitar la actualización de los fragmentos en el servidor destino (sockets)
50      actualizar información de atributos en xConexión:
51          id del fragmento,id de la tabla(cambiar a null)
```

Figura 3.25. Algoritmo de fragmentación vertical

En la línea 1 se observa el parámetro *fragmento*, y representa un elemento que contiene información sobre el nombre del fragmento, el sitio a donde debe ser asignado, los atributos incluidos, el número de operaciones que utilizan sus atributos, el costo de desempeño y, por último, si es un fragmento que considera CBIR.

La lista de atributos incluida en el parámetro *fragmento*, contiene, para cada atributo, su nombre, el tipo de dato, el tamaño, si es un atributo multimedia, si es un atributo descriptor y si es un atributo de llave primaria.

La línea 3 muestra la variable *lConexión*, representando la conexión a la base de datos local del usuario, donde originalmente, en el escenario anterior a una primera fragmentación, se encuentra la tabla o colección sin fragmentar y, en el caso de la fragmentación dinámica, donde se encuentra el vigilante-fragmentador que indicó una refragmentación. La información de *lConexión* se obtiene mediante la aplicación Web de XAMANA.

rConexión se obtiene mediante la información del parámetro fragmento. *rConexión* es la conexión remota a donde se asignará el fragmento.

xConexión representa la conexión a la base de datos de XAMANA, para almacenar la información requerida para aplicar o continuar con las refragmentaciones.

Se muestra en la línea 6 la condición para determinar el tipo de base de datos utilizada por el usuario. Se contemplan MySQL, PostgreSQL y Postgres-XL como bases de datos relacionales y MongoDB como base de datos no relacional. En las líneas 8-11, para las bases de datos relacionales, se desarrolla la línea de atributos para la creación de la tabla. En las líneas 12-15 se determina la sentencia de creación de tabla, incluyendo la línea de atributos. La línea 16 ejecuta la sentencia en la conexión remota.

Las líneas 17-22 obtienen los valores desde la conexión local para la creación de la sentencia de inserción. El modo de obtención de valores en la conexión local es por tupla y no por lote, tomando en cuenta de esta manera los datos multimedia, caracterizados por contener más información en cada elemento que los alfanuméricos.

La línea 30 ejecuta la inserción en la conexión remota para cada tupla obtenida en *SentenciaConsulta*.

Las líneas 33-38 representan las instrucciones necesarias para llevar a cabo la fragmentación en MongoDB, sin embargo, la implementación de MongoDB en cada lenguaje de programación es diferente. La línea 33 crea la colección en la conexión remota, la 34 obtiene los documentos en la conexión local mediante una proyección de atributos, la 37 inserta cada documento en la conexión remota.

Las líneas 39-42 muestran los datos que se almacenan en la base de datos de XAMANA. Cuando el fragmentador se ejecuta de manera estática (la primera fragmentación, llevada a cabo en XAMANA) se debe almacenar la información relacionada con los atributos incluidos en el fragmento recién creado. Cuando el fragmentador se ejecuta en un entorno dinámico

(las siguientes fragmentaciones, realizadas por el vigilante-fragmentador) se solicita, en la línea 49, la actualización de los fragmentos en el sitio remoto a donde fue asignado el fragmento recién creado. Por último, se actualiza la información de los atributos que, en el caso de la fragmentación dinámica, ya se encuentran agregados a la base de datos de XAMANA. El Algoritmo 3 tiene en cuenta las mismas condiciones previas que el Algoritmo 2.

La fragmentación vertical dinámica para fragmentos que no consideran CBIR utiliza el Algoritmo 1 para determinar cuándo llevar a cabo una nueva fragmentación.

La fragmentación en MongoDB contiene una principal diferencia con el resto de gestores, ya que una colección no tiene un esquema estricto y no se requiere declarar previamente si se añadirá un nuevo campo a los documentos en una inserción. En bases de datos relacionales, si una tupla contiene un atributo que no está presente en el esquema de la tabla, se debe definir previamente la modificación del esquema (ALTER TABLE). Esta modificación afecta a toda la estructura de la tabla y no solo a la tupla. La fragmentación dinámica de esta investigación en MongoDB al no tener una estructura estricta, no realiza el análisis de inserción, modificación y borrado de campos (atributos). La fragmentación dinámica en PostgreSQL, Postgres-XL y MySQL sí toma en cuenta las modificación en la estructura de los fragmentos.
El Algoritmo 4 presenta el proceso para la obtención del costo de desempeño, sin embargo, el valor más relevante es el costo de desempeño de cada atributo por sitio, que se muestra en la línea 21, ya que al determinar el conjunto de atributos que componen cada fragmento (línea 15) se determina el costo de desempeño actual de cada fragmento. El valor de retorno, *costoDesempeño*, es la acumulación de costos de todos los fragmentos analizados por el vigilante-fragmentador.

Algoritmo 4. Costo de desempeño

```
1  input: logs[], posiciónActualArchivo, posiciónActualCarácter, fragmentosObservados[], miIP
2  output: costoDesempeño
3  for i from posiciónActualArchivo to size of logs[] do
4      while posiciónActualCarácter < size of logs[i] do
5          líneaActual ← leer línea desde posiciónActualCarácter
6          if líneaActual contiene {("find" or "select") or "delete" or "insert" or "update"} then
7              for j from 0 to size of fragmentosObservados[] do
8                  if líneaActual contiene fragmentosObservados[j].nombre then
9                      valorTipodeOperación ← {1 si es ("find" or "select"), 2 si es ("delete" or "insert"),
10                                             3 si es "update"}
11                     IPoperación ← obtener IP desde donde la operación se desempeñó
```

```
12                  if IPoperación is not miIP then
13                      valorTipodeOperación ← ValorTipodeOperación*2
14                  nJoins ← (obtener ocurrencias de {"join" or "$lookup"} en líneaActual)+1
15                  atributosActuales[] ← fragmentosObservados[j].Atributos[]
16                  for k from 0 to size of atributosActuales[] do
17                      if líneaActual contiene atributosActuales[k].nombre
18                      or líneaActual no contiene {"projection" or "*"} then
19                          valorActual ← valorTipodeOperación * nJoins * atributosActuales[k].tamañobytes
20                          costoDeAtributoPorSitioActual ← atributosActuales[k].costo de atributo por sitio
21                          atributosActuales[k].colocar costo de atributo por sitio(IPoperación,
22                                                                  costoDeAtributoPorSitioActual+valorActual)
23  costoDesempeño ← 0
24  for m from 0 to size of fragmentosObservados[] do
25      atributosActuales[] ← fragmentosObservados[m].atributos[]
26      for n from 0 to size of atributosActuales[] do
27          sitiosUsados[] ← costo de sitios que ocupan este atributo
28          foreach u ← sitiosUsados[] do
29              costoDesempeño ← costoDesempeño +u.costo
30  actualizar posiciónActualArchivo y posiciónActualCarácter
31  return costoDesempeño
```

Figura 3.26. Algoritmo para la obtención del costo de desempeño de la fragmentación vertical

Los parámetros *posiciónActualArchivo* y *posiciónActualCarácter* son indicadores útiles para no repetir el análisis de los archivos de registro y solo escrutar las nuevas líneas que llegan con las operaciones realizadas (líneas 3, 4 y 30). La línea 5 obtiene la fila actual del archivo *logs[i]*.

Las líneas 6 y 8 funcionan como filtros para que solo se analicen las líneas que contienen el nombre de la tabla o fragmento y los nombres de las operaciones de interés en esta investigación. La línea 7 representa la iteración de los fragmentos analizados, que, en el caso de la fragmentación estática, es solo la tabla original. La línea 9 determina el valor de la operación, teniendo en cuenta que las operaciones de lectura en MongoDB están bajo el comando *find*. La línea 13 suma el costo remoto y la línea 14 el costo de reuniones. Los atributos del fragmento (tabla original en la fragmentación estática) se obtienen en la línea 15. Se analiza, para cada atributo, si la línea obtenida lo contiene o si incluye todos los atributos (asterisco para SGBDs relacionales o ausencia de proyección en MongoDB). La línea 19 se encarga de aplicar el resto de variables del modelo de costo, almacenando el resultado en *valorActual*. El valor se agrega al costo del atributo en ese sitio en la línea 21. Las líneas 23 a 29 obtienen el resultado del costo de desempeño de todos los fragmentos analizados bajo este vigilante-fragmentador.

La fragmentación vertical contempla una característica de interés: la afinidad de los atributos. Las operaciones realizadas en la base de datos suelen utilizar conjuntos de atributos, la afinidad es la relación que existe entre estos atributos bajo un mismo modelo de costos. Para determinar la afinidad entre los atributos se utiliza el Algoritmo 5.

Algoritmo 5. Algoritmo de afinidad

```
1  input: atributoSitioCosto[][], conjuntoAtributos[], sitios[]
2  output: esquema
3  atributosSitioCostos[][] ← matriz vacia
4  for i from 0 to size of conjuntoAtributos[] do
5      for j from 0 to size of sitios[] do
6          sumaSitio ← 0
7          atributos ← conjuntoAtributos[i]
8          if atributoSitioCosto[cada atributo en atributos][sitios[j]]>0 then
9              for k from 0 to size of atributos do
10                 sumaSitio ← sumaSitio+
11                             atributoSitioCosto[obtener de atributos la posición k][sitios[j]]
12             atributosSitioCostos[atributos][sitios[j]] ← sumaSitio
13         else
14             atributosSitioCostos[atributos][sitios[j]] ← 0
15 aEliminar ← lista vacia
16 foreach arr ← atributosSitioCostos do
17     hayMultimedia ← false
18     esSoloPK ← false
19     for i from 0 to size of arr and !hayMultimedia do
20         if arr en la posición i es multimedia then
21             hayMultimedia ← true
22         if arr en la posición i es llave primaria then
23             esSoloPK ← true
24     if hayMultimedia=true or (esSoloPK=true and size of arr=1) then
25         agregar arr a aEliminar
26 foreach arr ← aEliminar do
27     remover arr de atributosSitioCostos
28 atributosSuma ← lista vacia
29 foreach arr ← atributosSitioCostos do
30     sum ← 0.0
31     foreach ip ← obtener ips de atributosSitioCostos[arr] do
32         sum ← sum+atributosSitioCostos[arr][ip]
33     agregar tupla en atributosSuma de arr y sum
34 ordenar por costo atributosSuma
35 filtrados ← lista vacia
36 foreach o ← atributosSuma do
37     yaLoContiene ← false
38     for j from 0 to size of lista de atributos en o do
39         if la posición j en lista de atributos en o se
40         encuentra en filtrados then
41             yaLoContiene ← true
42     if yaLoContiene!=false then
43         agregar o a filtrados
44 si hay atributos faltantes, agregarlos en un nuevo conjunto a filtrados
45 esquema ← lista vacia
46 foreach listaAtt ← filtrados do
47     tupla ← objeto tupla
48     agregar listaAtt a los atributos de tupla
49     agregar el costo de listaAtt en filtrados a costo en tupla
50     agregar false a multimedia en tupla
51     agregar tupla a esquema
52 agregrar fragmentos multimedia a esquema
53 foreach tc ← esquema do
54     costoFragmento ← 0.0
```

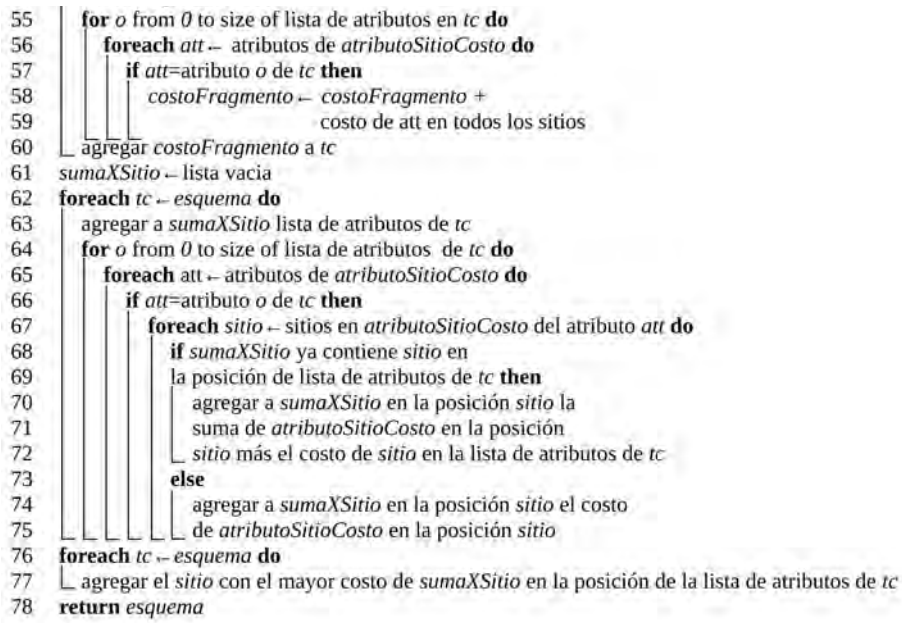


Figura 3.27. Algoritmo para la obtención de la afinidad de los atributos y la determinación del esquema de la fragmentación vertical

El Algoritmo 5 muestra en su primera actividad la creación de la matriz del costo de atributos por sitio, la cual suma el costo de los atributos que presentan actividades en un mismo sitio, si uno de los atributos del conjunto no presenta actividades se coloca 0, representando que no hay relación en ese sitio con esos atributos (líneas 1-15).

La siguiente actividad presenta la creación de la matriz con la suma del costo de los atributos, la cual concentra los costos independientemente del sitio al que pertenecen (líneas 28-33), para realizar la tercera actividad, que elimina los conjuntos de atributos multimedia y conjuntos con solo la clave primaria y ordena los restantes por costo (líneas 16-25 y 34). En la siguiente actividad, una vez ordenada la matriz anterior, se eliminan los conjuntos que contienen atributos repetidos, respetando los atributos que se encuentran primero (más costosos) (líneas 36-43). La quinta actividad agrega el conjunto de los atributos faltantes y los conjuntos multimedia (líneas 44 y 52). La siguiente actividad recalcula los costos de cada conjunto producido, como en la primera actividad, pero sin considerar si todos los atributos tienen actividad en cada sitio, es decir, incluso si no todos los atributos de un conjunto presentan actividad en un determinado sitio, el costo se suma a los atributos que sí lo tienen

(líneas 53-60). Finalmente, se genera el esquema de fragmentación vertical, determinando el sitio más costoso para cada conjunto de atributos (líneas 62-77).

La fragmentación híbrida se lleva a cabo aplicando los flujos de trabajo de la fragmentación vertical y horizontal, mezclando los esquemas resultantes y determinando cuál es menos costoso, el horizontal-vertical o el vertical-horizontal. Los esquemas horizontal-vertical y vertical-horizontal se diferencian en sus asignaciones, ya que ambos esquemas producen los mismos fragmentos, pero se asignan a sitios diferentes. La fragmentación horizontal-vertical respeta las asignaciones horizontales y la vertical-horizontal respeta las asignaciones verticales.

El Algoritmo 6 muestra el pseudocódigo para obtener el costo de los fragmentos, previamente propuestos al mezclar los esquemas horizontal y vertical.

Algoritmo 6. Costo de desempeño híbrido

```
1  input: log[], esquemaNuevo[]
2  output: esquemaNuevo[]
3  for i from 0 to size of logs[] do
4      actual ← log[i]
5      ipLinea ← obtener la dirección del lugar a donde se ejecutó la operación actual
6      valorAsignadoOperacion ← 1 si es lectura, 2 si es creación o eliminación y 3 si es actualización
7      if actual contiene el nombre de la tabla a fragmentar then
8          nReuniones ← 0
9          atributosOpe[] ← buscar atributos utilizados en la operación actual
10         fragmentosResuelvenAtributos[] ← lista vacía
11         foreach fragmento ← esquemaNuevo[] do
12             seUsa ← false
13             foreach l ← atributos de fragmento do
14                 if atributosOpe contiene l then
15                     seUsa ← true
16         if seUsa = true then
17             agregar fragmento a fragmentosResuelvenAtributos[]
18         predicado ← obtener predicado de la operación actual
19         tuplasDelPredicado ← obtener el número de tuplas que representa el predicado
20         fragmentosResuelvenPredicados ← lista vacía
21         if tuplasDelPredicado > 0 then
22             foreach fragmento ← fragmentosResuelvenAtributos[]
23                 tuplas ← obtener el número de tuplas del predicado y el predicado del fragmento
24                 if tuplas > 0 then
25                     agregar a fragmentosResuelvenPredicados el fragmento
26         nJoins ← tamaño de fragmentosResuelvenPredicados
27         siHay ← false
28         foreach fragmento ← fragmentosResuelvenPredicados do
29             if el sitio de asignación vertical del fragmento = ipLinea then
30                 siHay ← true
```

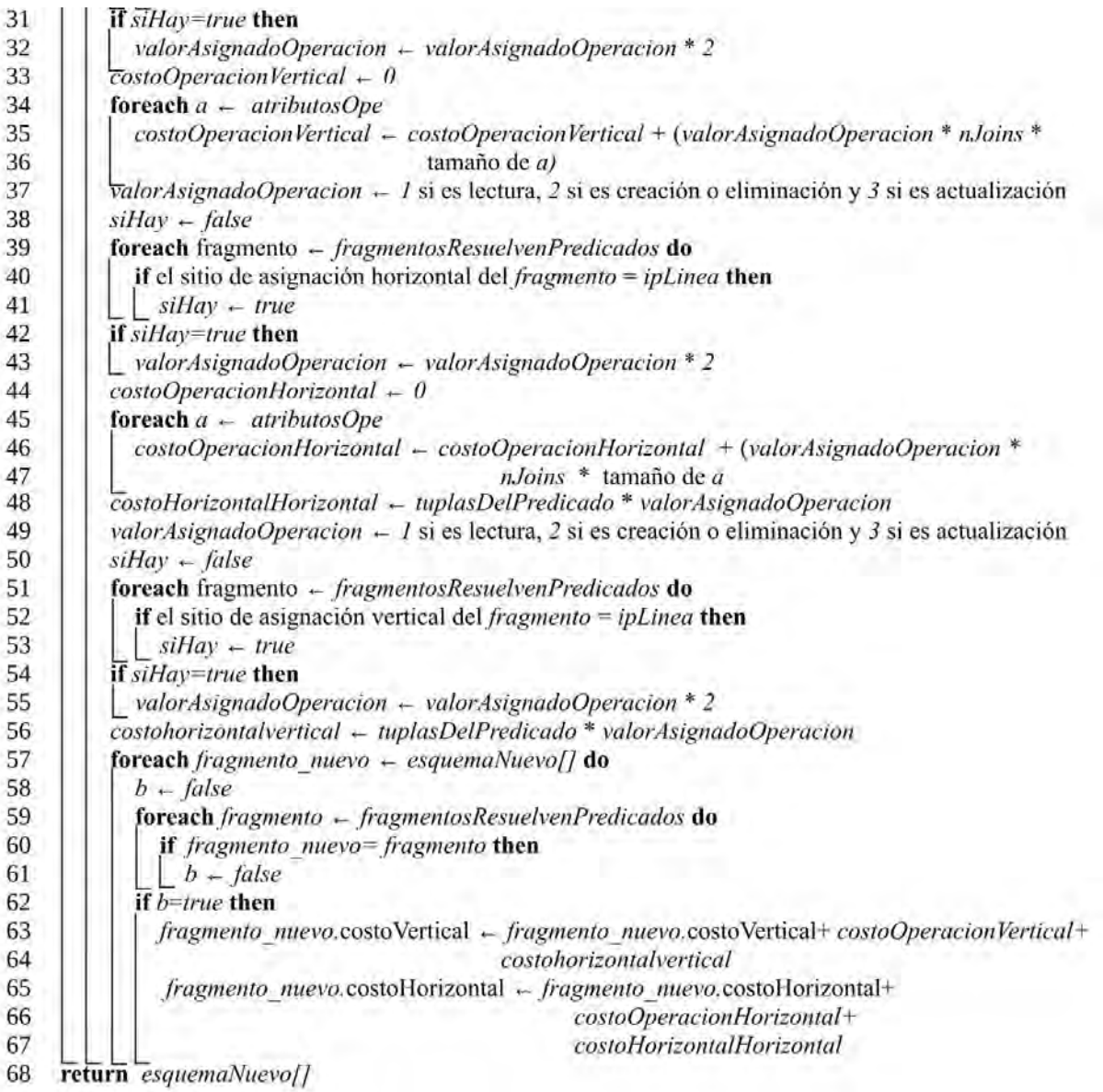


Figura 3.28. Algoritmo para la obtención del costo de fragmentos híbridos

Los fragmentos que recibe el Algoritmo 6 en el arreglo *esquemaNuevo* son los producidos por la mezcla de los esquemas horizontal y vertical, cada fragmento contendrá el costo de los dos sitios bajo el modelo de costos de la fragmentación híbrida. Inicialmente, los fragmentos contienen ambos sitio (horizontal y vertical), los cuales se obtuvieron mediante cada flujo de trabajo por separado. El objetivo del Algoritmo 6 es determinar el costo horizontal y el costo vertical de cada fragmento. Al finalizar, mediante estos costos, se determinan las asignaciones que se deben respetar (los sitios menos costosos).

Cada operación se analiza, obteniendo como en las fragmentaciones anteriores, el valor del tipo de operación y la IP (líneas 5 y 6). Las líneas 11-17 determinan los fragmentos de *esquemaNuevo* que son capaces de resolver la operación de manera parcial o total en sus atributos. De ese arreglo (*fragmentosResuelvenAtributos*) se determinan los fragmentos que también resuelven la operación de manera parcial o total en su predicado (líneas 21-25). Al obtener los fragmentos que resuelven la operación en sus atributos y en sus predicados, se establece si la operación se resuelve de manera remota o local frente al sitio vertical (líneas 28-32). La variable *costoOperacionVertical* contiene el costo obtenido mediante el modelo de costos de la fragmentación vertical frente al sitio vertical (línea 35). La línea 37 recalcula el *valorAsignadoOperacion,* ya que se modificó en la línea 32. La variable *costoOperacionHorizontal* contiene el costo obtenido mediante el modelo de costos de la fragmentación vertical frente al sitio horizontal (línea 46). La variable *costoHorizontalHorizontal* contiene el costo obtenido mediante el modelo de costos de la fragmentación horizontal frente al sitio horizontal del fragmento (línea 48). La línea 49 nuevamente recalcula el *valorAsignadoOperacion* modificado en la línea 43. La variable *costoHorizontalVertical* contiene el costo obtenido mediante el modelo de costos de la fragmentación horizontal frente al sitio vertical del fragmento (línea 56). Las líneas 57-67 presentan la asignación de los costos obtenidos a los fragmentos relacionados en la variable *esquemaNuevo*, para que el mismo esquema que se recibió, se entregue con los costos de cada asignación.

El esquema híbrido se caracteriza por contener fragmentos que se definen por un subconjunto de atributos de la tabla original y un subconjunto de tuplas determinadas por un predicado. El traslape de predicados híbridos tiene el mismo objetivo que el traslape de predicados horizontales, el cual es respetar en el traslape a los predicados más costosos. Para comprender el traslape de predicados híbridos, la Figura 3.29 muestra un ejemplo del traslape de predicados horizontales.

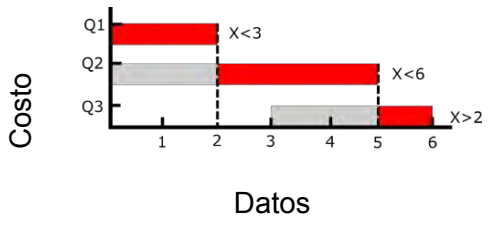


Figura 3.29. Traslape de predicados horizontales.

La Figura 3.29 presenta un ejemplo del traslape de predicados horizontales, el cual consiste en priorizar los predicados con mayor costo, aplicándolos sin importar qué predicados menos costosos ocupen el mismo o un conjunto inferior de tuplas dentro de estos mismos. El predicado *X<3* es el más costoso y se aplica respetando su dominio por completo, mientras que el predicado *X< 6* solo es respetado en la parte que no se traslapa con el anterior. Por último, el predicado *X>2* es el menos costoso y solo se respeta la parte restante que no se traslapa con todos los anteriores. Para lograr este objetivo, los predicados más costosos se niegan en predicados menos costosos, es decir, para la definición del predicado *X<6* , se considera *!(X<3) && (X<6)* y para la definición del predicado *X>2*, se considera *!(X<3) && !(x<6) && (x>2)*. El traslape de predicados híbridos cumple el mismo principio, sin embargo, considera una característica distintiva, los predicados se repiten en la definición de los fragmentos, es decir, un mismo predicado define a dos o más fragmentos con diferentes atributos. En el traslape de predicados híbridos se niegan los predicados menos costosos, pero sin incluir negaciones repetidas. XAMANA toma en cuenta MongoDB para todas las fragmentaciones que se realizan, y por esta razón, se incluye la notación de predicados en JSON, que es la utilizada por este gestor NoSQL.

CAPÍTULO 4. RESULTADOS

Este capítulo describe los resultados obtenidos al aplicar el diseño y llevar a cabo evaluaciones que validen el enfoque. Se presenta una aplicación Web llamada XAMANA, encargada de aplicar el flujo de trabajo de los tres tipos de fragmentación.

4.1. Resultados del análisis de trabajos relacionados

Después de haber mostrado la breve descripción de cada artículo en el Capítulo 2 se presentan los resultados del análisis mediante gráficas y tablas, que reflejan cifras de los 80 trabajos obtenidos. De esta manera se observa con mayor claridad otro enfoque de los datos.

La Figura 4.1 muestra una gráfica de barras con la distribución de los artículos analizados según el año de publicación. Se observa que en el año 2019 se concentra la mayor cantidad de artículos. Se observa que, de los nueve artículos, tres son de Springer (Costa, Costa, & Santos, 2019; Olma et al., 2019; Durand et al., 2019), la editorial con el mayor número de artículos, como se observa en las Figuras 4.2 y 4.3; dos pertenecen a ACM (Schreiner et al., 2019; Mourão & Magalhães, 2019); dos a IEEE (Santos, Ghita, & Masala, 2019; Sharify et al., 2019), y dos que pertenecen a la categoría "Otras" (Le, Kantere & Orazio, 2019; Sharma & Bawa, 2019).

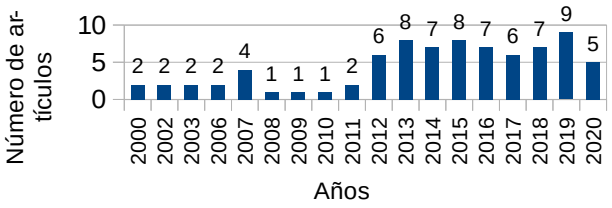


Figura 4.1. Número de artículos por año de publicación

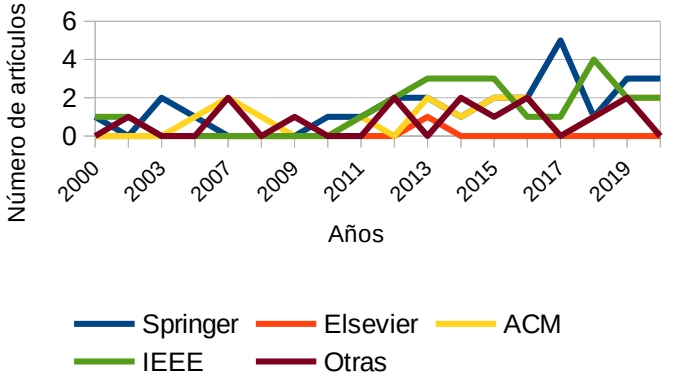


Figura 4.2. Número de artículos por año de publicación por cada editorial

La Figura 4.3 muestra una gráfica de barras que relaciona el número de artículos con la editorial; la mayoría de los artículos se concentran en la editorial Springer. Los artículos de Springer abordaron principalmente la fragmentación vertical (Pérez et al., 2000; Fung, Leung & Li, 2003; Fung, Karlapalem & Li, 2003; Rodríguez y Li, 2011a; Rodríguez-Mazahua et al., 2016; Durand et al., 2019; Costa, Costa & Santos, 2019; Amer, 2020) y horizontal (Saad et al., 2006; Hauglid, Ryeng & Nørvåg, 2010; Liroz-Gistau et al., 2012; Bellatreche et al., 2013; Liroz-Gistau et al., 2013; Rodríguez-Mazahua et al., 2014; Abdel Raouf, Badr & Tolba, 2016; Elghamrawy, 2016; Elghamrawy & Hassanien, 2017; Olma et al., 2019), 18 de 26 artículos. Solo un artículo considera tanto la fragmentación como las consultas basadas en contenido (Saad et al., 2006).

Figura 4.3. Número de artículos por editorial

En la Fig. 4.4 se observa una gráfica de barras que muestra el número de artículos por tipo de fragmentación. La fragmentación más empleada en los trabajos analizados es la horizontal. La categoría "Otro" muestra el número de artículos que utilizan otro tipo de fragmentación que no es ninguno de los otros tres, como la fragmentación de documentos en Cuzzocrea et al. (2009) y Torjmen, Pinel-Sauvagnat & Boughanem (2008), la fragmentación de videos en Mettes et al. (2015), la fragmentación de rejilla en Le, Kantere & Orazio (2019), entre otros. De los trabajos que abordaron la fragmentación horizontal, cuatro también contemplaron consultas basadas en contenido (Fasolin et al., 2013; Getahun et al., 2007a; Getahun et al., 2007b; Saad et al., 2006).

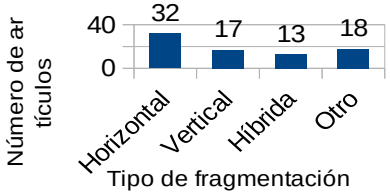


Figura 4.4. Número de artículos por tipo de fragmentación

En la Tabla 4.1 se muestran los costos que se encontraron a lo largo del análisis y los artículos que los utilizan. Diferentes artículos contemplan más de un tipo de costo, es por esta razón que el total de artículos en la segunda columna no es el total de artículos que se analizaron. El costo de trasporte, de acceso, de almacenamiento y de ejecución son los más utilizados, con un total de 63 menciones en los artículos.

Tabla 4.1. Costos obtenidos en la etapa de análisis.

Costo	Artículos
Costo de accesos a datos irrelevantes	Rodríguez & Li (2011); Rodríguez-Mazahua et al. (2014); Rodríguez-Mazahua et al. (2016); Cuzzocrea et al. (2009).
Costo de procesamiento de consultas	Ma et al. (2006); Wang et al. (2014); Kulba & Somov (2020); Hung & Huang (2012); Torjmen et al. (2008).
Costo de almacenamiento	Zhao et al. (2015); Ma et al. (2006); Ma et al. (2007); Abdalla & Amer (2012); Baron & Iacob (2014); Zar Lwin & Naing (2018); Chbeir & Laurent (2010); Kulba & Somov (2020); Durand et al. (2018); Hung & Huang (2012); Santos & Masala (2018); Santos et al. (2019).
Costo de transporte	Castro-Medina et al. (2020); Rodríguez & Li (2011); Gu et al. (2012); Alagiannis et al. (2014); Pinto & Torres (2002); Ma et al. (2006); Ma et al. (2007); Bellatreche et al. (2011); Derrar et al. (2013); Baron & Iacob (2014); Rodríguez-Mazahua et al. (2014); Khan (2016); Zar Lwin & Naing (2018); Al-Kateb et al. (2016); Rodríguez-Mazahua et al. (2016); Schreiner et al. (2018); Pinnecke et al. (2020); Sleit et al. (2007); Hung & Huang (2012); Le at al. (2019).
Costo de acceso	Pérez et al. (2000); Fung, Karlapalem & Li (2002); Fung, Leung & Li (2003); Fung, Karlapalem & Li (2003); Rodríguez & Li (2011); Alagiannis et al. (2014); Zhao et al. (2015); Rodríguez-Mazahua et al. (2016); Sharify et al. (2019); Amer (2020); Pinto & Torres (2002); Hauglid et al. (2010); Derrar et al. (2013); Serafini et al. (2016); Olma et al. (2019); Pinnecke et al. (2020); Hung & Huang (2012).
Costo de ejecución	Fung, Karlapalem & Li (2002); Fung, Leung & Li (2003); Fung, Karlapalem & Li (2003); Alagiannis et al.

Costo	Artículos
	(2014); Li & Gruenwald (2014); Rodríguez-Mazahua et al. (2016); Durand et al. (2019); Sharify et al. (2019); Schroeder et al. (2020); Tekli et al. (2007); Bellatreche et al. (2013); Derrar et al. (2013); Herrmann et al. (2014); Fetai et al. (2015); Jindal & Dittrich (2012); Chen et al. (2015); Rodríguez-Mazahua et al. (2016).
Costo de actualización	Tekli et al. (2007); Abdel et al. (2016); Chbeir & Laurent (2010); Durand et al. (2018); Pinnecke et al. (2020); Sleit et al. (2007).
Costo computacional de creación de fragmentos	Saad et al. (2006); Lim et al. (2013); Fetai et al. (2015); Jindal & Dittrich (2012).
Costo de red	Bellatreche et al. (2011).
Costo computacional	Tekli et al. (2007); Al-Kateb et al. (2016).
Costo de comunicación	Durand et al. (2019); Amer (2020); Schroeder et al. (2020); Hauglid et al. (2010); Abdalla & Amer (2012); Baron & Iacob (2014); Abdel et al. (2016); Wang et al. (2014); Chen et al. (2015); Kulba & Somov (2020); Hung & Huang (2012); Santos et al. (2019).
Costo de mantenimiento	Bellatreche et al. (2013).
Costo de replicación	Turcu et al. (2016).
Costo de movimiento	Turcu et al. (2016).
Costo de tiempo de respuesta	Rodríguez-Mazahua et al. (2016); Li & Gruenwald (2014).
Costo de accesos locales	Baron & Iacob (2014); Hung & Huang (2012).
Costo de uso de atributos	Li & Gruenwald (2013).
Costo de tipo de operación	Li & Gruenwald (2013).
Costo de carga	Olma et al. (2019).
Costo de migración	Serafini et al. (2016).
Costo de distribución	Abdel et al. (2016).
Costo de asignación	Vogt et al. (2018).
Costo de sobrecarga	Vogt et al. (2018).

En la Figura 4.5 se muestra el número de artículos por SGBD. La mayoría de los artículos no mencionan o no usan SGBD para llevar a cabo su investigación. Se observa que los gestores más utilizados son Oracle, MongoDB, PostgreSQL y Cassandra.

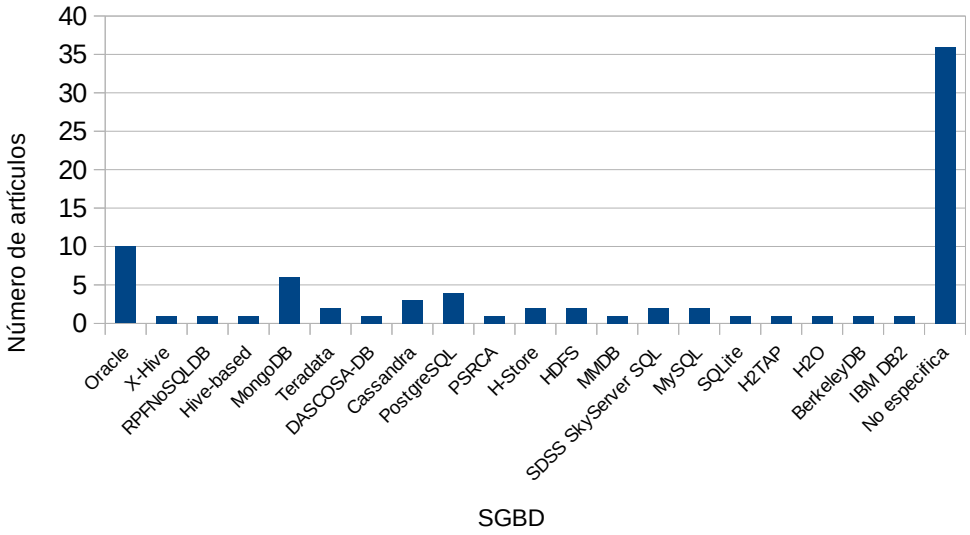


Figura 4.5. Número de artículos por SGBD

En la Tabla 4.2 se muestran los *benchmarks* más utilizados. De igual manera, la columna derecha no refleja el número de artículos totales, ya que muchos artículos utilizan más de un *benchmark.* El *benchmark* TPC-H es el más utilizado.

Tabla 4.2. *Benchmarks* obtenidos en la etapa de análisis.

Benchmark	Artículos
XWeb	Cuzzocrea et al. (2009).
OO7	Ma et al. (2006); Tekli et al. (2007).
THUMOS	Mettes et al. (2015).
TRECVID	Mettes et al. (2015).
CRM	Le et al. (2019).
TPC-H	Castro-Medina et al. (2020); Rodríguez et al. (2013); Gu et al. (2012); Li & Gruenwald (2013); Li & Gruenwald (2014); Durand et al. (2019); Costa et al. (2019); Lim (2013); Herrmann et al. (2014); Jindal & Dittrich (2012); Al-Kateb et al. (2016); Le et al. (2019).
SSB	Costa et al. (2019); Bellatreche et al. (2011); Jindal & Dittrich (2012).
Linkbench	Nguyen & Lee (2017).
YCSB	Elghamrawy (2016); Elghamrawy & Hassanien (2017); Schreiner et al. (2018); Nguyen & Lee (2017).
TPC-W	Schreiner et al. (2018); Turcu et al. (2016).

Benchmark	Artículos
SPECWeb2009	Sauer & Hao (2015).
TPC-C	Fetai et al. (2015); Serafini et al. (2016); Wang et al. (2014); Schreiner et al. (2018); Pinnecke et al. (2020); Turcu et al. (2016).
AuctionMark	Turcu et al. (2016).
EPinions	Turcu et al. (2016).
Re-Twis	Turcu et al. (2016).
APB-1	Bellatreche et al. (2013); Derrar et al. (2013).
Sloan Digital Sky	Alagiannis et al. (2014); Zhao et al. (2015); Liroz-Gistau et al. (2012); Liroz-Gistau et al. (2013).
FIO	Khan et al. (2018).
Fashion-MNIST	Lu et al. (2018).
SIFT	Lu et al. (2018).
GloVe	Lu et al. (2018).
NoBench	Sharify et al. (2019).
OLTP-Bench	Schreiner et al. (2019).
BSBM	Schroeder et al. (2020).

4.2. Fragmentación horizontal multimedia y CBIR

Para obtener los resultados de este enfoque se utilizó una base de datos multimedia denominada HITO (Historia del Instituto Tecnológico de Orizaba) (Arauz et al. 2020), implementada en MongoDB con más de 1.5 GB de almacenamiento. Las evaluaciones de la fragmentación horizontal multimedia y CBIR se llevaron a cabo en dos sitios. El primer sitio utiliza Ubuntu 20.04.3 LTS como sistema operativo, 4 GB de RAM y un procesador AMD A6 con una frecuencia de reloj de 2,1 GHz. El segundo sitio usa Windows 10 como sistema operativo, 6 GB de RAM y un procesador Intel I7 con una frecuencia de reloj de 3 GHz. La base de datos centralizada se encuentra en el primer sitio. Se utilizó la colección *multimedia_records* que se describe en la Tabla 4.3. Las operaciones desempeñadas se obtuvieron mediante SimulQuery, un simulador de operaciones desarrollado para este trabajo de investigación.

Tabla 4.3. Atributos de la tabla *multimedia_records*

Nombre del atributo	Tipo de dato
descripción	String
tipo	String

Nombre del atributo	Tipo de dato
imagen	String
validado	String
_id	String
descriptor	Array
nombre	String

El conjunto de operaciones realizadas se analizan bajo el esquema de la matriz de costo de operaciones basada en el modelo de costos multimedia alfanumérico. Esta matriz alberga de cada operación, el predicado, el costo y el sitio desde donde se ejecutó. El costo de cada operación se obtiene mediante el modelo de costos de la fragmentación horizontal. La Tabla 4.4 presenta la matriz de costo de operaciones de esta evaluación, considerando un conjunto de operaciones reducido para observar el funcionamiento y las ventajas de forma especifica.

Tabla 4.4. Matriz de costo de operaciones para desempeñar la fragmentación horizontal multimedia.

Predicado	192.168.100.3	192.168.100.20
{"descripcion":"..."}	262	0
{"nombre":"LEYBO"}	2	0
{"nombre":"17NOV89"}	2	0
{"nombre":{"$ne":"VERMONFE"}}	3738	0
{"descripcion":"."}	5220	10440
{"nombre":{"$eq":"CARICATURASITO2005_13~2"}}	1	1
{"nombre":{"$ne":"6OCT94"}}	1869	0
{"nombre":"RAMNAVML"}	2	0
{"tipo":{"$eq":"personal"}}	2622	13984
{"tipo":{"$eq":"event"}}	2172	5792
{"descripcion":{"$eq":"Andrade Meza María Antonieta"}}	0	0
{"nombre":{"$eq":"Un nuevo edificio"}}	6	0
{"nombre":{"$ne":"CARICATURASITO2005_30~3"}}	1869	0
{"nombre":{"$eq":"Morramro"}}	2	0
{"descripcion":"Carrera Flores Rafael"}	0	4
{"nombre":"PRENA3-1"}	2	0
{"descripcion":{"$ne":"."}}	2825	1130
{"descripcion":{"$eq":"Caricatura a mano"}}	205	1230
{"nombre":{"$eq":"CARICATURASITO2005_10"}}	1	0

Predicado	192.168.100.3	192.168.100.20
{"tipo":{"$ne":"equipment"}}	5298	14128
{"nombre":"TERORTBA"}	0	4
{"descripcion":{"$eq":"."}}	3915	10440
{"_id":{"$eq":{"$oid":"615e997af0f0e30bc46132b0"}}}	0	2
{"validado":{"$ne":"null"}}	14960	33660
{"nombre":{"$ne":"XXXVII-6"}}	0	7476
{"tipo":{"$ne":"event"}}	0	27504
{"_id":{"$ne":{"$oid":"615e99b9f0f0e30bc46133b7"}}}	0	11214
{"nombre":{"$eq":"conc3.1"}}	0	4
{"descripcion":{"$ne":"Alumnos realizando una práctica en el Laboratorio Diesel del Taller de Máquinas de Combustión Interna, 1980"}}	1870	0
{"nombre":{"$eq":"menespbe"}}	4	0
{"nombre":{"$ne":"BECOLICE"}}	3738	0
{"nombre":{"$eq":"Depor15"}}	0	4
{"_id":{"$ne":"IMG631"}}	0	7476
{"nombre":{"$ne":"NAVARTCO"}}	0	3738
{"nombre":{"$eq":"Diploma2"}}	0	4
{"tipo":"event"}	1448	0
{"nombre":{"$ne":"CARICATURASITO2005_9"}}	0	7476
{"nombre":{"$eq":"RODBRELA"}}	0	4
{"_id":{"$ne":{"$oid":"615e980bf0f0e30bc461319c"}}}	5607	0
{"nombre":"CARICATURASITO2005_32~3"}	0	4
{"descripcion":{"$eq":"..."}}	262	0
{"_id":{"$oid":"615e999df0f0e30bc4613303"}}	3	0
{"nombre":{"$eq":"ALICIA"}}	0	4
{"validado":"null"}	0	0
{"_id":{"$ne":"IMG96"}}	0	3738
{"tipo":{"$ne":"building"}}	3404	10212
{"nombre":{"$eq":"1989"}}	3	0
{"validado":{"$eq":"null"}}	0	0
{"tipo":{"$ne":"personal"}}	0	3984
{"nombre":"EN198xxxxx-019(JMH)"}	2	0
{"nombre":{"$eq":"MADCERJE"}}	0	8
{"nombre":{"$ne":"28ENE94~1"}}	3738	0
{"nombre":"LabMec17"}	2	0
{"nombre":{"$ne":"ed-nfr02"}}	1869	0

Predicado	192.168.100.3	192.168.100.20
{"tipo":"personal"}	3496	3496
{"_id":{"$eq":{"$oid":"615e97a2f0f0e30bc4613124"}}}	0	4

La Tabla 4.4 muestra el costo de cada predicado por sitio, siguiendo el flujo de trabajo de la fragmentación horizontal. Para obtener la tabla ALP se suma cada predicado, agrupando el costo por cada atributo. El resultado se observa en la Tabla 4.5.

Tabla 4.5. Tabla ALP de los predicados presentados en la Tabla 4.4.

Atributo	Costo
descripcion	37803.0
tipo	97540.0
validado	48620.0
_id	28044.0
nombre	35576.0

Los atributos que no se incluyen en la Tabla 4.5 no presentan operaciones en el archivo *log*. El atributo más costoso es *tipo*, y este determina el esquema que se produce. Todos los predicados relacionados con *tipo* de la Tabla 4.4 fragmentan la relación. La Figura 4.6 muestra el esquema producido bajo este flujo de trabajo.

Fragmento	Predicado	Predicado traslape	Sitio	Número de tuplas	Valor de desempeño
multimedia_records_f0	{"tipo": {"$ne":"event"}}	{$and:[{"tipo": {"$ne":"event"}}]}	192.168.100.20	1146	27504.0
multimedia_records_f1	{"tipo": {"$ne":"equipment"}}	{$and:[{$or:[{"tipo": {$not: {"$ne":"event"}}}]}, {"tipo": {"$ne":"equipment"}}]}	192.168.100.20	724	19426.0

Figura 4.6. Esquema producido mediante XAMANA con el flujo de trabajo de la fragmentación horizontal.

Se observa en el esquema producido la columna *Predicado*, la cual se obtuvo mediante la Tabla 4.4, mientras que la columna *Predicado traslape* se crea mediante el traslape de predicados horizontales, el cual se describe en la Figura 3.34. El traslape de predicados

horizontales, cuando los fragmentos producidos no incluyen todas las tuplas de la relación original, crea un nuevo fragmento con el predicado que niega a todos los demás, para que de este modo, el esquema incluya a toda la relación y se considere una fragmentación válida. Ambos fragmentos mostrados en la Figura 4.6 incluyen todas las tuplas de la relación original y por esta razón, el fragmento con el predicado que niega a todos los demás no se incluye.

El nuevo esquema, una vez aplicado, se comparó con el esquema anterior respecto al tiempo de ejecución de las consultas y lo redujo en un 32.1%. El nuevo esquema, ademas de reducir los tiempos de ejecución, se adapta de mejor manera a la carga de trabajo, ya que las operaciones se realizan bajo fragmentos más pequeños frente a la tabla original.

Para llevar a cabo la nueva evaluación de la fragmentación horizontal CBIR estática y dinámica, se ejecutaron las operaciones que se muestran en la Tabla 4.6 sobre la colección *multimedia_records*, que se muestra en la Tabla 4.3. De esta forma se obtiene un archivo *log* que servirá para obtener estadísticas de la operaciones antes de la primera fragmentación con XAMANA.

Tabla 4.6. Operaciones desempeñadas en la base de datos multimedia centralizada

Ítem	Operación	Sitio	Tiempo de ejecución en milisegundos
1	*db.multimedia_records.find({tipo:"equipment"})*	1	42
2	*db.multimedia_records.find({tipo:"event"})*	1	38
3	*db.multimedia_records.find({tipo:"equipment"})*	1	51
4	*db.multimedia_records.find({descripcion:"building"})*	1	6
5	*db.multimedia_records.find({tipo:"personal"})*	1	21
6	*db.multimedia_records.find({tipo:"event"})*	2	43
7	*db.multimedia_records.find({tipo:"event"})*	2	26
8	*Nguyen & Lee (2017).*	2	8
9	*db.multimedia_records.find({tipo:"equipment"})*	2	27
10	*db.multimedia_records.find({tipo:"event"})*	2	16

Los tipos de datos que se muestran en la Tabla 4.6 se interpretaron utilizando Java. Como se observa en la Tabla 4.6, las operaciones de lectura se realizan desde dos sitios diferentes. El sitio 1 representa la IP 192.168.200.5 y el sitio 2 representa la IP 192.168.200.30. La Figura

4.7 muestra la vista inicial de XAMANA solicitando el tipo de base de datos que se fragmentará. Como complemento de JavaServer Faces en la aplicación Web, se utilizó la biblioteca PrimeFaces para agregar una interfaz de usuario enriquecida de código abierto.

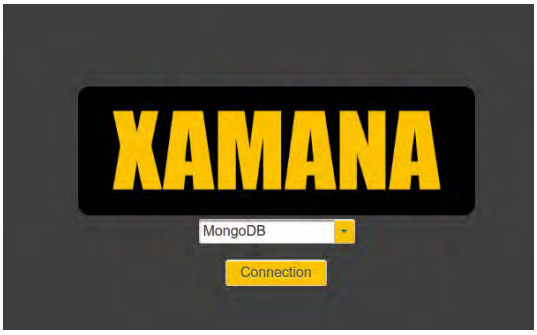


Figura 4.7. Página principal de la aplicación Web XAMANA

La vista de configuración de la conexión se muestra en la Figura 4.8. Los datos de la conexión al SGBD son los mencionados en las condiciones previas, los cuales deben ser iguales en todos los sitios de un usuario (excepto la IP). La dirección de la base de datos debe permitir la conexión remota desde la IP del servidor XAMANA. Los campos de nombre de usuario y contraseña no son obligatorios en las bases de datos donde la autenticación está deshabilitada. Al colocar los datos de conexión, se muestra automáticamente la lista de tablas encontradas.

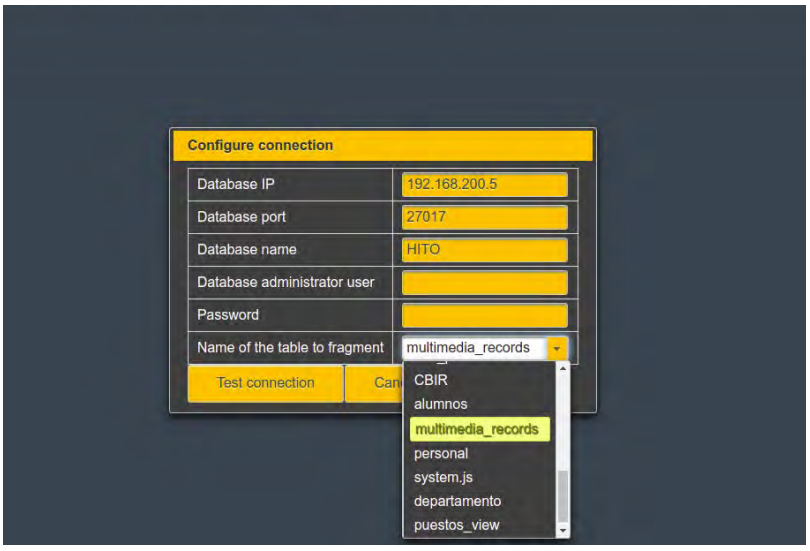


Figura 4.8. Configurar conexión remota de XAMANA para la base de datos del usuario

El estado de la conexión se muestra en el cuadro de diálogo de la Figura 4.9. Si los datos de la conexión no permiten la comunicación entre XAMANA y la base de datos del usuario, un mensaje informa que los datos deben ingresarse correctamente. Si los datos de conexión son correctos, la fragmentación debe configurarse en la siguiente vista.

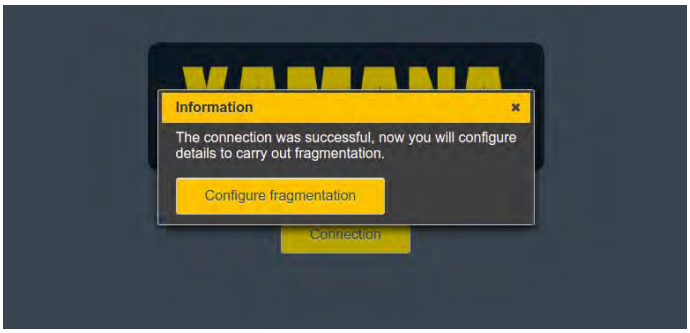


Figura 4.9. Estado de la conexión remota después de usar los datos para probar la conexión

La pantalla de configuración de la fragmentación se muestra en la Figura 4.10. En la Figura 4.10 se observa que se seleccionó la fragmentación horizontal, siendo uno de los tres tipos de fragmentación que permite la aplicación Web. La opción CBIR, cuando se selecciona, muestra el diálogo de la Figura 4.11. El siguiente campo solicitado por el formulario presenta la opción de agregar los archivos de registro donde se almacenan los registros de las operaciones ejecutadas anteriormente. Para esta evaluación, los umbrales se establecieron con los porcentajes mostrados.

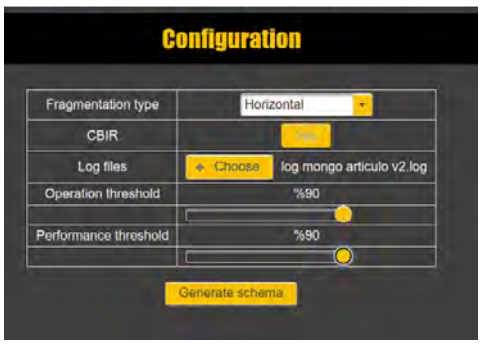


Figura 4.10. Configuración de la fragmentación que se llevará a acabo

En la Figura 4.11 se muestra la selección del atributo descriptor, mediante el cual se realizará el agrupamiento de tuplas y producción de fragmentos en la primera etapa de fragmentación horizontal. El atributo descriptor puede ser de cualquier tipo de dato, teniendo en cuenta que

debe formar grupos de tuplas que compartan los mismos valores. Es de interés mencionar que el atributo descriptor contemplado en este trabajo no comparte similitud con los descriptores visuales.

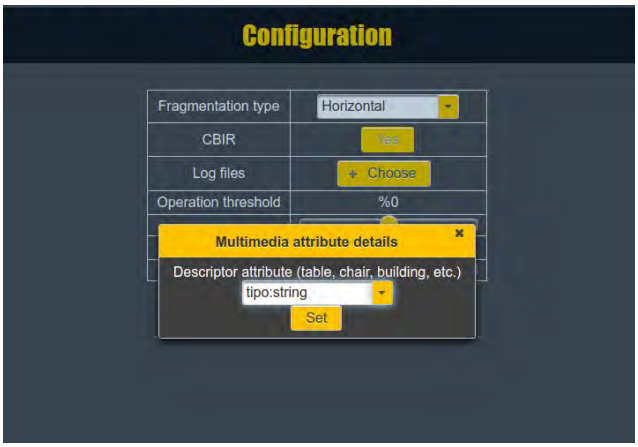


Figura 4.11. Configuración de la fragmentación que se llevará a cabo

La Figura 4.12 muestra la estructura final de los fragmentos. La primera columna indica el nombre del fragmento, la segunda columna contiene el conjunto que define el fragmento, la tercera columna representa el número de tuplas que contendrá cada fragmento, la cuarta columna muestra el porcentaje que representa cada fragmento con respecto a la tabla entera, la quinta columna contiene el valor de la operación y la sexta columna el valor de rendimiento.

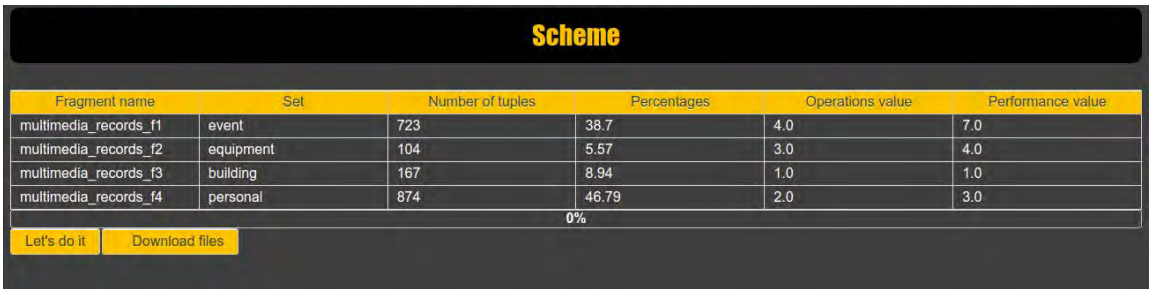


Figura 4.12. Esquema propuesto por el método de fragmentación horizontal

Los valores de las dos últimas columnas se obtienen del archivo de registro. El cálculo de los valores iniciales de las operaciones y el rendimiento se obtienen buscando en el registro todas las operaciones que contienen el conjunto de cada fragmento en sus predicados. El vigilante-

fragmentador contiene un mecanismo para guardar las operaciones realizadas y omitir el archivo de registro en futuras fragmentaciones. Los valores iniciales están ampliamente relacionados para determinar la ejecución de la siguiente fragmentación.

La vista de la estructura final de los fragmentos proporciona dos botones finales. El botón para aplicar el esquema físicamente sobre la tabla y el botón para descargar archivos para iniciar la fragmentación dinámica. Para el SGBD de PostgreSQL, además del vigilante-fragmentador, se ofrecen funciones y disparadores para mantener un diseño adecuado entre los fragmentos. Las funciones y los disparadores analizan los cambios en la estructura de la tabla original, es decir, agregan, eliminan y modifican atributos. Si alguna de estas operaciones se ejecuta en la tabla original, también se deben realizar en los fragmentos relacionados con esta tabla. De esta forma el diseño de los fragmentos se sincroniza con el diseño original, manteniendo adecuadamente la estructura de los atributos.

El enfoque de Rojas Ruiz (2018) para atender consultas basadas en contenido y buscar mediante el método de clasificación kNN los elementos multimedia más parecidos utilizando el descriptor SURF, realiza una búsqueda por consulta de todos los descriptores almacenados previamente. Estas operaciones de consulta modifican los patrones de acceso en el archivo *log* y provocan que las asignaciones no se deban basar en este. Cada fragmento horizontal obtenido mediante el flujo de trabajo CBIR en XAMANA se asigna al sitio local.

Luego de realizar la primera fragmentación e iniciar la fragmentación dinámica, se realizan nuevas operaciones desde los sitios 1 y 2 sobre los fragmentos observados que desencadenan una nueva fragmentación. La Tabla 4.7 muestra las operaciones realizadas por ambos sitios.

Tabla 4.7. Operaciones desempeñadas en la base de datos multimedia fragmentada.

Clave primaria del ítem buscado	Operación	Sitio	Nombre de fragmento
IMG1662	CR	1	*multimedia_records_f3*
IMG1875	R	1	*multimedia_records_f3*
IMG1581	RU	1	*multimedia_records_f1*
ObjectId("615e967df0f0e30bc461306e")	U	1	*multimedia_records_f4*
ObjectId("615e94d292bd5f6ac041f431")	C	1	*multimedia_records_f2*
ObjectId("615e94d892bd5f6ac041f432")	D	2	*multimedia_records_f2*

Clave primaria del ítem buscado	Operación	Sitio	Nombre de fragmento
ObjectId("615e9684f0f0e30bc461307c")	RR	2	*multimedia_records_f4*
ObjectId("615e967af0f0e30bc461306b")	RRR	2	*multimedia_records_f4*
IMG1791	UU	2	*multimedia_records_f3*
IMG800	CD	2	*multimedia_records_f1*

Después de realizar las 17 operaciones que se observan en la Tabla 4.7, se lleva a cabo el análisis de los umbrales. Los valores de operación y desempeño anteriores mostrados en la Figura 4.12 representan el 100% de los umbrales. Sin embargo, en esta evaluación, los umbrales se fijaron en el 90% y esto significa que ambos valores deben ser inferiores a los anteriores. La Tabla 4.8 describe qué fragmentos superan los umbrales.

Tabla 4.8. Análisis de los umbrales en la fragmentación dinámica

Nombre de fragmento	Valor de operación previo	Valor de operación actual	Valor de operación al 90%	Valor de desempeño previo	Valor de desempeño actual	Valor de desempeño al 90%
multimedia_records_f1	4	4	3.6	7	R+U+2C+2D=12	6.3
multimedia_records_f2	3	2	2.7	4	C+2D=6	3.6
multimedia_records_f3	1	5	.9	1	C+R+R+2U+ +2U=16	.9
multimedia_records_f4	2	6	1.8	3	U+2R+2R+2R+2R +2R=13	2.7

En la Tabla 4.8 se muestran los umbrales alcanzados por cada fragmento a través de las operaciones realizadas en la Tabla 4.7. Se observa que todos los fragmentos superan los umbrales del 90% excepto el fragmento *multimedia_records_f2* en su umbral de operaciones. Para representar la operación de refragmentación, se muestra la Figura 4.13 del fragmento *multimedia_records_f1*.

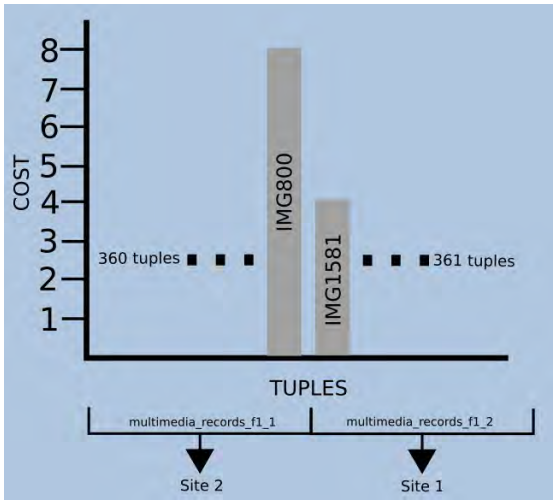


Figura 4.13. Fragmentación horizontal dinámica de *multimedia_records_f1*

Las tuplas se colocan en un diagrama que representa la curva Gaussiana para normalizar los costos. La Figura 4.13 muestra que la curva de Gauss las dividirá para generar dos nuevos fragmentos. El vigilante-fragmentador busca el lugar donde las tuplas de cada fragmento representan el mayor costo para realizar la asignación. Los valores actuales de operaciones y desempeño se consideran como valores previos en la siguiente operación de refragmentación. Bajo el nuevo esquema de refragmentación, se ejecutan las mismas operaciones presentadas en la Tabla 4.7. La Tabla 4.9 muestra la comparación de los tiempos de ejecución.

Tabla 4.9. Comparación de los tiempos de ejecución usando los ítems de las operaciones de la Tabla 4.7.

Ítem	Sitio	Tiempo de ejecución sin fragmentación	Tiempo de ejecución con este enfoque
1	1	42	3
2	1	38	5
3	1	51	1
4	1	6	Menos de 1
5	1	21	Menos de 1
6	2	43	8
7	2	26	6
8	2	8	6
9	2	27	Menos de 1
10	2	16	Menos de 1

La Tabla 4.9 muestra una reducción en los tiempos de respuesta en un 88,8% debido a que se redujo la comunicación remota y se incrementó la comunicación local. Las operaciones iniciales encontradas en el registro fueron solo lecturas y no incluyeron consultas basadas en contenido. Se realizó un nuevo experimento contemplando la recuperación de todos los descriptores para realizar consultas basadas en contenido utilizando el mismo atributo que la descripción multimedia del experimento anterior. Los fragmentos iniciales producidos por la aplicación Web son los mismos que en la Figura 4.12. La Tabla 4.10 presenta las nuevas operaciones encontradas en el *log*, realizadas en un nuevo experimento.

Tabla 4.10. Nuevas operaciones desempeñadas en la base de datos multimedia

Ítem	Operación	Sitio	Tiempo de ejecución en milisegundos
1	*db.multimedia_records.find({},{"descriptor:1"})*	1	945
2	*db.multimedia_records.update({_id:"IMG1790"},{$set:{"nombre":"cancha01modificado"}})*	1	58
3	*db.multimedia_records.find({},{"descriptor:1"})*	1	42
4	*db.multimedia_records.deleteOne({_id:"IMG1790"})*	1	61
5	*db.multimedia_records.insert({_id:"IMG1790",tipo:"building",descripcion:"image of the chemistry group 1999",validado:false,nombre:"img20"});*	1	41
6	*db.multimedia_records.find({},{"descriptor:1"})*	2	83
7	*db.multimedia_records.find({validado:{$ne:null}},{imagen:0,descriptor:0})*	2	13204
8	*db.multimedia_records.find({},{"descriptor:1"})*	2	16
9	*db.multimedia_records.update({_id:ObjectId("615e94dc92bd5f6ac041f437")},{$set:{"validado":"no"}}*	2	73
10	*db.multimedia_records.find({tipo:"event"})*	2	13

Como en la anterior evaluación, esta utiliza el 90% en ambos umbrales y se aplica en las mismas condiciones en una red de 2 sitios. La Tabla 4.11 muestra las operaciones que se desempeñan en el vigilante-fragmentador bajo el mismo esquema producido en la Figura 4.12. La Tabla 4.12 representa el cálculo de los umbrales para cada fragmento después de ejecutar las operaciones de la Tabla 4.10.

Tabla 4.11. Operaciones realizadas en la base de datos fragmentada en la nueva evaluación

Clave primaria del ítem buscado	Operación	Sitio	Nombre de fragmento
IMG103	CRRU	1	*multimedia_records_f1*
IMG110	UUU	1	*multimedia_records_f1*
IMG1663	RRD	1	*multimedia_records_f3*
ObjectId("615e9680f0f0e30bc4613072")	RRRD	1	*multimedia_records_f4*
ObjectId("615e94dc92bd5f6ac041f436")	RDC	1	*multimedia_records_f2*
IMG1570	UDC	2	*multimedia_records_f1*
IMG1561	RR	2	*multimedia_records_f1*
ObjectId("615e950692bd5f6ac041f460")	RU	2	*multimedia_records_f2*
ObjectId("615e96ccf0f0e30bc46130ba")	R	2	*multimedia_records_f4*
IMG1856	DC	2	*multimedia_records_f3*

Tabla 4.12. Análisis de umbrales en la fragmentación dinámica bajo las operaciones realizadas en la Tabla 4.11.

Nombre de fragmento	Valor de operación previo	Valor de operación actual	Valor de operación al 90%	Valor de desempeño previo	Valor de desempeño actual	Valor de desempeño al 90%
multimedia_records_f1	6	12	5.4	10	34	9
multimedia_records_f2	6	5	5.4	14	13	12.6
multimedia_records_f3	8	5	7.2	15	12	13.5
multimedia_records_f4	5	5	4.5	8	7	7.2

La Tabla 4.12 muestra que el único fragmento que superó ambos umbrales fue *multimedia_records_f1*. La refragmentación se realiza de la misma forma que en la Figura 4.13, colocando las tuplas más utilizadas en el centro del diagrama y produciendo dos nuevos fragmentos. Los tiempos de ejecución de las operaciones realizadas en la Tabla 4.10 se vuelven a comparar en la Tabla 4.13 para observar los diferentes tiempos de ejecución en esta nueva evaluación.

Tabla 4.13. Comparación de los tiempos de ejecución usando los ítems de las operaciones de la Tabla 4.10

Ítem	Sitio	Tiempo de ejecución sin fragmentación	Tiempo de ejecución con este enfoque
1	1	945	21
2	1	58	23
3	1	42	16
4	1	61	8
5	1	41	Menos de 1
6	2	83	29
7	2	13204	84
8	2	16	14
9	2	73	4
10	2	13	2

La Tabla 4.13 presenta una mejora en los tiempos de respuesta del 98%. Además, el trabajo presentado en Castro-Medina et al. (2020) muestra un simulador de operaciones para MySQL. Se enriquecieron las funcionalidades del simulador de consultas para obtener archivos de registro que simulan cargas de trabajo y sitios en el DBMS de este trabajo. Los tiempos de ejecución se obtuvieron utilizando un programa escrito en Java antes de la fragmentación y después de la fragmentación. Se simularon 200 consultas en cada sitio en la misma colección multimedia de MongoDB para obtener un porcentaje de mejora más preciso. Los tiempos de respuesta promedio resultaron en una reducción del 86%.

4.3. Fragmentación vertical multimedia y CBIR

La Figura 4.14 muestra la configuración del simulador de consultas SimulQuery para generar 8000 operaciones (2000 por sitio, en 4 sitios). La configuración es la misma para los diferentes SGBDs en esta evaluación, simplemente cambiando la selección a la izquierda de esta figura (Selecciona tu gestor).

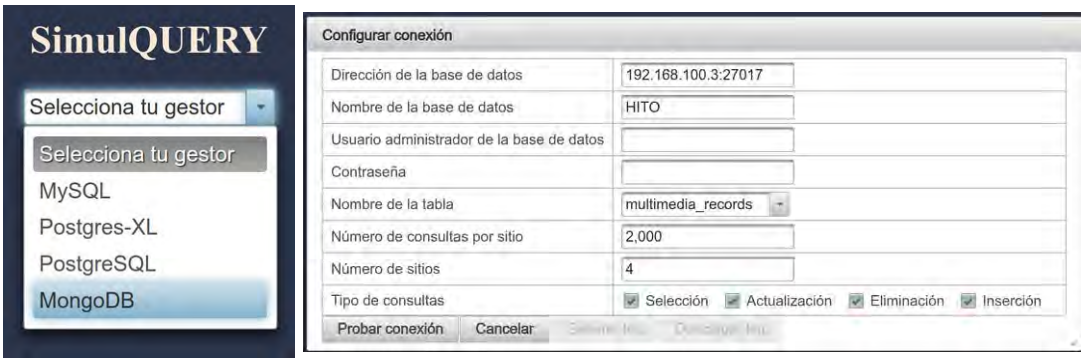


Figura 4.14. Configuración de SimulQUERY para evaluación con MongoDB.

La conexión al SGBD y a la relación se utilizan para simular operaciones con predicados tomados de la misma tabla. En el caso de MongoDB, a diferencia de los relacionales, la conexión también se usa para utilizar campos que están presentes solo en ciertos documentos seleccionados aleatoriamente. La configuración que se observa en la Figura 4.14 muestra que no se ingresó el nombre de usuario y la contraseña, ya que la autenticación está deshabilitada en esta base de datos.

La Figura 4.15 muestra la primera configuración de XAMANA que, al igual que el simulador, es la selección del SGBD y la forma de obtener los datos de conexión de la base de datos del usuario.

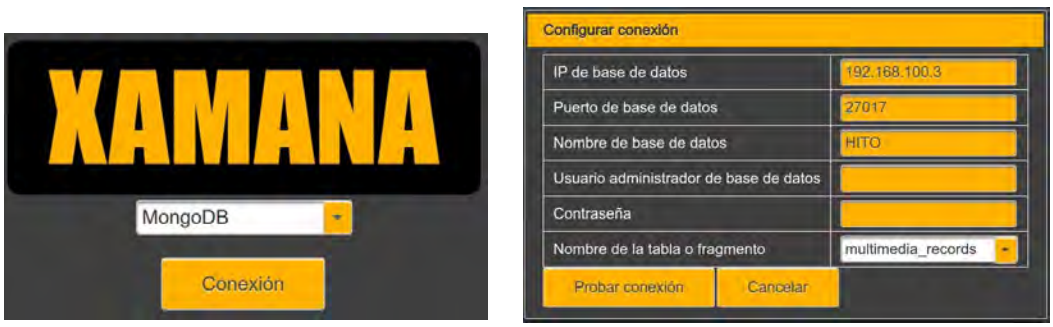


Figura 4.15. La selección del SGBD y la primera configuración de XAMANA.

La Figura 4.16 presenta el diálogo en donde se asignaron los tamaños a los campos en MongoDB, respetando el limite de 16 MB.

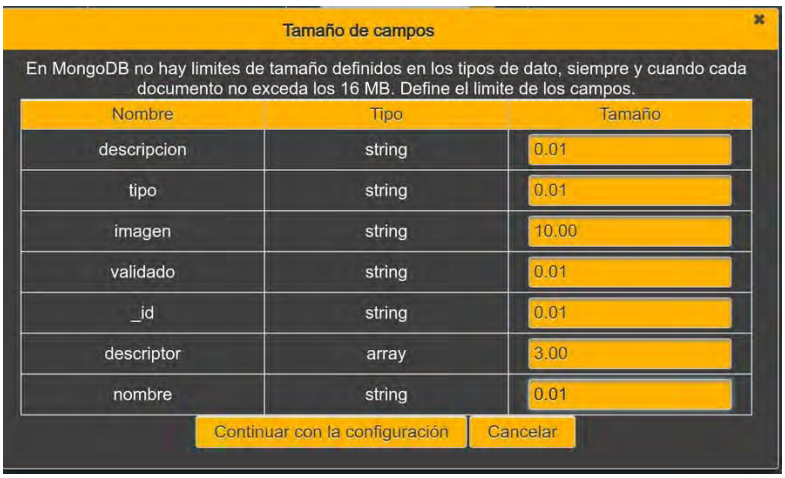


Figura 4.16. Asignación de tamaños de campos para MongoDB.

Se llevó a cabo la primera prueba bajo la configuración que incluye CBIR. La Figura 4.17 presenta el cuadro de diálogo para la configuración de fragmentación. Luego de seleccionar la fragmentación vertical y asignar los tamaños de los campos, al indicar que es una colección que incluye CBIR, se despliega un diálogo de selección de los pares de atributos CBIR, como se muestra en la Figura 4.18. El diálogo tiene la capacidad de seleccionar más de un par de atributos, la columna izquierda permite seleccionar el atributo multimedia y la columna central permite seleccionar el atributo descriptor. La última columna muestra las selecciones de los pares que se han realizado, permitiendo su eliminación.

Figura 4.17. Diálogo para la configuración de la fragmentación considerando CBIR.

Figura 4.18. Selección de atributos CBIR.

En la Figura 4.17 se selecciona el *log* que previamente se generó con SimulQUERY. Además, se muestran los dos umbrales que definen la frecuencia de la fragmentación dinámica. El 100% representa que el costo que se recabó inicialmente en los *logs* obtenidos mediante el formulario de esta misma figura se deben alcanzar. Sin embargo, los umbrales solo juegan un papel en la fragmentación dinámica cuando no se tiene en cuenta CBIR y cuando la fragmentación CBIR produce fragmentos que no son CBIR, como en esta evaluación (el único fragmento CBIR es *multimedia_records_1* que se muestra en la Figura 4.19).

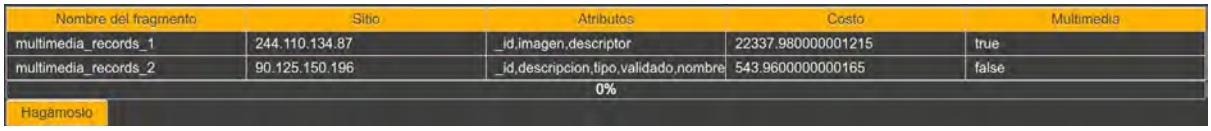


Figura 4.19. Esquema de fragmentación vertical CBIR propuesto.

La Figura 4.19 propone el nuevo esquema producido por la fragmentación vertical CBIR. Se muestra que *multimedia_records_1* es el fragmento CBIR asignado a la dirección que termina en 87, *multimedia_records_2* contiene todos los atributos que se excluyeron en el fragmento CBIR y están igualmente asignados al sitio que más los usa. En este flujo de trabajo de CBIR, se observa que dos sitios no tienen atributos, ya que se simularon cuatro sitios, sin embargo, es más importante que los atributos estén donde más se usan. En la fragmentación dinámica, que incluye los dos enfoques juntos (usando CBIR y cuando no está en uso), los fragmentos que no son CBIR esperan a los umbrales para ser refragmentados. Lo mismo ocurre con el

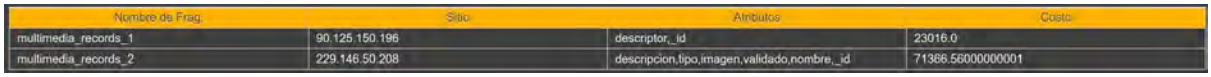
fragmento *multimedia_records_2*, pero después de que haya comenzado la fragmentación dinámica.

El observador-fragmentador debe estar presente en todos los nodos de la red que realizan operaciones hacia los fragmentos, incluidos los sitios a los que están asignados. Los observadores-fragmentadores se comunican entre sí para que sus esquemas de fragmentos estén actualizados. El observador-fragmentador dispone de ambos flujos de trabajo, detectando a través de la primera y posteriores fragmentaciones el tipo de cada uno de los fragmentos (CBIR o no). El observador-fragmentador en MongoDB analiza las nuevas operaciones que se realizan sobre los fragmentos que pertenecen al sitio donde se encuentra y realiza nuevas fragmentaciones cuando supera los umbrales establecidos en la Figura 4.17. Si es un fragmento CBIR, se reasigna si otro sitio lo ocupa más que el actual.

Los resultados obtenidos de esta evaluación muestran un aumento del costo del esquema en la fragmentación estática del 6.35%, pero una reducción en el del fragmento CBIR. Sin embargo, en fragmentos posteriores, el costo del esquema se reduce hasta en un 12.85%, medido con la misma carga de trabajo inicial. Las cargas de trabajo cambian con el tiempo, y este trabajo se destaca de otros enfoques porque adapta el esquema a esos cambios, reduciendo el costo a su valor más bajo frente a nuevas cargas de trabajo.

Para el flujo de trabajo de fragmentación vertical cuando no incluye CBIR, se utiliza la misma base de datos para realizar la evaluación sin tener en cuenta la naturaleza del descriptor SURF.

El esquema propuesto se muestra en la Figura 4.20. El costo de la carga de trabajo bajo el enfoque centralizado fue de 94,382.56. Como se mencionó anteriormente, se desarrolló un método para calcular el costo del nuevo esquema con la carga de trabajo actual, es decir, el costo que representaría el nuevo esquema bajo la misma carga de trabajo si se aplicara. El costo del nuevo esquema es de 24,714.02, teniendo una importante disminución del 73.81%.

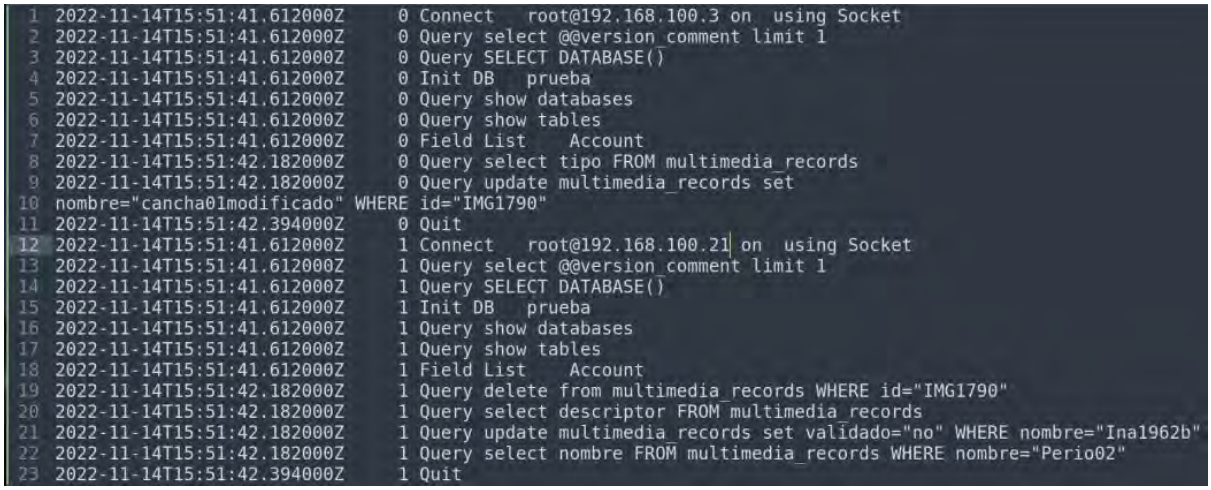


Nombre de Frag.	Sitio	Atributos	Costo
multimedia_records_1	90.125.150.196	descriptor,_id	23016.0
multimedia_records_2	229.146.50.208	descripcion,tipo,imagen,validado,nombre,_id	71386.56000000001

Figura 4.20. Esquema de fragmentación propuesto sin considerar CBIR.

Se presenta una nueva evaluación para observar las mejoras en los tiempos de respuesta de las operaciones ejecutadas en un ambiente distribuido bajo la misma base de datos HITO adaptada al SGBD MySQL. La base de datos está centralizada sin fragmentación en el sitio 192.168.100.3, que utiliza Ubuntu 22.04.1 LTS como sistema operativo, un procesador AMD Ryzen 7 5700g, 32 GB de RAM y almacenamiento SSD. Algunas operaciones se realizan desde otro sitio (192.168.100.21). El segundo sitio tiene el mismo sistema operativo, un procesador AMD A6, 4 GB de RAM y almacenamiento SSD.

La Tabla 4.14 enumera la línea de la operación y su tiempo de ejecución. La estructura de la tabla fue reconstruida luego de las operaciones que la modifican para que los tiempos de ejecución puedan observarse nuevamente luego de aplicar el enfoque de fragmentación. Los tiempos de ejecución se obtuvieron utilizando *set profiling = 1*.



```
 1 2022-11-14T15:51:41.612000Z     0 Connect   root@192.168.100.3 on  using Socket
 2 2022-11-14T15:51:41.612000Z     0 Query select @@version_comment limit 1
 3 2022-11-14T15:51:41.612000Z     0 Query SELECT DATABASE()
 4 2022-11-14T15:51:41.612000Z     0 Init DB   prueba
 5 2022-11-14T15:51:41.612000Z     0 Query show databases
 6 2022-11-14T15:51:41.612000Z     0 Query show tables
 7 2022-11-14T15:51:41.612000Z     0 Field List    Account
 8 2022-11-14T15:51:42.182000Z     0 Query select tipo FROM multimedia_records
 9 2022-11-14T15:51:42.182000Z     0 Query update multimedia_records set
10 nombre="cancha01modificado" WHERE id="IMG1790"
11 2022-11-14T15:51:42.394000Z     0 Quit
12 2022-11-14T15:51:41.612000Z     1 Connect   root@192.168.100.21 on  using Socket
13 2022-11-14T15:51:41.612000Z     1 Query select @@version_comment limit 1
14 2022-11-14T15:51:41.612000Z     1 Query SELECT DATABASE()
15 2022-11-14T15:51:41.612000Z     1 Init DB   prueba
16 2022-11-14T15:51:41.612000Z     1 Query show databases
17 2022-11-14T15:51:41.612000Z     1 Query show tables
18 2022-11-14T15:51:41.612000Z     1 Field List    Account
19 2022-11-14T15:51:42.182000Z     1 Query delete from multimedia_records WHERE id="IMG1790"
20 2022-11-14T15:51:42.182000Z     1 Query select descriptor FROM multimedia_records
21 2022-11-14T15:51:42.182000Z     1 Query update multimedia_records set validado="no" WHERE nombre="Ina1962b"
22 2022-11-14T15:51:42.182000Z     1 Query select nombre FROM multimedia_records WHERE nombre="Perio02"
23 2022-11-14T15:51:42.394000Z     1 Quit
```

Figure 4.21. *Log* de MySQL para la nueva evaluación.

Tabla 4.14. Tiempos de ejecución de las operaciones mostradas en la Figura 4.21.

Línea de operación	Con fragmentación (μs)	Sin fragmentación (μs)
8	753.75	806
9	499.75	18,077.5

Línea de operación	Con fragmentación (μs)	Sin fragmentación (μs)
19	233.40	8,562
20	530,323.5	1,900,879.25
21	352,589	3,948,183
22	880.75	915.5

La Tabla 4.14 muestra que el enfoque presentado en este trabajo reduce los tiempos de ejecución de las operaciones en un 84.93% en la primera fragmentación. La fragmentación dinámica preserva o mejora los costos ante cualquier cambio en las tendencias de las operaciones.

Un nuevo experimento se llevó a cabo bajo las mismas condiciones, utilizando un archivo *log* con 12.000 operaciones y 6 sitios. Se produjo una carga de trabajo no uniforme, es decir, el uso de atributos es desigual entre los sitios utilizados. Se observa el comportamiento de XAMANA, produciendo un nuevo esquema mostrado en la Figura 4.22.
El esquema propuesto muestra 5 fragmentos con diferentes atributos y los costos relacionados con cada uno de ellos. Bajo el esquema centralizado se produjo un costo de 1457.41. Aplicando la primera fragmentación este costo se reduce a 336.25, una reducción del 76.2%.
El vigilante-fragmentador se colocó en todos los sitios remotos, sin embargo, se observó el comportamiento de *multimedia_records_3*, recibiendo operaciones desde 218.1.87.199. Se simularon las operaciones necesarias para superar los umbrales. El vigilante-fragmentador propuso apropiadamente el nuevo esquema, ya que el par de atributos los reasignaba al sitio ya mencionado. Al recibir nuevas consultas remotas, el fragmento no retiene su costo inicial, y al reasignarlo al sitio donde se realizan, vuelve a reducir su costo. Los fragmentos que contienen la cantidad mínima de atributos (clave principal y atributo, o *_id* y campo) no se deben volver a fragmentar.

Adicionalmente se presenta en la Figura 4.22 una nueva forma de desplegar el esquema. Esta se caracteriza por permitir la modificación de cada uno de los fragmentos, teniendo la opción de agregar o quitar atributos, así como eliminar o agregar fragmentos nuevos. Esta característica se habilitó para que el DBA tome las últimas decisiones administrativas de

forma completamente independiente del modelo de costos en el mismo esquema.

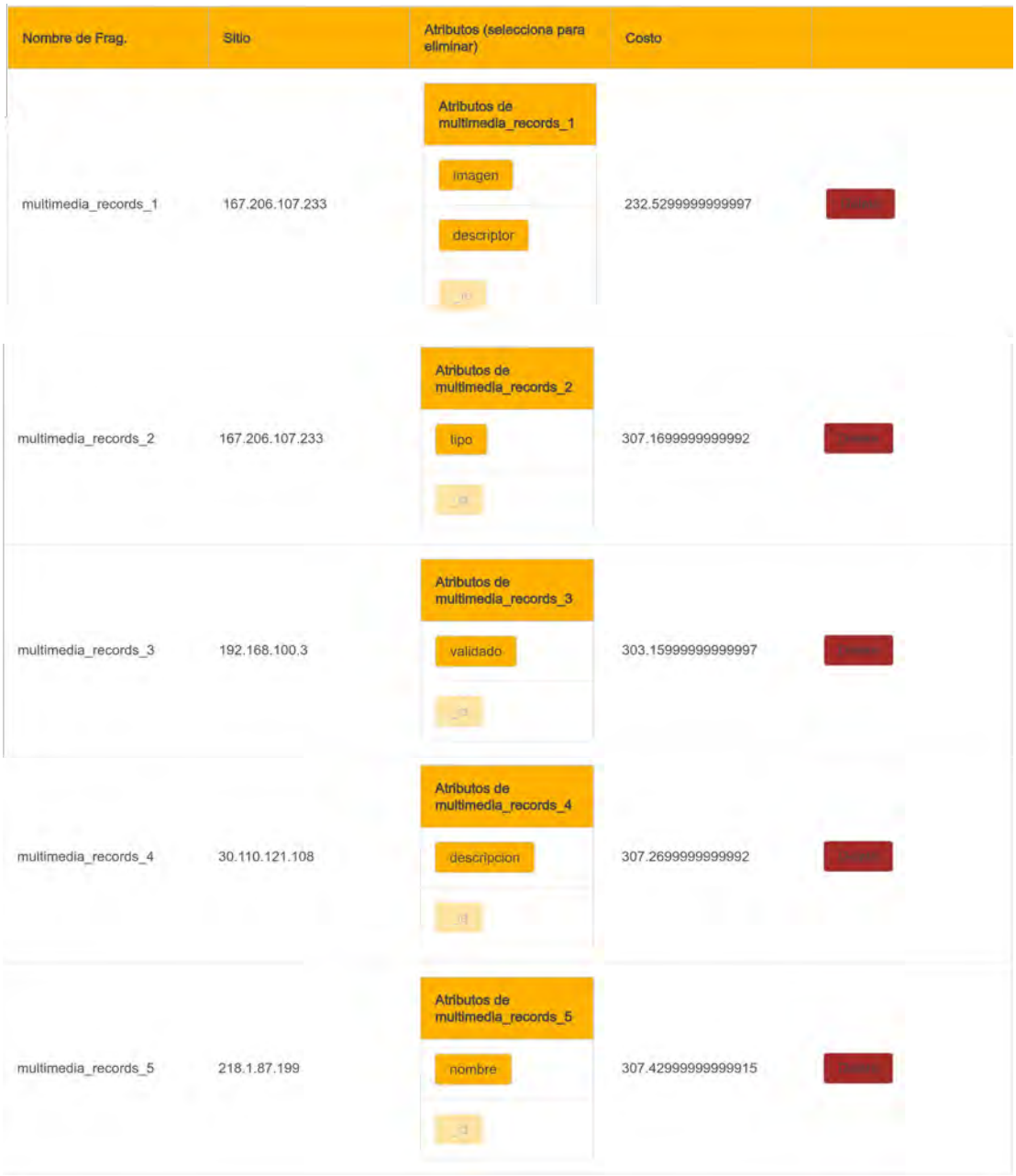


Figura 4.22. La estructura propuesta por XAMANA en la primera fragmentación.

Se llevó a cabo una estancia de investigación en la Centro Universitario UAEM (Universidad Autónoma del Estado de México) Zumpango y se desarrolló un enfoque para modificar la implementación del análisis de las operaciones mediante muestreo. De esta manera, se reducen las operaciones analizadas y mejora el desempeño de XAMANA, contemplado solo las operaciones más relevantes y descartando las irrelevantes. La formula utilizada se

presenta en la Ecuación (6).

$$n = \frac{N \sigma^2 Z^2}{(N-1)\mathrm{e}^2 + \sigma^2 Z^2} \tag{6}$$

Donde σ es la desviación estándar de la población, *N* es el tamaño de la población, *Z* es el valor obtenido de la distribución normal para un nivel de confianza del 95%, e es el límite aceptable de error y *n* es el tamaño mínimo de la población objetivo esperado para un nivel de confianza del 95%.

El resultado de esta modificación se obtuvo de manera satisfactoria, sin embargo, los esquemas producidos mediante una reducción de las operaciones analizadas cambian drásticamente frente al análisis de todas las operaciones realizadas. Si el número de operaciones realizadas es muy grande para el servidor en donde se encuentra XAMANA, esta modificación es de gran utilidad para la reducción del trabajo computacional.

4.4. Fragmentación híbrida multimedia y CBIR

La evaluación se llevó a cabo en dos sitios. El primer sitio utiliza Ubuntu 22.04.3 LTS como sistema operativo, 32 GB de RAM y un procesador AMD RYZEN 7 con una frecuencia de reloj de 3.8 GHz. El segundo sitio usa el mismo sistema operativo y 8 GB de RAM. La base de datos centralizada se encuentra en el primer sitio. El conjunto de operaciones para ambos flujos de trabajo (estático y dinámico) se obtienen mediante el simulador de operaciones SimulQuery. Para la fragmentación en XAMANA, se obtuvo un archivo *log* con 100 operaciones, el cual se selecciona en la etapa de la configuración de la fragmentación. La Figura 4.23 presenta la configuración de la fragmentación híbrida CBIR para el SGBD MongoDB.

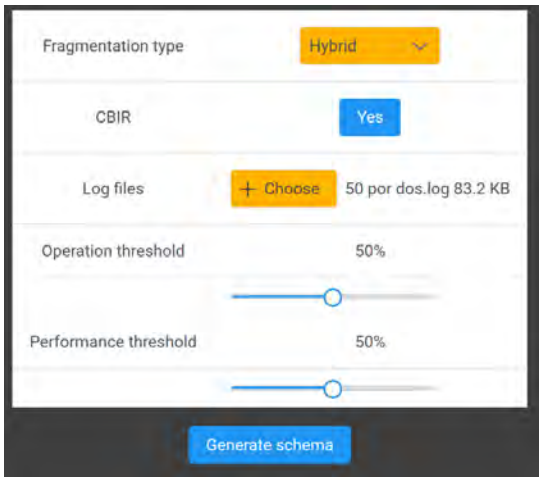


Figura 4.23. Configuración de la fragmentación híbrida CBIR

La Figura 4.23 presenta la selección de las configuraciones para desempeñar la fragmentación híbrida. Cuando se indica que se desea fragmentar de manera híbrida la relación, se despliega el diálogo para la asignación de los tamaños de los atributos, dicho diálogo solo se relaciona con el SGBD de MongoDB, ya que este no establece tamaños de campos, sino solo un tamaño máximo de documento (16 MB). Al seleccionar la configuración CBIR, se despliegan dos diálogos, el primero muestra la selección de los atributos multimedia y descriptor. En el flujo de trabajo de la fragmentación vertical, los atributos multimedia y descriptor se colocan en un fragmento por separado de los otros atributos no CBIR. El segundo diálogo solicita la selección del atributo del descriptor textual, el cual se utiliza en el flujo de trabajo de la fragmentación horizontal para identificar grupos y agruparlos mediante predicados diferentes de valores incluidos en ese atributo. En esta evaluación, los umbrales se colocan al 50%.

El esquema propuesto se muestra en la Figura 4.24 y se observa que se obtienen por fragmento dos costos y dos sitios. Para la determinación del costo vertical, se utiliza el modelo de costos de la fragmentación vertical y el modelo de costos de la fragmentación horizontal, ambos con la asignación vertical (sitio vertical). Para determinar el costo horizontal, de igual manera se utilizan ambos modelos de costos, pero el sitio utilizado para calcular el costo es el horizontal. La sumatoria de los costos de cada tipo producen el costo horizontal (HV) y el costo vertical (VH) del esquema entero. Se observa en la Figura 4.24 el

traslape de predicados híbridos, el cual es un tópico ya descrito en el Capítulo 3.

XAMANA muestra un mensaje, indicando el esquema menos costoso, que en el caso de esta evaluación es el vertical. Al aplicar el esquema y finalizar la fragmentación, se muestra un mensaje indicando el *token* relacionado con este conjunto de fragmentos. El *token* es el mecanismo por el cual el vigilante-fragmentador identifica todos los datos necesarios para desempeñar su trabajo.

Las asignaciones verticales mostradas en la Figura 4.24 presentan que todos los fragmentos se asignarán al sitio 192.168.100.20. La base de datos centralizada se encuentra en el 192.168.100.3.

Después de ejecutar el vigilante-fragmentador en cada sitio, este solicita el *token* generado en XAMANA y la dirección de los archivos *log*. Al colocar esos datos, el vigilante-fragmentador recupera el esquema actual, y todos los datos necesarios para realizar el análisis de costos inicial. Para minimizar el trabajo computacional, la fragmentación dinámica realiza un primer análisis directo de fragmentación híbrida (sin considerar las asignaciones verticales y horizontales), bajo su propio modelo de costos. Al detectar que un fragmento incluido en el mismo sitio que en el vigilante superó los umbrales, se lleva a cabo el flujo de trabajo de la fragmentación vertical, el flujo de trabajo de la fragmentación horizontal, la mezcla de los esquemas y el flujo de trabajo de la fragmentación híbrida (nuevamente, pero considerando los sitios de asignación vertical y horizontal), en ese orden.

Nombre	Predicado con traslape	Atributos	Sitio horizontal	Sitio vertical	Costo vertical	Costo horizontal
multimedia_records_f1	{"tipo":{$eq:"personal"}}	_id,imagen,descriptor	192.168.100.3	192.168.100.20	103234.86000000004	122327.58000000006
multimedia_records_f2	{"tipo":{$eq:"personal"}}	descripcion,tipo,validado,_id,nombre	192.168.100.3	192.168.100.20	205233.99999999997	227394.59000000003
multimedia_records_f3	{"tipo":{$not:{$eq:"personal"}}},{"tipo":{$eq:"event"}}	_id,imagen,descriptor	192.168.100.3	192.168.100.20	102009.44	121256.92000000007
multimedia_records_f4	{"tipo":{$not:{$eq:"personal"}}},{"tipo":{$eq:"event"}}	descripcion,tipo,validado,_id,nombre	192.168.100.3	192.168.100.20	182612.25000000006	189666.24000000008
multimedia_records_f5	{"tipo":{$not:{$eq:"event"}}},{"tipo":{$not:{$eq:"personal"}}},{"tipo":{$eq:"equipment"}}	_id,imagen,descriptor	192.168.100.3	192.168.100.20	96421.20000000003	116596.80000000005
multimedia_records_f6	{"tipo":{$not:{$eq:"event"}}},{"tipo":{$not:{$eq:"personal"}}},{"tipo":{$eq:"equipment"}}	descripcion,tipo,validado,_id,nombre	192.168.100.3	192.168.100.20	168541.91000000006	188289.34000000005
multimedia_records_f7	{"tipo":{$not:{$eq:"equipment"}}},{"tipo":{$not:{$eq:"event"}}},{"tipo":{$not:{$eq:"personal"}}},{"tipo":{$eq:"building"}}	_id,imagen,descriptor	192.168.100.3	192.168.100.20	86272.96000000002	106512.68000000005
multimedia_records_f8	{"tipo":{$not:{$eq:"equipment"}}},{"tipo":{$not:{$eq:"event"}}},{"tipo":{$not:{$eq:"personal"}}},{"tipo":{$eq:"building"}}	descripcion,tipo,validado,_id,nombre	192.168.100.3	192.168.100.20	174287.94000000006	194099.49000000005

Costo HV: 240811.00999999992

Costo VH: 217383.3499999999

Figura 4.24. Esquema híbrido CBIR propuesto por la fragmentación estática realizada en XAMANA.

El vigilante-fragmentador no posee una interfaz gráfica, ya que los datos que solicita se proporcionan desde la línea de comandos o la terminal. La Figura 4.25 muestra la solicitud de los dos datos ya mencionados.

```
Coloca el token obtenido de la aplicación Web para iniciar el método:
107
Coloca la dirección del archivo log de la base de datos:
/home/pipe/Documentos/log
```

Figura 4.25. Solicitud de datos del vigilante-fragmentador híbrido.

Al proporcionar el archivo *log* al vigilante-fragmentador, analiza el costo de las operaciones desempeñadas y determina que los fragmentos superaron el umbral establecido. La fragmentación dinámica continua con la producción de un esquema vertical, el cual se muestra en la Tabla 4.15.

El vigilante-fragmentador se comunica con los otros vigilantes colocados en los otros servidores en todo momento, ya que si un vigilante externo realiza una fragmentación de los fragmentos que le corresponden, pero uno de ellos se debe asignar al sitio a donde se encuentra el vigilante local, este debe ahora también analizar y contemplar el nuevo fragmento recién llegado. Esta actividad se lleva a cabo mediante *sockets*.

La Tabla 4.16 presenta el esquema de fragmentación horizontal propuesto por la fragmentación híbrida y por ultimo, la Tabla 4.17, el esquema híbrido.

Tabla 4.15. Esquema vertical propuesto por el vigilante-fragmentador híbrido.

Número de fragmento	Atributos	Sitio	Origen
1	*_id, imagen, descriptor*	192.168.100.3	*multimedia_records_f1*
2	*_id, imagen, descriptor*	192.168.100.3	*multimedia_records_f3*
3	*_id, imagen, descriptor*	192.168.100.3	*multimedia_records_f5*
4	*_id, imagen, descriptor*	192.168.100.3	*multimedia_records_f7*
5	*tipo, _id*	192.168.100.3	*multimedia_records_f2*
6	*_id, validado, nombre*	192.168.100.3	*multimedia_records_f2*
7	*descripcion, _id*	192.168.100.3	*multimedia_records_f2*
8	*tipo, _id*	192.168.100.3	*multimedia_records_f4*
9	*_id, validado, nombre*	192.168.100.3	*multimedia_records_f4*

Número de fragmento	Atributos	Sitio	Origen
10	*descripcion, _id*	192.168.100.3	*multimedia_records_f4*
11	*tipo, _id*	192.168.100.3	*multimedia_records_f6*
12	*_id, validado, nombre*	192.168.100.3	*multimedia_records_f6*
13	*descripcion, _id*	192.168.100.3	*multimedia_records_f6*
14	*tipo, _id*	192.168.100.3	*multimedia_records_f8*
15	*_id, validado, nombre*	192.168.100.3	*multimedia_records_f8*
16	*descripcion, _id*	192.168.100.3	*multimedia_records_f8*

Tabla 4.16. Esquema horizontal propuesto por el vigilante-fragmentador híbrido.

Número de fragmento	Predicado	Sitio	Origen
1	*{tipo:{$eq:"personal"}}*	192.168.100.20	*multimedia_records_f2*
2	*{tipo:{$eq:"event"}}*	192.168.100.20	*multimedia_records_f4*
3	*{tipo:{$eq:"equipment"}}*	192.168.100.20	*multimedia_records_f4*
4	*{tipo:{$eq:"building"}}*	192.168.100.20	*multimedia_records_f4*
5	*{tipo:{$eq:"building"}}*	192.168.100.20	*multimedia_records_f6*
6	*{tipo:{$eq:"personal"}}*	192.168.100.20	*multimedia_records_f6*
7	*{tipo:{$eq:"equipment"}}*	192.168.100.20	*multimedia_records_f6*
8	*{tipo:{$eq:"building"}}*	192.168.100.20	*multimedia_records_f8*
9	*{tipo:{$eq:"personal"}}*	192.168.100.20	*multimedia_records_f8*
10	*{tipo:{$eq:"event"}}*	192.168.100.20	*multimedia_records_f8*
11	*{"_id":{"$ne":"IMG471"}}*	192.168.100.3	*multimedia_records_f1*
12	*{"_id":{"$ne":"IMG471"}}*	192.168.100.3	*multimedia_records_f3*
13	*{"_id":{"$ne":"IMG471"}}*	192.168.100.3	*multimedia_records_f5*
14	*{"_id":{"$ne":"IMG471"}}*	192.168.100.3	*multimedia_records_f7*

El esquema de fragmentación horizontal presentando en la Tabla 4.16 muestra los primeros 10 fragmentos obtenidos mediante la fragmentación horizontal CBIR. La columna “Origen”

contiene los fragmentos originales, de los cuales se produjeron los fragmentos. Los últimos 4 fragmentos no son CBIR y se obtienen mediante el otro flujo de trabajo de la fragmentación horizontal.

Tabla 4.17. Esquema híbrido propuesto por el vigilante-fragmentador.

Nombre del fragmento	Atributos	Predicado	Sitio horizontal	Sitio vertical	Costo horizontal	Costo vertical
multimedia_records_f2_1	*tipo, _id*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	463384.56	444057.16
multimedia_records_f2_2	*_id, validado, nombre*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	587380.14	568052.74
multimedia_records_f2_3	*descripcion, _id*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	552213.54	552201.42
multimedia_records_f6_4	*tipo, _id*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	463384.56	444057.16
multimedia_records_f6_5	*_id, validado, nombre*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	587380.14	568052.74
multimedia_records_f6_6	*descripcion, _id*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	552213.54	552201.42
multimedia_records_f8_7	*tipo, _id*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	463384.56	444057.16
multimedia_records_f8_8	*_id, validado, nombre*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	587380.14	568052.74
multimedia_records_f8_9	*descripcion, _id*	*{tipo:{$eq:"personal"}}*	192.168.100.20	192.168.100.3	552213.54	552201.42
multimedia_records_f4_10	*tipo, _id*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	463311.12	443983.72
multimedia_records_f4_11	*_id, validado, nombre*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	587251.8	567924.4
multimedia_records_f4_12	*descripcion, _id*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	541033.2	521705.8
multimedia_records_f6_13	*tipo, _id*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	463311.12	443983.72
multimedia_records_f6_14	*_id, validado, nombre*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	587251.8	567924.4
multimedia_records_f6_15	*descripcion, _id*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	541033.2	521705.8
multimedia_records_f8_16	*tipo, _id*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	463311.12	443983.72
multimedia_records_f8_17	*_id, validado, nombre*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	587251.8	567924.4
multimedia_records_f8_18	*descripcion, _id*	*{tipo:{$eq:"building"}}*	192.168.100.20	192.168.100.3	541033.2	521705.8
multimedia_records_f4_19	*tipo, _id*	*{tipo:{$eq:"equipment"}}*	192.168.100.20	192.168.100.3	463311.12	443983.72
multimedia_records_f4_20	*_id, validado, nombre*	*{tipo:{$eq:"equipment"}}*	192.168.100.20	192.168.100.3	587251.8	567924.4
multimedia_records_f4_21	*descripcion, _id*	*{tipo:{$eq:"equipment"}}*	192.168.100.20	192.168.100.3	541033.2	521705.8
multimedia_records_f6_22	*tipo, _id*	*{tipo:{$eq:"equipment"}}*	192.168.100.20	192.168.100.3	463311.12	443983.72

Nombre del fragmento	Atributos	Predicado	Sitio horizontal	Sitio vertical	Costo horizontal	Costo vertical
multimedia_records_f6_23	*_id, validado, nombre*	*{tipo:{$eq:"equipment"}}*	192.168.100.20	192.168.100.3	587251.8	567924.4
multimedia_records_f6_24	*descripcion, _id*	*{tipo:{$eq:"equipment"}}*	192.168.100.20	192.168.100.3	541033.2	521705.8
multimedia_records_f1_25	*_id, imagen, descriptor*	*{"_id":{"$ne":"IMG471"}}*	192.168.100.3	192.168.100.3	480852.24	461524.84
multimedia_records_f3_26	*_id, imagen, descriptor*	*{"_id":{"$ne":"IMG471"}}*	192.168.100.3	192.168.100.3	480778.8	461451.4
multimedia_records_f5_27	*_id, imagen, descriptor*	*{"_id":{"$ne":"IMG471"}}*	192.168.100.3	192.168.100.3	480852.24	461524.84
multimedia_records_f7_28	*_id, imagen, descriptor*	*{"_id":{"$ne":"IMG471"}}*	192.168.100.3	192.168.100.3	480852.24	461524.84
multimedia_records_f4_29	*tipo, _id*	*{tipo:{$eq:"event"}}*	192.168.100.20	192.168.100.3	418044.48	412252.4
multimedia_records_f4_30	*_id, validado, nombre*	*{tipo:{$eq:"event"}}*	192.168.100.20	192.168.100.3	489856.44	470529.04
multimedia_records_f4_31	*descripcion, _id*	*{tipo:{$eq:"event"}}*	192.168.100.20	192.168.100.3	443637.84	424310.44
multimedia_records_f8_32	*tipo, _id*	*{tipo:{$eq:"event"}}*	192.168.100.20	192.168.100.3	418044.48	412252.4
multimedia_records_f8_33	*_id, validado, nombre*	*{tipo:{$eq:"event"}}*	192.168.100.20	192.168.100.3	489856.44	470529.04
multimedia_records_f8_34	*descripcion, _id*	*{tipo:{$eq:"event"}}*	192.168.100.20	192.168.100.3	443637.84	424310.44
multimedia_records_fx_35	*_id, imagen, descriptor, descripcion, tipo, validado, nombre*	*{"$and":[{$or:[{tipo:{$not:{$eq:"event"}}}]},{$or:[{"_id":{$not:{"$ne":"IMG471"}}}]},{$or:[{tipo:{$not:{$eq:"equipment"}}}]},{$or:[{tipo:{$not:{$eq:"building"}}}]},{$or:[{tipo:{$not:{$eq:"personal"}}}]}]}*	192.168.100.20	192.168.100.20		

El esquema de fragmentación híbrida dinámica presenta la mezcla entre los esquemas horizontal y vertical, incluyendo el fragmento con el predicado que niega a todos los demás (traslape híbrido). Cada fragmento contiene además el predicado con traslape, que es el encargado de definir los fragmentos híbridos en su parte horizontal.

Este enfoque se destaca por llevar a cabo un nuevo método de fragmentación híbrida, que incluye factores como el traslape de predicados híbridos y horizontales, la afinidad de atributos vertical y esquemas de refragmentación que toman en cuenta el fragmento origen. Esto último se ve reflejado en el esquema presentado en la Tabla 4.17, ya que un enfoque estático (también incluido en esta investigación) debe limitar los esquemas para que cada

fragmento no contenga predicados y atributos iguales, sin embargo, en este enfoque dinámico, esto es probable que suceda en diferentes escenarios y se debe permitir siempre y cuando se tengan en cuenta los orígenes de cada fragmento. Esto se observa en diferentes fragmentos, por ejemplo, en el fragmento *multimedia_records_f4_29* y *multimedia_records_f8_32*. Ambos poseen los mismos atributos y los mismos predicados, sin embargo, su origen es diferente y los predicados aplicados producirán diferentes tuplas.

El esquema centralizado posee un costo de 247,634.67, el cual se reduce al costo vertical mostrado en la Figura 4.24, el cual es de 217,383.34. La reducción del costo del esquema bajo el modelo de costos híbrido es del 12.22%, sin embargo, una de las ventajas, además de la reducción del costo, es la capacidad de adaptar el esquema a los cambios que sufren los patrones de acceso.

CONCLUSIONES

El enfoque presentado mejora el desempeño del SGBD al reducir el tiempo de ejecución de las operaciones asignando los fragmentos a los sitios donde representan un mayor costo. La implementación de CBIR es una tarea que requiere un gran esfuerzo computacional. Proponer estrategias de fragmentación para reducir la carga de trabajo en la búsqueda de elementos cercanos (k-NN) es una necesidad. Fragmentar una base de datos es una tarea ardua, ya que en grandes bases de datos la cantidad de datos que se mueven de un lugar a otro es significativa. Realizar la fragmentación sin un sistema en su lugar es una tarea difícil para los administradores de bases de datos y aún más difícil de llevar a cabo de forma dinámica. El enfoque de fragmentación dinámica propuesto en este trabajo facilita notablemente la tarea de llevar a cabo la fragmentación de forma precisa y adecuada.

El trabajo propuesto presenta las tres estrategias de fragmentación dentro de XAMANA, de manera estática y dinámica. El administrador de bases de datos elige la configuración que más se adecúe a sus necesidades, eligiendo el tipo de fragmentación, los umbrales y el tipo de consultas que atienden sus SGBD.

Los resultados muestran que la misma carga de trabajo en un entorno fragmentado bajo este enfoque mejora en gran medida los tiempos de respuesta. Los costos se minimizan debido a la reducción del transporte de datos y la disminución del acceso a tuplas irrelevantes. Al recorrer todos los descriptores para encontrar imágenes similares en consultas basadas en contenido, el acceso a tuplas irrelevantes es muy grande. Este trabajo propone un enfoque completamente innovador, ya que ninguno de los trabajos relacionados aborda los diferentes temas incluidos en esta investigación.

RECOMENDACIONES

La aplicación de los tres enfoques de fragmentación de esta investigación es de gran utilidad para la industria y la academia, sin embargo, se recomienda un profundo análisis de cada caso de estudio para su aplicación, para que de esta manera se obtengan las mejores ventajas de este trabajo.

Para la fragmentación horizontal:

- Se debe considerar una modificación en el flujo de trabajo al crear la matriz multimedia alfanumérica. La producción del número de fragmentos es directamente proporcional a los predicados relacionados con el atributo con el mayor valor ALP, sin embargo, en conjuntos de operaciones de gran escala, al contener una cantidad considerable de predicados diferentes, el número de fragmentos es muy alto, esto repercute directamente en el aumento del costo de reuniones para resolver nuevas operaciones.

Para la fragmentación híbrida:

- En la aplicación del enfoque de fragmentación híbrida se debe considerar que la producción de fragmentos requiere un alto trabajo computacional, tanto en su versión estática como dinámica.
- Los casos de estudio deben ser analizados de manera profunda para corroborar que este tipo de fragmentación es el que mejor se adapta a las necesidades, ya que la aplicación en un caso de estudio que no se adapte al producto (esquema) de este tipo de fragmentación, repercutirá de manera negativa al desempeño.
- La consideración del fragmento que contiene el predicado que niega a todos los demás (traslape híbrido), debe incluirse por fragmento origen, es decir, debe existir un fragmento "negado" por cada fragmento original que se refragmente (considerando que los nuevos fragmentos no contengan todas las tuplas del fragmento original).

REFERENCIAS BIBLIOGRÁFICAS

Abdalla, H., & Amer, A. (2012). Dynamic horizontal fragmentation, replication and allocation model in DDBSs. 2012 International Conference On Information Technology And E-Services. doi: 10.1109/icites.2012.6216603

Abdel Raouf, A., Badr, N., & Tolba, M. (2016). Distributed Database System (DSS) Design Over a Cloud Environment. Intelligent Systems Reference Library, 97-116. doi: 10.1007/978-3-319-44270-9_5

Abeles, P. (2021). BoofCV. Retrieved 4 November 2021, from http://boofcv.org/index.php?title=Main_Page

Alagiannis, I., Idreos, S., & Ailamaki, A. (2014). H2O A Hands-free Adaptive Store. Proceedings Of The 2014 ACM SIGMOD International Conference On Management Of Data - SIGMOD '14. doi: 10.1145/2588555.2610502

Al-Kateb, M., Sinclair, P., Au, G., & Ballinger, C. (2016). Hybrid row-column partitioning in teradata®. Proceedings Of The VLDB Endowment, 9(13), 1353-1364. doi: 10.14778/3007263.3007273

Amer, A. (2020). On K-means clustering-based approach for DDBSs design. Journal Of *Big Data*, 7(1). doi: 10.1186/s40537-020-00306-9

Arauz, M.J., Rodríguez-Mazahua, L., Arrioja-Rodríguez, M.L., Abud-Figueroa, M.A., Peláez-Camarena, S.G., & Martínez-Méndez, L.D. (2020). Design of a Multimedia Data Management System that Uses Horizontal Fragmentation to Optimize Content-based Queries. In Proceedings of the Tenth International Conference on Advances in Information Mining and Management.

Baron, C., & Iacob, N. (2014). A New Dynamic Data Fragmentation and Replication Model in DDBMSs Cost Functions. Econpapers, 6(1), 158-161.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded Up Robust Features. Computer Vision – ECCV 2006, 404-417. doi: 10.1007/11744023_32

Bellatreche, L., Benkrid, S., Ghazal, A., Crolotte, A., & Cuzzocrea, A. (2011). Verification of Partitioning and Allocation Techniques on Teradata DBMS. In ICA3PP'11: Proceedings of the 11th international conference on Algorithms and architectures for parallel processing (pp. 158-168). Futuroscope, France: ACM Digital Library.

Bellatreche, L., Bouchakri, R., Cuzzocrea, A., & Maabout, S. (2013). Incremental Algorithms for Selecting Horizontal Schemas of Data Warehouses: The Dynamic Case. Lecture Notes In Computer Science, 13-25. doi: 10.1007/978-3-642-40053-7_2

Campero Durand, G., Piriyev, R., Pinnecke, M., Broneske, D., Gurumurthy, B., & Saake, G. (2019). Automated Vertical Partitioning with Deep Reinforcement Learning. Communications In Computer And Information Science, 126-134. doi: 10.1007/978-3-030-30278-8_16

Castro-Medina, F., Rodríguez-Mazahua, L., López-Chau, A., Cervantes, J., Alor- Hernández, G., and Machorro-Cano, I. "Application of dynamic fragmentation methods in multimedia databases: A review," Entropy, vol. 22, no. 12. MDPI AG, pp. 1–25, Dec. 01, 2020. doi: 10.3390/e22121352.

Castro-Medina, F., Rodriguez-Mazahua, L., Lopez-Chau, A., Abud-Figueroa, M., & Alor-Hernandez, G. (2020). FRAGMENT: A Web Application for Database Fragmentation, Allocation and Replication over a Cloud Environment. IEEE Latin America Transactions, 18(06), 1126-1134. doi: 10.1109/tla.2020.9099751

Chbeir, Richard & Laurent, Dominique. (2010). Enhancing Multimedia Data Fragmentation. JMPT. 1. 112-130.

Chen, K., Zhou, Y., & Cao, Y. (2015). Online Data Partitioning in Distributed Database Systems. EDBT.

Chernishev, G. (2017). The design of an adaptive column-store system. Journal Of *Big Data*, 4(1). doi: 10.1186/s40537-017-0069-4

Chi-wai Fung, Karlapalem, K., & Qing Li. (2002). An evaluation of vertical class partitioning for query processing in object-oriented databases. IEEE Transactions On Knowledge And Data Engineering, 14(5), 1095-1118.

Costa, E., Costa, C., & Santos, M. (2019). Evaluating partitioning and bucketing strategies for Hive-based *Big Data* Warehousing systems. Journal Of *Big Data*, 6(1). doi: 10.1186/s40537-019-0196-1

Cuzzocrea, A., Darmont, J., & Mahboubi, H. (2009). Fragmenting very large XML data warehouses via K-means clustering algorithm. International Journal Of Business Intelligence And Data Mining, 4(3/4), 301. doi: 10.1504/ijbidm.2009.029076

Derrar, H., Nacer, M., & Boussaid, O. (2013). Exploiting data access for dynamic fragmentation in data warehouse. International Journal Of Intelligent Information And Database Systems, 7(1), 34. doi: 10.1504/ijiids.2013.051736

Durand, G., Pinnecke, M., Piriyev, R., Mohsen, M., Broneske, D., & Saake, G. et al. (2018). GridFormation: Towards Self-Driven Online Data Partitioning using Reinforcement Learning. Proceedings Of The First International Workshop On Exploiting Artificial Intelligence Techniques For Data Management - Aidm'18. doi: 10.1145/3211954.3211956

Elghamrawy, S. (2016). An Adaptive Load-Balanced Partitioning Module in Cassandra Using Rendezvous Hashing. Advances In Intelligent Systems And Computing, 587-597. doi: 10.1007/978-3-319-48308-5_56

Elghamrawy, S., & Hassanien, A. (2017). A partitioning framework for Cassandra NoSQL database using Rendezvous hashing. The Journal Of Supercomputing, 73(10), 4444-4465. doi: 10.1007/s11227-017-2027-5

Fasolin, K., Fileto, R., Krugery, M., Kasterz, D., Ferreirax, M., & Cordeirox, R. et al. (2013). Efficient Execution of Conjunctive Complex Queries on Big Multimedia Databases. 2013 IEEE International Symposium On Multimedia. doi: 10.1109/ism.2013.112

Fetai, I., Murezzan, D., & Schuldt, H. (2015). Workload-driven adaptive data partitioning and distribution — The Cumulus approach. 2015 IEEE International Conference On *Big Data* (*Big Data*). doi: 10.1109/bigdata.2015.7363940

Foley, J., Kwan, P., & Welch, M. (2017). A Web-Based Infrastructure for the Assisted Annotation of Heritage Collections. Journal On Computing And Cultural Heritage, 10(3), 1-25. doi: 10.1145/3012287

Fung, C., Leung W., & Li, E. (2003). Efficient Query Execution Techniques in a 4DIS Video Database System for eLearning. Multimedia Tools and Applications, 20(1), pp.25-49.

Fung, C., Karlapalem, K., & Li, Q. (2003). Cost-driven vertical class partitioning for methods in object oriented databases. The VLDB Journal The International Journal On Very Large Data Bases, 12(3), 187-210. doi: 10.1007/s00778-002-0084-7

Getahun, F., Tekli, J., Atnafu, S. and Chbeir, R. (2007a). The use of semantic-based predicates implication to improve horizontal multimedia database fragmentation. Workshop on multimedia information retrieval on The many faces of multimedia semantics - MS '07 (pp. 29-38). Augsburg, Germany: ACM.

Getahun, F., Tekli, J., Atnafu, S., & Chbeir, R. (2007b). Towards Efficient Horizontal Multimedia Database Fragmentation using Semantic-based Predicates Implication. XXII Simpósio Brasileiro de Banco de Dados, 68-82.

Gu, X., Yang, X., Wang, W., Jin, Y., & Meng, D. (2012). CHAC: An Effective Attribute Clustering Algorithm for Large-Scale Data Processing. 2012 IEEE Seventh International Conference On Networking, Architecture, And Storage. doi: 10.1109/nas.2012.16

Guerrero, C.G., Domínguez, V.H., & Pech, J.P. (2014). UWE en sistema de recomendación de objetos de aprendizaje aplicando ingeniería web: un método en caso de estudio. Revista Latinoamericana de Ingenieria de Software 2(3):137-143

Hauglid, J., Ryeng, N., & Nørvåg, K. (2010). DYFRAM: dynamic fragmentation and replica management in distributed database systems. Distributed And Parallel Databases, 28(2-3), 157-185. doi: 10.1007/s10619-010-7068-1

Heni, H., & Gargouri, F. (2015). A Methodological Approach for Big Data Security: Application for NoSQL Data Stores. Neural Information Processing, 685-692. doi: 10.1007/978-3-319-26561-2_80

Herrmann, K., Voigt, H., & Lehner, W. (2014). Cinderella - Adaptive online partitioning of irregularly structured data. 2014 IEEE 30Th International Conference On Data Engineering Workshops. doi: 10.1109/icdew.2014.6818342

Hung, T., & Huang, C. (2012). A Dynamic Data Fragmentation and Distribution Strategy for Main-Memory Database Cluster. Advanced Materials Research, 490-495, 1231-1236. doi: 10.4028/www.scientific.net/amr.490-495.1231

Islam KHAN, S. (2016). Efficient Partitioning of Large Databases without Query Statistics. Database System Journals, 7(2), 34-53.

Jenni, K., Mandala, S., & Shahrizal Sunar, M. (2015). Content Based Image Retrieval Using Colour Strings Comparison. In 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15) (pp. 374-379.). Elsevier B.V.

Jindal, A., & Dittrich, J. (2012). Relax and Let the Database Do the Partitioning Online. Lecture Notes In Business Information Processing, 65-80. doi: 10.1007/978-3-642-33500-6_5

Papadias, D. (2009). Nearest Neighbor Query. In *Encyclopedia of Database Systems*. Boston, MA.: Springer.

Khan, A., Lee, C., Hamandawana, P., Park, S., & Kim, Y. (2018). A Robust Fault-Tolerant and Scalable Cluster-Wide Deduplication for Shared-Nothing Storage Systems. 2018 IEEE 26Th International Symposium On Modeling, Analysis, And Simulation Of Computer And Telecommunication Systems (MASCOTS). doi: 10.1109/mascots.2018.00016

Kulba, V., & Somov, S. (2020). Dynamic Fragment Allocation in Distributed System with Time-Varying Parameters of its Operation. In 2020 13th International Conference "Management of large-scale system development" (MLSD) (pp. 1-4). Moscow, Russia: IEEE.

Kumar, R., & Gupta, N. (2014). An extended approach to Non-Replicated dynamic fragment allocation in distributed database systems. 2014 International Conference On Issues And Challenges In Intelligent Computing Techniques (ICICT). doi: 10.1109/icicict.2014.6781394

Le, T., Kantere, V., & Orazio, L. (2019). Optimizing DICOM data management with NSGA-G. In International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data. Francia. Retrieved from https://hal.archives-ouvertes.fr/hal-02285736

Lew, M., Sebe, N., Djeraba, C., & Jain, R. (2006). Content-based multimedia information retrieval. ACM Transactions On Multimedia Computing, Communications, And Applications, 2(1), 1-19. doi: 10.1145/1126004.1126005

Li, L. and Gruenwald, L. (2013). Self-managing online partitioner for databases (SMOPD). Proceedings of the 17th International Database Engineering & Applications Symposium on - IDEAS '13.

Li, L., & Gruenwald, L. (2014). SMOPD-C: An autonomous vertical partitioning technique for distributed databases on cluster computers. Proceedings Of The 2014 IEEE 15Th International Conference On Information Reuse And Integration (IEEE IRI 2014). doi:

10.1109/iri.2014.7051887

Lim, L. (2013). Elastic data partitioning for cloud-based SQL processing systems. 2013 IEEE International Conference On Big Data. doi: 10.1109/bigdata.2013.6691766

Lin, W., & Veeravalli, B. (2003). Object management in distributed database systems for stationary and mobile computing environments: A competitive approach (12th ed., pp. 94-103). Ámsterdam, NL: Kluwer Academic Publishers.

Liroz-Gistau, M., Akbarinia, R., Pacitti, E., Porto, F., & Valduriez, P. (2012). Dynamic Workload-Based Partitioning for Large-Scale Databases. Lecture Notes In Computer Science, 183-190. https://doi.org/10.1007/978-3-642-32597-7_16

Liroz-Gistau, M., Akbarinia, R., Pacitti, E., Porto, F., & Valduriez, P. (2013). Dynamic Workload-Based Partitioning Algorithms for Continuously Growing Databases. Lecture Notes In Computer Science, 105-128. https://doi.org/10.1007/978-3-642-45315-1_5

Lu, Y., Bo, Y., He, W., & Nabatchian, A. (2018). Dynamic Partition Forest: An Efficient and Distributed Indexing Scheme for Similarity Search based on Hashing. 2018 IEEE International Conference On Big Data (Big Data). doi: 10.1109/bigdata.2018.8622321

Ma, H., Schewe, K., & Wang, Q. (2006). A heuristic approach to cost-efficient fragmentation and allocation of complex value databases. ADC '06: Proceedings Of The 17Th Australasian Database Conference, 49, 183-192.

Ma, H., Schewe, K., & Wang, Q. (2007). A Heuristic Approach to Cost-Efficient Derived Horizontal Fragmentation of Complex Value Databases. In ADC '07: Proceedings of the eighteenth conference on Australasian database (pp. 103-111). Palmerston North, New Zealand: ACM Digital Library.

Mettes, P., van Gemert, J., Cappallo, S., Mensink, T., & Snoek, C. (2015). Bag-of-Fragments. Proceedings Of The 5Th ACM On International Conference On Multimedia Retrieval -

ICMR '15. doi: 10.1145/2671188.2749404

Mourão, A., & Magalhães, J. (2017). Balancing search space partitions by sparse coding for distributed redundant media indexing and retrieval. International Journal Of Multimedia Information Retrieval, 7(1), 57-70. doi: 10.1007/s13735-017-0140-0

Mourão, A., & Magalhães, J. (2019). Towards Cloud Distributed Image Indexing by Sparse Hashing. Proceedings Of The 2019 On International Conference On Multimedia Retrieval. doi: 10.1145/3323873.3325046

Nam, Y., Hwang, E., & Kim, D. (2005). CLOVER: A Mobile Content-Based Leaf Image Retrieval System. Digital Libraries: Implementing Strategies And Sharing Experiences, 139-148. doi: 10.1007/11599517_16

Nguyen, T., & Lee, S. (2017). Optimizing MongoDB Using Multi-streamed SSD. In 7th International Conference on Emerging Databases (pp. 1-13). Suwon, Korea.

Olma, M., Karpathiotakis, M., Alagiannis, I., Athanassoulis, M., & Ailamaki, A. (2019). Adaptive partitioning and indexing for in situ query processing. The VLDB Journal, 29(1), 569-591. doi: 10.1007/s00778-019-00580-x

Oonhawat, B., & Nupairoj, N. (2017). Hotspot management strategy for real-time log data in MongoDB. 2017 19Th International Conference On Advanced Communication Technology (ICACT). doi: 10.23919/icact.2017.7890087

Özsu, T., & Valduriez, P. (2020). Principles of Distributed Database Systems (4th ed.). New Jersey: Springer.

Pèrez, J., Pazos, R., Frausto, J., Romero, D., & Cruz, L. (2000). Vertical Fragmentation and Allocation in Distributed Databases with Site Capacity Restrictions Using the Threshold Accepting Algorithm. Lecture Notes In Computer Science, 75-81. doi: 10.1007/10720076_7

Pinnecke, M., Campero Durand, G., Broneske, D., Zoun, R., & Saake, G. (2020). GridTables: A One-Size-Fits-Most H2TAP Data Store. Datenbank-Spektrum, 20(1), 43-56. doi: 10.1007/s13222-019-00330-x

Pinto, D., & Torres, G. (2002). On Dynamic Fragmentation of Distributed Databases Using Partial Replication. Advances In Systems Theory, Mathematical Methods And Applications, 208-211.

Rojas Ruiz, R. (2018). Desarrollo De Un Sistema Que Permita Realizar Consultas Basadas En Contenido En Una Base De Datos Multimedia. Tesis de maestria, Tecnológico Nacional de México.

Rodríguez, L., Xiaoou Li, Cuevas-Rasgado, A., & Garcia-Lamont, F. (2013). DYVEP: An active database system with vertical partitioning functionality. 2013 10Th IEEE INTERNATIONAL CONFERENCE ON NETWORKING, SENSING AND CONTROL (ICNSC), 457-462.

Rodríguez-Mazahua, Lisbeth, Alor-Hernández, Giner, Cervantes, Jair, López-Chau, Asdrúbal, & Sánchez-Cervantes, José Luis. (2016). A hybrid partitioning method for multimedia databases. DYNA, 83(198), 59-67.

Rodríguez-Mazahua, L., Alor-Hernández, G., Li, X., Cervantes, J., & López-Chau, A. (2017). Active rule base development for dynamic vertical partitioning of multimedia databases. Journal Of Intelligent Information Systems, 48(2), 421-451.

Rodríguez, L., & Li, X. (2011a). A Vertical Partitioning Algorithm for Distributed Multimedia Databases. En A. Hameurlain, S. W. Liddle, K.-D. Schewe, & X. Zhou (Eds.), Database and Expert Systems Applications (pp. 544-558). Springer Berlin Heidelberg.

Rodríguez-Mazahua, L., Alor-Hernández, G., Abud-Figueroa, M., & Peláez-Camarena, S. (2014). Horizontal Partitioning of Multimedia Databases Using Hierarchical Agglomerative Clustering. Nature-Inspired Computation And Machine Learning, 8857, 296-309. doi:

10.1007/978-3-319-13650-9_27

Rodríguez-Vaamonde, S., Torre-Bastida, A., & Garrote, E. (2014). Tecnologías Big Data para análisis y recuperación de imágenes web. El Profesional De La Informacion, 23(6), 567-574. doi: 10.3145/epi.2014.nov.02

Saad, S., Tekli, J., Chbeir, R., & Yetongnon, K. (2006). Towards Multimedia Fragmentation. Advances In Databases And Information Systems, 415-429.

Santos, N., & Masala, G. (2018). Big Data Security on Cloud Servers Using Data Fragmentation Technique and NoSQL Database. Intelligent Interactive Multimedia Systems And Services, 5-13. doi: 10.1007/978-3-319-92231-7_1

Santos, N., Ghita, B., & Masala, G. (2019). Enhancing Data Security in Cloud using Random Pattern Fragmentation and a Distributed NoSQL Database. 2019 IEEE International Conference On Systems, Man And Cybernetics (SMC). doi: 10.1109/smc.2019.8914454

Sauer, B., & Wei Hao. (2015). Horizontal cloud database partitioning with data mining techniques. 2015 12Th Annual IEEE Consumer Communications And Networking Conference (CCNC). doi: 10.1109/ccnc.2015.7158079

Schreiner, G.A., Duarte, D., & Mello, R.D. (2018). An Autonomous Hybrid Data Partition for NewSQL DBs. SBBD Companion.

Schreiner, G., Duarte, D., Dal Bianco, G., & Mello, R. (2019). A Hybrid Partitioning Strategy for NewSQL Databases. Proceedings Of The 21St International Conference On Information Integration And Web-Based Applications & Services. Doi: 10.1145/3366030.3366062

Schroeder, R., Penteado, R., & Hara, C. (2020). A data distribution model for RDF. Distributed And Parallel Databases, 39(1), 129-167. doi: 10.1007/s10619-020-07296-w

Serafini, M., Taft, R., Elmore, A., Pavlo, A., Aboulnaga, A., & Stonebraker, M. (2016). Clay fine-grained adaptive partitioning for general database schemas. Proceedings Of The VLDB Endowment, 10(4), 445-456. doi: 10.14778/3025111.3025125

Sharify, S., Lu, A., Chen, J., Bhattacharyya, A., Hashemi, A., Koudas, N., & Amza, C. (2019). An Improved Dynamic Vertical Partitioning Technique for Semi-Structured Data. 2019 IEEE International Symposium On Performance Analysis Of Systems And Software (ISPASS). doi: 10.1109/ispass.2019.00037

Sharma, S., & Bawa, S. (2019). CBDR: An efficient storage repository for cultural Big Data. Digital Scholarship In The Humanities. doi: 10.1093/llc/fqz083

Sleit, A., AlMobaidee, W., Al-Areqi, S., & Yahya, A. (2007). A Dynamic Object Fragmentation and Replication Algorithm In Distributed Database Systems. American Journal Of Applied Sciences, 4(8), 613-618. doi: 10.3844/ajassp.2007.613.618

Son, J., & Kim, M. (2004). An adaptable vertical partitioning method in distributed systems. Journal Of Systems And Software, 73(3), 551-561. doi: 10.1016/j.jss.2003.04.002

Springmann, M., Kopp, D., & Schuldt, H. (2010). QbS. Proceedings Of The International Conference On Multimedia Information Retrieval - MIR '10. doi: 10.1145/1743384.1743456

Torjmen, M., Pinel-Sauvagnat, K., & Boughanem, M. (2008). Towards a structure-based multimedia retrieval model. Proceeding Of The 1St ACM International Conference On Multimedia Information Retrieval - MIR '08. doi: 10.1145/1460096.1460153

Torjmen-Khemakhem, M., Pinel-Sauvagnat, K., & Boughanem, M. (2013). Investigating the document structure as a source of evidence for multimedia fragment retrieval. Information Processing & Management, 49(6), 1281-1300. doi: 10.1016/j.ipm.2013.06.001

Turcu, A., Palmieri, R., Ravindran, B., & Hirve, S. (2016). Automated Data Partitioning for Highly Scalable and Strongly Consistent Transactions. IEEE Transactions On Parallel And

Distributed Systems, 27(1), 106-118. doi: 10.1109/tpds.2015.2388448

Vazquez, J. (2000). A dynamic virtual fragmentation method for query recovery optimization. Proceedings 20Th International Conference Of The Chilean Computer Science Society. doi: 10.1109/sccc.2000.890391

Vogt, M., Stiemer, A., & Schuldt, H. (2018). Polypheny-DB: Towards a Distributed and Self-Adaptive Polystore. 2018 IEEE International Conference On Big Data (Big Data). doi: 10.1109/bigdata.2018.8622353

Wang, X., Fan, X., Chen, J., & Du, X. (2014). Automatic Data Distribution in Large-scale OLTP Applications. International Journal Of Database Theory And Application, 7(4), 37-46. doi: 10.14257/ijdta.2014.7.4.04

Wu, Q., Chen, C., & Jiang, Y. (2016). Multi-source heterogeneous Hakka culture heritage data management based on MongoDB. 2016 Fifth International Conference On Agro-Geoinformatics (Agro-Geoinformatics). doi: 10.1109/agro-geoinformatics.2016.7577628

Wycislik, L. (2015). Independent Data Partitioning in Oracle Databases for LOB Structures. Advances In Intelligent Systems And Computing, 687-696. doi: 10.1007/978-3-319-23437-3_59

Zar Lwin, N., & Naing, T. (2018). Non-Redundant Dynamic Fragment Allocation with Horizontal Partition in Distributed Database System. 2018 International Conference On Intelligent Informatics And Biomedical Sciences (ICIIBMS). doi: 10.1109/iciibms.2018.8550032

Zhao, W., Cheng, Y., & Rusu, F. (2015). Vertical partitioning for query processing over raw data. Proceedings Of The 27Th International Conference On Scientific And Statistical Database Management - SSDBM '15. doi: 10.1145/2791347.2791369

PRODUCTOS ACADÉMICOS

Felipe Castro Medina, Lisbeth Rodríguez Mazahua, Asdrúbal López Chau, Jair Cervantes, Giner Alor Hernández, Isaac Machorro Cano.

Application of dynamic fragmentation methods in multimedia databases: a review

Entropy 2020, 22(12), 1352; doi: 10.3390/e22121352

Páginas 1-25

Estado: publicado

Felipe Castro Medina, Lisbeth Rodríguez Mazahua, Asdrúbal López Chau, Jair Cervantes, Giner Alor Hernández, Mario Leoncio Arrioja-Rodríguez.

A new method of dynamic horizontal fragmentation for multimedia databases contemplating content-based queries

Electronics 2022, 11(288); ISSN: 2079-9292; doi: https://doi.org/10.3390/electronics11020288

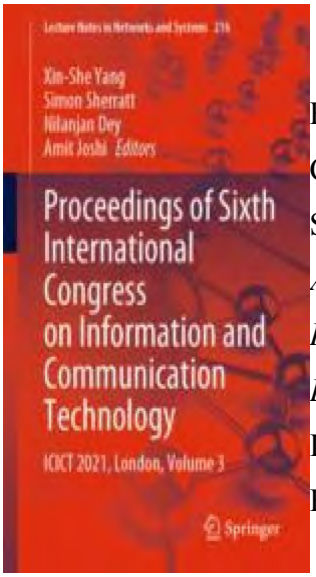
Estado: publicado

Felipe Castro Medina, Lisbeth Rodríguez Mazahua, Asdrúbal López Chau, María Antonieta Abud Figueroa, Giner Alor Hernández

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

IEEE Latin America Transactions 2020, 18(06); ISSN: 1548-0992; doi: 10.1109/TLA.2020.9099751

Estado: publicado

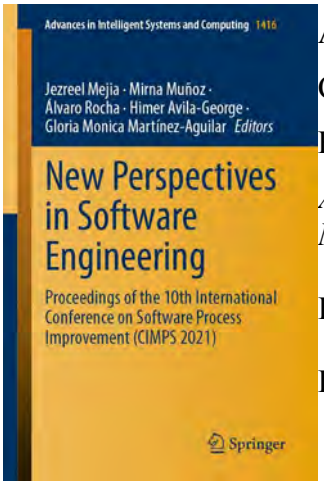


Felipe Castro-MedinaLisbeth Rodríguez-MazahuaAsdrúbal López-ChauGiner Alor-HernándezUlises Juárez-MartínezCuauhtémoc Sánchez-Ramírez

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

ISBN: 978-981-16-1781-2

Estado: publicado

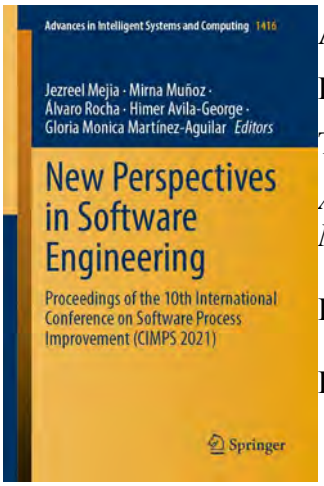


Abraham Castillo García, Lisbeth Rodríguez Mazahua, Felipe Castro Medina, Beatriz A. Olivares Zepahua, María A. Abud Figueroa.

A Review of Horizontal Fragmentation Methods Considering Multimedia Data and Dynamic Access Patterns

ISBN: 978-3-030-89908-0

Estado: publicado

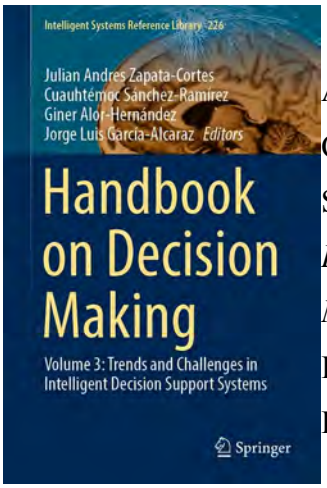


Aldo Osmar Ortiz Ballona, Lisbeth Rodríguez Mazahua, Asdrúbal López Chau, María Antonieta Abud Figueroa, Celia Romero Torres, Felipe Castro Medina

A Brief Review of Vertical Fragmentation Methods Considering Multimedia Databases and Content-Based Queries

ISBN: 978-3-030-89908-0

Estado: publicado

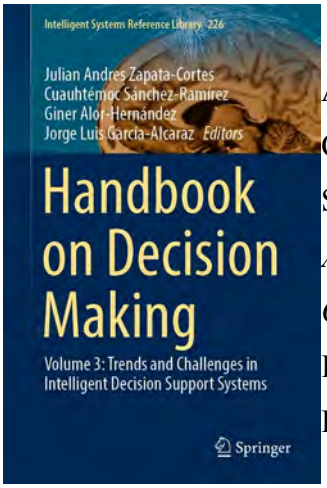


Abraham Castillo García, Lisbeth Rodríguez Mazahua, Felipe Castro Medina, Mario Leoncio Arrioja Rodríguez, José Luis Sánchez Cervantes & Nidia Rodríguez Mazahua

Design of a Dynamic Horizontal Fragmentation Method for Multimedia Databases

ISBN: 978-3-031-08246-7

Estado: publicado



Aldo Osmar Ortiz Ballona, Lisbeth Rodríguez Mazahua, Felipe Castro Medina, Mario Leoncio Arrioja Rodríguez, José Luis Sánchez Cervantes & Nidia Rodríguez Mazahua

A Vertical Fragmentation Method for Multimedia Databases Considering Content-Based Queries

ISBN: 978-3-031-08246-7

Estado: publicado

Oscar Crescencio Rico, Lisbeth Rodríguez Mazahua, Felipe Castro Medina, Giner Alor Hernández, José Luis Sánchez Cervantes

Método de Fragmentación Híbrida Dinámica para Bases de Datos Multimedia

DOI: https://doi.org/10.29057/icbi.v11iEspecial2.10716