DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

OPCIÓN I.- TESIS

TRABAJO PROFESIONAL

"FRAGMENTACIÓN HORIZONTAL DE ALMACENES DE DATOS USANDO ÁRBOLES DE DECISIÓN"

QUE PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS

DE LA INGENIERÍA

PRESENTA:

M.I.A NIDIA RODRÍGUEZ MAZAHUA

DIRECTOR DE TESIS:

DR. GINER ALOR HERNÁNDEZ

CODIRECTOR DE TESIS:

DR. ASDRÚBAL LÓPEZ CHAU

ORIZABA, VERACRUZ, MÉXICO MARZO 2023

Orizaba, Veracruz, 17/02/2023
Dependencia: División de Estudios de Posgrado e Investigación
Asunto: Autorización de Impresión
OPCION: I

C. NIDIA RODRÍGUEZ MAZAHUA
Candidata 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:

"Fragmentación Horizontal de Almacenes de Datos usando Árboles de Decisión"

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

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

ATENTAMENTE
Excelencia en Educación Tecnológica®
CIENCIA – TÉCNICA – CULTURA®

Cuauhtémoc Sánchez R.

DR. CUAUHTÉMOC SÁNCHEZ RAMÍREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN

Orizaba, Veracruz, febrero 15, de 2023.
Asunto: **Revisión de trabajo escrito**

C. CUAUHTÉMOC SÁNCHEZ RAMÍREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
P R E S E N T E.-

Los que suscriben, miembros del jurado, han realizado la revisión de la Tesis del (la) C.
NIDIA RODRÍGUEZ MAZAHUA

La cual lleva el título de:

"FRAGMENTACIÓN HORIZONTAL DE ALMACENES DE DATOS USANDO ÁRBOLES DE DECISIÓN"

Y concluyen que se acepta.

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

PRESIDENTE:	DR. GINER ALOR HERNÁNDEZ	FIRMA
SECRETARIO:	DRA. GUADALUPE LUNA SOLANO	FIRMA
VOCAL:	DRA. LAURA NELY SÁNCHEZ MORALES	FIRMA
VOCAL:	DRA. VIVIANA YAREL ROSALES MORALES	FIRMA
VOCAL:	DR. ASDRÚBAL LÓPEZ CHAU	FIRMA
VOCAL SUP.:	DRA. LISBETH RODRÍGUEZ MAZAHUA	FIRMA

TA-09 -18

Declaración de originalidad y cesión de derechos

Orizaba, Veracruz, el día 23 del mes de febrero del año 2023.

La que suscribe

C. Nidia Rodríguez Mazahua

Declaro que esta tesis, que tiene una extensión de 150 cuartillas, ha sido escrita por mí y constituye el registro escrito del trabajo de la tesis titulada

"Fragmentación Horizontal de Almacenes de datos usando Arboles de decisión"

del programa: Doctorado en Ciencias de la Ingeniería bajo la asesoría y dirección del Dr. Giner Alor Hernández y el Dr. Asdrúbal López Chau y no ha sido sometida en ninguna otra institución previamente.

Todos los datos y las referencias a materiales ya publicados están debidamente identificados con su respectivo crédito e incluidos en las notas bibliográficas y en las citas que se destacan como tal y, en los casos que así lo requieran, cuento con las debidas autorizaciones de quienes poseen los derechos patrimoniales. Por lo tanto, me hago responsable de cualquier litigio o reclamación relacionada con derechos de propiedad intelectual, exonerando de toda responsabilidad al Tecnológico Nacional de México campus Orizaba.

También declaro que, al presentar esta tesis, cedo los derechos del trabajo al Tecnológico Nacional de México campus Orizaba para su difusión, con fines académicos y de investigación, bajo las regulaciones propias de la institución y que si existe algún acuerdo de confidencialidad de la información lo haré saber en forma escrita para que se omitan las secciones correspondientes.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y del director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección: depi_orizaba@tecnm.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente de este.

Nombre y firma

Nidia Rodríguez Mazahua

Agradecimientos

Primero a Dios, porque si existo es por su misericordia.

A mi madre Celerina, por su apoyo incondicional, por brindarme las oportunidades y los medios que me permitieron llegar hasta este momento, por sus consejos y por impulsarme siempre a sacar lo mejor de mí.

A mi padre Juan, por enseñarme que la educación y el conocimiento son las armas más poderosas que tiene toda persona, por ser un apoyo constante a lo largo de mi etapa estudiantil y por estar para nosotros cuando más lo necesitamos.

A mi hermana Lisbeth por ser el mejor ejemplo para mí, por su apoyo, paciencia, amor y comprensión, por haber hecho posible este sueño que recorrimos juntas, finalmente podemos decir: ¡lo logramos! Y por demostrarme cada día que todo es posible con un poco de imaginación y perseverancia.

A mi esposo Marcos Fidel, por ser el compañero ideal, el cómplice perfecto en cada aventura.

Al miembro del jurado por su valiosa aportación a este trabajo de tesis.

Al Instituto Tecnológico de Orizaba por recibirme en sus instalaciones durante los últimos 8 años.

Contenido

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

El almacenamiento de datos provee arquitecturas y herramientas para que los ejecutivos comerciales organicen, entiendan y usen sistemáticamente los datos con el fin de tomar decisiones estratégicas. Los sistemas de almacenamiento de datos son herramientas útiles en un mundo competitivo y cambiante. Recientemente, varias compañías han invertido millones de dólares en el desarrollo de almacenes de datos para toda la empresa, por ello, el almacenamiento de datos representa una nueva estrategia de mercadotecnia indispensable y una manera de mantener a los consumidores, al aprender más sobre sus necesidades. Uno de los principales problemas que enfrentan los diseñadores de los Data Warehouse (DW) es la fragmentación. Existen varias técnicas de fragmentación horizontal que se basan en minería de datos, centrados en disminuir el tiempo de respuesta de las consultas y el costo de ejecución para hacer que el DW sea más eficiente. Sin embargo, se encontró que no existía alguna técnica de fragmentación horizontal que emplee un árbol de decisión para llevar a cabo la fragmentación en el DW. Dada la importancia de los árboles de decisión en la clasificación, ya que permiten obtener particiones puras (subconjuntos de tuplas) en un conjunto de datos utilizando medidas como el índice de Gini, la ganancia de información, la proporción de ganancia y, este trabajo de tesis tuvo como objetivo utilizar árboles de decisión en la fragmentación de un DW. Para hacer esto posible, se determinaron los requisitos necesarios para realizar la fragmentación horizontal mediante árboles de decisión y se diseñó el método de fragmentación llamado FTree, mismo que consiste en determinar las consultas OLAP (*On-line Analytical Processing*) más frecuentes, analizando los predicados utilizados por las consultas, y con base en esto construir el árbol de decisión para seleccionar el esquema de fragmentación horizontal. El método se implementó y validó en primera instancia utilizando SSB (*Star Schema Benchmark*), posteriormente se llevó a cabo la construcción de un Data Warehouse Turistíco donde FTree demostró su eficacia. Más tarde, se realizaron algunos experimentos para comparar los esquemas de fragmentación horizontal obtenidos por FTree con J48 versus el esquema seleccionado por otros clasificadores utilizando un modelo de costo. Finalmente, se comparó FTree contra un método del estado del arte.

ABSTRACT

Data Warehousing gives architectures and instruments for business executives to systematically prepare, understand, and utilize the data to make strategic decisions. Data warehouse systems are helpful tools in the fast-changing and combative world. In recent times, many firms have spent millions of dollars constructing company-wide data warehouses, for these, data warehousing is the most current essential marketing strategy and a method to retain clients by knowing more about their requirements. Fragmentation is one of the main troubles fronted by Data Warehouse (DW) designers. Some researchers have proposed data mining-based horizontal fragmentation techniques, which center attention on reducing the query response time and execution cost to conceive the DW more efficiently. Nevertheless, to the best of our knowledge there not exists a horizontal fragmentation method that utilizes a decision tree to achieve fragmentation. Given the relevance of decision trees in classification because they obtain pure partitions (subsets of instances) in a data set using measures such as the Gini Index, Information Gain, and Gain Ratio, the goal of this study has been to employ decision trees in the DW fragmentation. For this reason, the requirements for the implementation of horizontal fragmentation utilizing decision trees were defined, and we designed the fragmentation technique called FTree. It chooses the most executed OLAP (On-line Analytical Processing) queries, analyzes them to obtain their predicates, and according to this, it builds the decision tree to choose the horizontal fragment scheme. We employed the Star Schema Benchmark (SSB) in the first instance to validate that the design is appropriate. Subsequently, we developed a tourist data warehouse and the fragmentation technique was verified on it. The outcomes of the experiments demonstrated the efficacy of FTree. After that, a cost model was proposed to compare the horizontal fragmentation schemes obtained by FTree with J48 versus the scheme chosen by other classifiers. Finally, FTree was compared against a state-of–the-art-method.

INTRODUCCIÓN

El *Data Warehousing* (Almacenamiento de Datos) brinda arquitecturas y herramientas para que los empleados de negocios organicen, entiendan y usen sistemáticamente los datos para llevar a cabo la toma de decisiones estratégicas. Los sistemas de almacenamiento de datos son herramientas valiosas en el mundo competitivo y de rápida evolución de hoy. Ya que existe un aumento en la complejidad de los datos y dificultades de gestión en los mismos, el almacenamiento de datos atrajo interés para aplicaciones en la vida real, Redes sociales, Mercadotecnia, Transporte, Educación, Medicina, Seguros y Telecomunicaciones representan los principales escenarios en los que se reporta haberse desarrollado almacenes de datos (Kimball & Ross, 2013), (Bilal et al., 2016). Debido a esto, resulta crucial el diseño y la implementación de almacenes de datos en distintos ámbitos. Recientemente, varias empresas han invertido inmensas cantidades de dinero en el desarrollo de almacenes de datos para sus empresas, un ejemplo de esto es , Assistance Publique-Hôpitaux de Paris (AP-HP) (Daniel et al., 2020), la División de Sistemas de Aviación Ames de la Administración Nacional de Aeronáutica y del Espacio (NASA) (Melton et al., 2022), John Deere y DowDuPont (Janzen & Ristino, 2018).

Mucha gente considera que con el aumento de la competencia en la industria, el almacenamiento de datos es el arma de mercadotecnia indispensable más reciente y una manera de conservar a los clientes al aprender de una mejor manera sobre sus necesidades. Un almacén de datos es una colección de datos que está orientada al tema, integrada, que varía en el tiempo y no es volátil para apoyar en el proceso de toma de decisiones de los gerentes (Han et al., 2012). La información de las fuentes de datos operativos se integra mediante el almacenamiento de datos en un repositorio central para comenzar el proceso de análisis y extracción de información integrada y se utiliza principalmente en la toma de decisiones estratégicas mediante técnicas de procesamiento analítico en línea (OLAP). Cada vez más dispositivos se unen a Internet y comparten datos. El DW depende altamente de los dispositivos y los datos interrelacionados. Cuanto más interconectados están los dispositivos, más potente y útil es el DW.

Aunque el almacenamiento de datos se considera como una tecnología facilitadora de minería de datos, no siempre es necesario, debido a que la mayoría de minería de datos no accede al DW, pero las firmas que deciden invertir en ellos aplican minería de datos más general y más profundamente en la organización. Un ejemplo de esto es que si el DW integra registros de las ventas y facturación, así como de recursos humanos, es posible usarlo para encontrar patrones característicos de empleados efectivos (Provost & Fawcett, 2013).

Por otro lado, la fragmentación es una técnica de diseño de bases de datos distribuidas que consiste en dividir cada tabla de la base de datos en particiones más pequeñas y tratar cada partición como un objeto de la base de datos por separado. Hay tres opciones para esto: fragmentación horizontal, fragmentación vertical y fragmentación híbrida o mixta (Ozsu & Valduriez, 2020). En el caso de la fragmentación horizontal, cada tupla de la tabla debe incluirse como mínimo en una de las particiones, de forma que se reconstruya la tabla original en caso necesario. Mientras que, en la fragmentación vertical, cada fragmento incluye un subconjunto de los atributos de la relación, además de su clave primaria. La fragmentación mixta o híbrida aplica tanto la fragmentación horizontal como la vertical en una tabla. La fragmentación en bases de datos distribuidas es necesaria porque la mayoría de las veces los usuarios de la base de datos requieren subconjuntos de datos que son particiones horizontales y/o verticales de las tablas globales y porque es necesario procesar consultas que accedan estas particiones de forma óptima (Rodríguez, 2012).

En cuanto a los árboles de decisión, J.Ross Quinlan desarrolló un algoritmo de árboles de decisión conocido como ID3 (*Iterative Dichotomiser*) y presentó más tarde C4.5 (un sucesor de ID3), que se convirtió en un punto de referencia para algoritmos de aprendizaje supervisado. Un grupo de estadísticos (L. Breiman, J. Friedman, R. Olshen y C. Stone) publicó el libro CART *Classisication and Regression Trees*, (Arboles de Clasificación y Regresión), que describe la generación de árboles de decisión binarios. ID3 y CART surgieron de forma independiente entre sí aproximadamente al mismo tiempo, pero consideran un enfoque similar para aprender árboles de decisión a partir de tuplas de entrenamiento. Estos dos algoritmos fundamentales originaron una serie de trabajos sobre la inducción de árboles de decisión. ID3, C4.5, y CART adoptan un enfoque voraz en el que los árboles de decisión se construyen de manera divide y vencerás recursiva descendente. La mayoría de los algoritmos para la inducción de árboles de decisión también siguen un enfoque descendente, que comienza con un conjunto de tuplas de entrenamiento y sus etiquetas de clase asociadas. El conjunto de entrenamiento se divide de forma recursiva en subconjuntos más pequeños mientras que se construye el árbol. La construcción de clasificadores de árboles de decisión no necesita ningún conocimiento de dominio ni configuración de parámetros, y por estas razones, es apropiada para el descubrimiento de conocimiento exploratorio. Los árboles de decisión son capaces de tratar con datos multidimensionales. La representación del conocimiento adquirido en forma de árbol es intuitiva, simple y generalmente fácil de asimilar por los humanos. Los pasos de aprendizaje y clasificación de la inducción de árboles de decisión son simples y rápidos. Generalmente, los clasificadores de árboles de decisión tienen buena precisión. No obstante, la utilización exitosa depende de los datos disponibles. Los algoritmos de inducción de árboles de decisión se utilizan para la clasificación en aplicaciones diversas como análisis financiero, fabricación y producción, medicina, astronomía y biología molecular. Los árboles de decisión son la base de varios sistemas de inducción de reglas comerciales (Han et al., 2012).

El presente documento provee una descripción de los hallazgos obtenidos durante la realización del proyecto de tesis, donde primero se realizó un exhaustivo análisis de los métodos de fragmentación, más tarde se seleccionaron aquellos que se centraban en la fragmentación horizontal para data warehouse y sus modelos de costo.

Posteriormente se estudiaron los diferentes algoritmos de árboles de decisión para elegir el que se utilizó en el método de fragmentación obteniendo que el mejor fue C4.5 (J48 en Weka), más tarde se realizó la implementación del data warehouse turístico siguiendo la metodología establecida en (Kimball & Ross, 2016), finalmente, se implementó el método de fragmentación diseñado llamado FTree.

FTree comienza seleccionando las consultas OLAP relevantes, examinando los predicados empleados por la carga de trabajo y luego, de acuerdo con estos, construye el árbol de decisión para elegir el esquema de fragmentación horizontal. Para verificar la efectividad de FTree, se utilizó en primera instancia SSB (*Star Schema Benchmark*), posteriormente, se probó el método de fragmentación en el data warehouse turístico desarrollado, y se lograron resultados que demuestran el beneficio obtenido por FTree. Más tarde, se realizaron algunos experimentos para comparar el esquema de fragmentación horizontal (HFS) obtenido por FTree con J48 versus el esquema seleccionado por otros clasificadores, para esta comparación se utilizó un modelo de costos considerando que el costo de una consulta OLAP q_i consiste en el costo de tuplas irrelevantes accedidas (TIC) así como el costo de tuplas relevantes ubicadas en otros fragmentos (TRC).

Este documento contiene cuatro capítulos; el primero incluye la definición de los conceptos básicos del marco teórico; el segundo presenta un resumen de los trabajos relacionados y un análisis comparativo de estos. La explicación de cada paso de la metodología se incluye en el tercer capítulo. Finalmente, el cuarto capítulo contiene los resultados obtenidos con FTree.

PLANTEAMIENTO DEL PROBLEMA

Es bien conocido que la fragmentación es uno de los principales problemas que encaran los diseñadores de almacenes de datos, diversos estudios han desarollado métodos de fragmentación horizontal (Ettaoufik & Ouzzif, 2014; Letrache et al., 2019; Valentikova et al., 2012), vertical (Awad & Jebreen, 2015; Yeruva et al., 2015) e híbrida (Elmansouri et al., 2013; Hamdi et al., 2015) que se enfocan en la optimización del tiempo y costo de ejecución de las consultas para hacer más eficiente al *data warehouse*.

Sin embargo, se encontró que ninguna técnica de fragmentación horizontal utiliza un árbol de decisión para llevar a cabo la fragmentación, dada la importancia que tienen los árboles de decisión en la clasificación, ya que permiten crear particiones (subconjuntos de tuplas) puras en un conjunto de datos utilizando medidas como el índice Gini, la ganancia de información y la proporción de ganancia (Han, Kamber, & Pei, 2012), en este trabajo se utilizaron árboles de decisión en la fragmentación del *data warehouse*.

Para esto se analizó primero el estado del arte con respecto a los métodos de fragmentación horizontal existentes para *data warehouse*, posteriormente se determinaron los requerimientos necesarios para llevar a cabo la fragmentación horizontal usando árboles de decisión, se diseñó el método de fragmentación, el cual consiste en determinar las consultas OLAP más frecuentes, analizar los predicados utilizados por las consultas y con base en esto construir el árbol de decisión, a partir del cual se generan los fragmentos del *data warehouse*. Para verificar que el diseño sea correcto se utilizó el *benchmark* SSB. Posteriormente se implementó el método y se validó por medio de un caso de estudio en el turismo.

JUSTIFICACIÓN

La justificación de este trabajo de tesis se basa en que en la actualidad no existe un algoritmo de fragmentación horizontal para almacenes de datos que use árboles de decisión, se reportó el uso de un árbol de decisión para fragmentar horizontalmente una base de datos relacional (Curino et al., 2010) considerando cargas de trabajo OLTP (*Online Transaction Processing*, Procesamiento de Transacciones en Línea), sin embargo, no se toman en cuenta las características de un DW como el modelo multidimensional y las consultas OLAP. El método que se desarrolló tiene la capacidad de obtener esquemas de fragmentación para reducir el costo de ejecución de las consultas OLAP.

Es ampliamente aceptado que para aumentar el rendimiento de las consultas, las tablas del almacén de datos deben fragmentarse (Barkhordari & Niamanesh, 2018). Además, a medida que surge la computación distribuida, el almacén de datos también necesita cambiar su arquitectura. Es necesario fragmentar el esquema del almacén de datos en varios sitios distribuidos para cumplir con los escenarios económicos locales y globalizados (Yeruva et al., 2015). Por otra parte, dado que la cantidad de datos almacenados crece continuamente y su procesamiento conlleva muchas complicaciones, las bases de datos analíticas (OLAP) basadas en los almacenes de datos ganan rápidamente popularidad en el procesamiento de datos analíticos. Además de muchos de los beneficios, el almacén de datos centralizado (CDW) presenta muchos problemas, como el rendimiento del sistema, el rápido crecimiento de los datos, la confiabilidad del sistema, los costos de mantenimiento, entre otros. Esto simplemente condujo a un concepto de almacén de datos distribuidos (DDW) para incrementar la fiabilidad general del sistema y la disponibilidad de datos (Valentikova et al., 2012).

La fragmentación de datos tiene como objetivo minimizar el costo y el tiempo de ejecución de la carga de trabajo OLAP mediante el procesamiento simultáneo de diferentes consultas en nodos distintos (paralelismo interconsultas), y la ejecución de la misma consulta al mismo tiempo en múltiples nodos (paralelismo intraconsultas); además, reduce el costo de mantenimiento del almacén de datos mediante operaciones de actualización específicas y paralelas; y el costo de propiedad de un almacén de datos mediante el uso de hardware básico con una arquitectura de nada compartido en lugar de costosas arquitecturas de servidor (Cuzzocrea & Moussa, 2013). Con la explosión del tamaño de las aplicaciones de almacenamiento de datos, la partición horizontal de datos está bien adaptada para reducir el costo de las complejas consultas OLAP y la capacidad de administración del almacén (Bellatreche et al., 2013).

Por otro lado, este trabajo es importante para el turismo, ya que, hace ya varias décadas, el turismo ha vivido un continuo crecimiento y una marcada diversificación, lo que lo ha llevado a ser considerado uno de los sectores económicos que crecen con más velocidad en el mundo. Actualmente, el volumen de negocio del turismo es igual o todavía superior al

de las exportaciones de petróleo, automóviles o productos alimentarios. La actividad turística se convirtió en uno de los actores del comercio internacional más importantes, y además se considera como una de las principales fuentes de ingresos de diversos países que están en desarrollo, (OMT, 2022). Sin embargo, para corregir cualquier desviación en el desempeño, los gerentes en la industria del turismo a menudo necesitan informes de análisis oportunos para medir y monitorizar la tasa de desempeño, el aumento y la disminución en el número de turistas, noches de turismo y el porcentaje de ocupaciones de hoteles, visitas a lugares de monumentos y los ingresos totales del sector turístico a nivel nacional. También necesitan un análisis oportuno de los informes que ayudan a tomar decisiones a largo plazo. Se observó que, en la mayoría de los informes y análisis, se dedicó tiempo a recopilar datos de los diversos sistemas antes de realizar el análisis. Hoy en día los gerentes desean y necesitan más información, pero los analistas solo proporcionan información mínima a un costo elevado. Para analizar un problema o una situación de manera más eficiente, se necesita un almacén de datos.

La evaluación de la actividad actual de la industria turística se realiza a través de estadísticas. Como el número de visitantes (total y área de origen - para turistas extranjeros), gasto en hoteles y establecimientos similares, la duración promedio de las estancias, entre otras. Estos indicadores solo permiten un análisis de la actividad turística, pero sin ofrecer la posibilidad de mejorar las decisiones que se toman en el turismo o para aquellas áreas con impacto directo o indirecto en el turismo. Un almacén de datos central y homogéneo es un requisito previo para los análisis que abarcan todos los destinos y partes interesadas. Un sistema de soporte de decisiones basado en el almacenamiento de datos en el turismo mejora la toma de decisiones con información oportuna, precisa y completa. Por lo tanto, este modo de organización de datos proporciona un buen medio para analizar, evaluar o predecir el desarrollo del mercado y la economía del turismo (Danubianu et al., 2009), (Höpken et al., 2015).

Finalmente, con el desarrollo de este proyecto de investigación se verán beneficiados no solo el sector turístico que en este caso será el caso de estudio, sino también los diseñadores de almacenes de datos que contarán con un método de fragmentación novedoso que brindará ventajas en el costo de ejecución de las consultas del DW y, por lo tanto, procesará las consultas OLAP de manera más eficiente. Además, los investigadores del área de Bases de Datos se beneficiarán debido a que utilizarán este método para compararlo con los algoritmos que desarrollen.

HIPÓTESIS

El método de fragmentación horizontal utiliza árboles de decisión para obtener esquemas de fragmentación que permiten reducir el costo de ejecución de las consultas en un DW.

OBJETIVO GENERAL

Desarrollar un método de fragmentación horizontal que utilice árboles de decisión para obtener esquemas de fragmentación que logren la optimización de consultas en un DW.

OBJETIVOS ESPECÍFICOS

- ✓ Analizar el estado del arte de los métodos de fragmentación horizontal para almacenes de datos.
- ✓ Estudiar los modelos de costo para comparar distintos esquemas de fragmentación horizontal en almacenes de datos.
- ✓ Realizar un análisis comparativo de los métodos de fragmentación horizontal y modelos de costos para almacenes de datos.
- ✓ Analizar los algoritmos de árboles de decisión para encontrar el más adecuado para desarrollar el método de fragmentación.
- ✓ Diseñar el método de fragmentación horizontal basado en árboles de decisión para DW.
- ✓ Implementar el método de fragmentación horizontal basado en árboles de decisión.
- ✓ Desarrollar un DW para el sector turístico en México.
- ✓ Evaluar el método de fragmentación horizontal en el DW Turístico.

APORTACIÓN AL CONOCIMIENTO

La aportación más importante de este trabajo es el desarrollo de un nuevo algoritmo de fragmentación para almacenes de datos que está basado en árboles de decisión que optimiza el costo de ejecución de consultas OLAP.

CAPÍTULO 1 ANTECEDENTES

Este capítulo presenta los conceptos principales más relevantes para este tema de tesis. Se incluye el marco teórico, el planteamiento del problema, los objetivos, la hipótesis, la justificación y la aportación al conocimiento.

1.1 Marco teórico

1.1.1. Data Warehouse

Un Data Warehouse (DW) es un silo de datos centralizados para una empresa que contiene datos fusionados, limpios e históricos (Sarka et al., 2014). Aunque el almacenamiento de datos se considera una tecnología facilitadora de minería de datos, no siempre es necesario, debido a que la mayoría de minería de datos no accede al DW, pero las firmas que deciden invertir en ellos aplican minería de datos más general y más profundamente en la organización. Un ejemplo de esto es que si el DW integra registros de las ventas y facturación, así como de recursos humanos, es posible usarlo para encontrar patrones característicos de empleados efectivos (Provost & Fawcett, 2013).

1.1.2. Modelo multidimensional

El modelo multidimensional es una técnica utilizada para presentar datos analíticos porque entrega datos que son comprensibles para los usuarios comerciales y logra un rendimiento de consulta rápido (Kimball & Ross, 2013).

Los esquemas del DW se simplifican y por lo tanto son más adecuados para generar reportes que los esquemas relacionales normalizados. Para un DW se usa generalmente un tipo especial de diseño lógico llamado esquema de estrella o una variante de este llamado esquema de copo de nieve. Las tablas del esquema de estrella o copo de nieve se dividen en tablas de dimensiones y tablas de hechos (Sarka et al., 2014).

1.1.3. Esquema de estrella

En este modelo, cada grupo de dimensiones se coloca en una tabla de dimensiones, los hechos se colocan en una tabla de hechos. El resultado es un esquema en estrella donde la tabla de hechos está en el centro rodeada por las tablas de dimensiones, las tablas de dimensiones contienen datos cualitativos representados en un gran número de atributos. Estos datos cualitativos apoyan muchos procesos de análisis. Por otro lado, la tabla de hechos tiene un número importante de instancias, cada tupla en la tabla de hechos tiene dos tipos de atributos:

- Claves externas que hacen referencia a la tabla de dimensiones.

- Un conjunto de medidas que se agregan para realizar operaciones.

La tabla de hechos generalmente está normalizada, pero las de dimensiones no lo están; las consultas que se utilizan en este esquema se denominan "Consultas de reunión de estrellas" (Sidi et al., 2016).

1.1.4. Esquema de copo de nieve

El esquema de copo de nieve refleja las jerarquías asociadas con cada dimensión. Cada tabla de dimensión se divide en una pluralidad de jerarquías. Este diagrama normaliza dimensiones, reduciendo el tamaño de cada una de las conexiones, permitiendo así formalizar el concepto de jerarquía dentro de una dimensión.

Las tablas que representan la jerarquía más fina están directamente vinculadas a la tabla de hechos. Las tablas que representan otras jerarquías están vinculadas entre sí según su nivel en la jerarquía (Sidi et al., 2016).

1.1.5. Esquema de constelación de hechos

Aplicaciones sofisticadas requieren múltiples tablas de hechos para compartir tablas de dimensiones. Este tipo de esquema se visualiza como una colección de estrellas y por lo tanto es llamado un esquema de galaxia o una constelación de hechos (Han et al., 2012).

1.1.6. OLAP

El término Procesamiento analítico en línea (OLAP) se introdujo en 1993 por E. Codd. Este modelo constituye un marco del sistema de apoyo a la toma de decisiones que tiene la capacidad de calcular, consolidar, ver y analizar datos de acuerdo con múltiples dimensiones. OLAP depende en gran medida de un modelo de datos conocido como bases de datos multidimensionales (MDB). En comparación con las bases de datos relacionales, MDB aumenta el rendimiento almacenando datos agregados y mejorando la presentación de datos. De hecho, los sistemas MDB ofrecen las siguientes tres ventajas:

- **Presentación:** MDB mejora la presentación de datos y navegación mediante vistas intuitivas similares a hojas de cálculo que son difícil de generar utilizando tecnologías SQL.
- **Facilidad de mantenimiento:** las bases de datos multidimensionales son muy fáciles de mantener, porque los datos se almacenan en estas de la misma manera en que se ven, es decir, de acuerdo con sus atributos, no se requieren recursos computacionales adicionales para el procesamiento de consultas.
- **Rendimiento:** los sistemas MDB aumentan el rendimiento. En efecto, a través de operaciones OLAP los sistemas MDB permiten navegar intuitivamente al analista por la base de datos y acceder muy rápido a un subconjunto particular de los datos.
- El entorno de BI (*Bussines Intelligence*) sigue creciendo y los analistas de información han adoptado los conceptos y tecnologías OLAP. Según las investigaciones de los observadores del mercado, como Pringle & Company y Gartner, el mercado para plataformas de BI seguirá siendo uno de los mercados de

software de más rápido crecimiento en la mayoría de las regiones (Cuzzocrea & Moussa, 2013) .

1.1.6.1. Cubo OLAP

Un esquema MDB contiene un modelo lógico que consta de Cubos OLAP. Un Cubo OLAP se caracteriza por una tabla de hechos, un conjunto de dimensiones y un conjunto de medidas.

- **Hechos:** una tabla de hechos consta de hechos de un proceso empresarial. Por ejemplo, para una empresa que vende productos a los consumidores, cada venta es un hecho que ocurre y la tabla de hechos se utiliza para registrar estos hechos.
- **Medidas:** cada medida cuantifica elementos como costos, ingresos o unidades de servicio, que se cuentan, se resumen o agregan. Para ello, las medidas utilizan funciones agregadas apropiadas tales como: suma, promedio edad, recuento, recuento distinto, entre otras.
- **Dimensiones:** las dimensiones son variables por las que las medidas se resumen. Cada dimensión está compuesta de niveles. Los niveles de una dimensión se organizan como jerarquía, es decir, un conjunto de relaciones padre-hijo, típicamente donde un miembro padre resume a sus hijos. Por ejemplo, algunos niveles comunes de una dimensión de tiempo son: año, trimestre, mes, semana y día. Cada nivel contiene propiedades. Por ejemplo, considerando la dimensión geográfica de las ventas según, país, ciudad, tienda. El nivel de país se describe por propiedades como área, moneda, población, ubicación y zona horaria del país (Cuzzocrea & Moussa, 2013).

1.1.7. Benchmarks

A continuación, se presentan algunos de los *benchmarks* más populares en la literatura para analizar métodos de fragmentación.

TPC-H: El TPC Benchmark ™ H (TPC-H) es un punto de referencia de soporte a la toma de decisiones. Está compuesto por un conjunto de consultas *ad hoc* orientadas al negocio y modificaciones de datos simultáneas. Las consultas y los datos que pueblan la base de datos se eligieron para contar con una gran relevancia en toda la industria. Este punto de referencia ilustra los sistemas de soporte de decisiones que analizan grandes cantidades de datos, ejecutan consultas complejas y brindan respuestas a interrogantes comerciales críticas. La métrica de rendimiento informada por TPC-H se denomina Métrica de rendimiento compuesta de consultas por hora de TPC-H (QphH @ Size) y refleja múltiples aspectos de la capacidad que tiene el sistema para procesar consultas. Los aspectosse integran por el tamaño de la base de datos seleccionada contra la cual se ejecutan las consultas, la potencia de procesamiento de consultas cuando las consultas se envían por un solo flujo y el rendimiento de las consultas cuando las consultas se envían por múltiples

usuarios concurrentes. La métrica de costo / rendimiento de TPC-H se expresa como $ / QphH @ Size, (*TPC-H Homepage*, 2020).

SSB: El punto de referencia *Star Schema*, o SSB, se diseñó para evaluar el rendimiento del sistema de base de datos de las consultas del almacén de datos del esquema en estrella. El esquema para SSB se basa en el punto de referencia TPC-H, pero en una forma muy modificada. Se considera que los detalles de la modificación son instructivos para responder una pregunta importante: dado un esquema de base de datos que no está en forma de esquema en estrella, ¿cómo se transforma a forma de esquema en estrella sin perder información importante de la consulta? El SSB se utiliza para medir una serie de productos de bases de datos comerciales importantes en Linux para evaluar un nuevo producto (O'neil et al., 2009).

APB-1: El Consejo OLAP patrocinó el desarrollo de un punto de referencia de procesamiento analítico, el APB-1. El punto de referencia simula una situación empresarial realista de procesamiento analítico en línea (OLAP) que ejecuta software basado en servidor. El objetivo del APB-1 es medir el rendimiento OLAP general de un servidor en lugar del rendimiento de tareas específicas. Para garantizar la relevancia del APB-1 para los entornos comerciales reales, las operaciones realizadas en la base de datos se eligen cuidadosamente para reflejar las operaciones comerciales comunes. Estas operaciones incluyen lo siguiente:

- Carga masiva de datos de fuentes de datos internas o externas
- Carga incremental de datos de sistemas operativos
- Agregación de datos de nivel de entrada a lo largo de jerarquías
- Cálculo de nuevos datos basados en modelos comerciales
- Análisis de series temporales
- Consultas con un alto grado de complejidad
- Desglose de las jerarquías
- Consultas *ad hoc*
- Varias sesiones en línea

Las aplicaciones OLAP exitosas deben proporcionar "información justo a tiempo". La clave para una toma de decisiones eficaz es tener la información adecuada en el momento adecuado. Juzgar la capacidad de un servidor para lograr este objetivo es más que simplemente medir el rendimiento de procesamiento de un servidor OLAP. Su capacidad para representar relaciones comerciales complejas y responder a los requisitos comerciales cambiantes es igualmente importante.

Si bien el APB-1 no intenta medir la capacidad de un sistema para responder al cambio, la cantidad y la claridad del código de programación se utilizan como una medida cualitativa de la capacidad de respuesta. Es necesario que los usuarios del APB-1 sean capaces de

evaluar una solución dada tanto en términos de su adecuación cuantitativa como cualitativa a la tarea. Por esta razón, la publicación de los resultados del *benchmark* APB-1 debe incluir el esquema de la base de datos y el código requerido para ejecutar el *benchmark* (*OLAP Benchmark Study*, 2020).

TPC-DS: El TPC *Benchmark* DS (TPC-DS) es un punto de referencia de soporte de decisiones que modela varios aspectos generalmente aplicables de un sistema de este tipo, incluidas las consultas y el mantenimiento de datos. TPC-DS proporciona una evaluación representativa del desempeño como un sistema de apoyo a las decisiones de propósito general. Un resultado de referencia mide el tiempo de respuesta de las consultas en el modo de usuario único, multiusuario y el rendimiento del mantenimiento de datos para un hardware, sistema operativo y una configuración del sistema de procesamiento de datos determinados bajo una carga de trabajo controlada, compleja y de soporte de decisiones multiusuario. El propósito de los puntos de referencia de TPC es proporcionar datos de rendimiento objetivos y relevantes a los usuarios de la industria. TPC-DS Versión 2 permite que las tecnologías emergentes, como los sistemas Big Data, ejecuten el punto de referencia (*TPC-DS Homepage*, 2020).

1.1.8. Fragmentación

La fragmentación es una técnica empleada en el diseño de bases de datos distribuidas que consiste en dividir cada tabla de la base de datos en particiones más pequeñas y manejar cada uno de estos fragmentos como un objeto de la base de datos por separado. Existen tres alternativa: la fragmentación horizontal, fragmentación vertical y fragmentación mixta o también llamada híbrida. En el caso de la fragmentación horizontal, cada una de las tuplas de la tabla tiene que pertenecer mínimo a una de las particiones, de forma que sea posible reconstruir la relación original si se necesitara. Mientras que en la fragmentación vertical, cada fragmento tiene un subconjunto de los atributos de la tabla y también su clave primaria. Finalmente, en la fragmentación mixta o híbrida se aplica al mismo tiempo la fragmentación horizontal y vertical en una tabla. Es necesaria la fragmentación en bases de datos distribuidas debido a que los usuarios de la base de datos por lo general requieren subconjuntos de datos (fragmentos) de las tablas globales y también por la necesidad de procesar consultas que recuperen estas particiones de forma óptima (Ozsu & Valduriez, 2020).

1.1.8.1. Tipos de fragmentación horizontal

La fragmentación horizontal divide una relación (tabla) de base de datos en subconjuntos de tuplas. Hay dos versiones: primaria y derivada. La fragmentación horizontal primaria de una tabla se realiza utilizando predicados que se definen en esa tabla. La fragmentación horizontal derivada, por otro lado, es la partición de una tabla que resulta de predicados definidos en otra tabla (Ozsu & Valduriez, 2020).

Vertical

La fragmentación vertical de una relación R produce fragmentos R_1, R_2,..... R_r, cada fragmento incluye un subconjunto de los atributos de *R*, además de la clave principal de *R*. El objetivo de la fragmentación vertical es dividir una tabla en un conjunto de tablas más pequeñas para que muchas de las aplicaciones de usuario se ejecuten en un solo fragmento. La fragmentación vertical es indiscutiblemente más complicada que la horizontal debido a la cantidad total de alternativas que están disponibles (Ozsu & Valduriez, 2020).

Híbrida

La mayoría de las veces, la partición horizontal o vertical de un esquema de base de datos no bastará para satisfacer los requisitos de las aplicaciones del usuario. En estos casos, es posible que una fragmentación vertical vaya seguida de una horizontal, o viceversa, lo que genera una partición estructurada en árbol. Debido a que los dos tipos de estrategias de fragmentación se aplican uno detrás de otro, esta alternativa se denomina fragmentación híbrida (Ozsu & Valduriez, 2020).

1.1.8.2. Reglas de corrección de la fragmentación

Se refiere a las tres reglas que se aplican durante la fragmentación, que juntas se aseguran de que la base de datos no sufra cambios semánticos en el proceso de la fragmentación.

Completitud. Si una instancia de relación *R* se descompone en fragmentos R_1, R_2,.... R_n, cada elemento de datos que se encuentra en *R* de igual forma se encuentra en uno o más de R_i. Esta propiedad es de suma importancia en la fragmentación, pues se asegura que los datos en una relación global se mapeen en fragmentos sin ninguna pérdida (Ozsu & Valduriez, 2020).

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

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

El operador será diferente para las distintas formas de fragmentación; en el caso de la horizontal es la unión y para la vertical es la reunión. La capacidad de reconstrucción de la tabla a partir de sus particiones garantiza que se conserven las restricciones definidas en los datos en forma de dependencias.

Disyunción. Si una relación *R* se descompone horizontalmente en los fragmentos $F_R = \{R_1, R_2, ..., R_n\}$ y el dato d_i, está en R_j, no está en ningún otro fragmento R_k ($k \neq j$). Este criterio se asegura de que las particiones horizontales estén disjuntas. Si la tabla R se descompone verticalmente, sus atributos de clave primaria se repiten normalmente en todas sus particiones.

1.1.9. Fragmentación horizontal en almacenes de datos

En las últimas décadas, se propusieron varios métodos de fragmentación horizontal para almacenes de datos porque esta técnica es capaz de reducir el tiempo de respuesta de consultas OLAP (*On-line Analytical Processing*) y tiene ventajas significativas durante las operaciones de carga y mantenimiento de tablas (Kimball et al., 2008).

En el contexto relacional, la partición horizontal derivada se conoce como más adecuada para DW porque tiene en cuenta los requisitos de consulta OLAP y evita cálculos innecesarios en las operaciones de combinación (Mahboubi & Darmont, 2008). Por lo tanto, la mayoría de las técnicas de fragmentación horizontal para almacenes de datos se derivan, es decir, la partición de la tabla de hechos se desarrolla de acuerdo con el esquema de fragmentación de una tabla de dimensión seleccionada.

1.1.10. Minería de datos

La minería de datos es entendida como el proceso de descubrir patrones y conocimientos interesantes a partir de inmensas cantidades de datos. Ejemplos de fuentes de datos incluyen bases de datos, almacenes de datos, la Web, así como otros repositorios de información o datos que se transmiten a los sistemas de forma dinámica (Han et al., 2012).

1.1.10.1. KDD

Mucha gente trata a la Minería de Datos como un sinónimo para otro término popular, Descubrimiento de Conocimiento a partir de los Datos o KDD, mientras otros ven a la minería de datos como un paso esencial en el proceso de descubrir conocimiento. Los pasos para el proceso de descubrimiento de conocimiento son los siguientes:

1. Limpieza de los datos (remover ruido y datos inconsistentes).
2. Integración de datos (donde se combinan múltiples fuentes de datos).
3. Selección de datos (donde los datos relevantes para la tarea de análisis se obtienen de la base de datos).
4. Transformación de datos (donde los datos se transforman y consolidan en formas apropiadas para realizar la minería por medio de la ejecución de operaciones de resumen o agregación).
5. Minería de datos (un proceso esencial donde se aplican métodos inteligentes para extraer patrones de datos).
6. Evaluación de patrones (identificar los patrones realmente interesantes que representan el conocimiento con base en algunas medidas de interés).
7. Presentación del conocimiento (donde se usan técnicas de visualización y representación del conocimiento para mostrar el conocimiento minado al usuario) (Han et al., 2012).

1.1.10.2. Tareas de la Minería de Datos

Algunas tareas importantes de la Minería de Datos se describen brevemente en este apartado.

Clasificación: es el proceso de hallar un modelo (o función) que describe y distingue clases o conceptos de datos. El modelo se construye con base en el análisis de un conjunto de datos de entrenamiento (esto es, objetos de datos que tienen una etiqueta de clase que es conocida). El modelo se utiliza para predecir la clase de objetos cuya etiqueta de clase se desconoce.

Regresión: el análisis de regresión es una metodología estadística que se emplea frecuentemente para predicción numérica, no obstante, existen otros métodos. La predicción de igual manera incluye la identificación de las tendencias de distribución con base en los datos que están disponibles.

Agrupamiento: analiza objetos de datos sin consultar una etiqueta de clase conocida. En varios casos, es posible que las etiquetas de clase simplemente no existan al inicio. Se puede usar el agrupamiento para generar etiquetas de clase para un grupo de datos. Los objetos se agrupan de acuerdo con el principio de maximizar la similitud dentro de las clases y minimizar la similitud entre ellas.

Asociación: los patrones frecuentes son aquellos que ocurren frecuentemente en los datos. Existen varios tipos de patrones frecuentes, incluidos conjuntos de elementos frecuentes, subsecuencias frecuentes (conocidas como patrones secuenciales) y subestructuras frecuentes. Un *conjunto de elementos frecuente* normalmente hace referencia a un conjunto de elementos que aparecen juntos frecuentemente en un conjunto de datos transaccionales, por ejemplo: la leche y el pan, que se compran juntos frecuentemente en establecimientos de abarrotes por una gran cantidad de clientes (Han et al., 2012).

1.1.10.3. Modelos de clasificación

Naïve Bayes: Naïve Bayes es un modelo de clasificación que emplea la teoría de la probabilidad para indicar la relación entre los atributos y las etiquetas de clase, este modelo es uno de los modelos de clasificación probabilística más fáciles y más utilizados. Naïve Bayes realiza una suposición simplificadora sobre las probabilidades condicionales de clase, entendida como suposición Naïve Bayes, con el empleo de esto se obtienen estimaciones confiables de probabilidades condicionales de clase, aun cuando la cantidad de atributos es grande [28].

Perceptrón multicapa: El perceptrón multicapa es un tipo de red neuronal artificial (ANN) y se considera un enfoque de clasificación poderoso que encuentra límites de decisión realmente complejos y no lineales que provienen únicamente de los datos. En Multi-layer Perceptron, el concepto básico de un perceptrón se generaliza a arquitecturas más complicadas de nodos que aprenden límites de decisión no lineales [28]. En la siguiente sección, se describen los hallazgos principales de la evaluación de FTree.

1.1.11. Árboles de decisión

En cuanto a los árboles de decisión, J.Ross Quinlan desarrolló un algoritmo de árboles de decisión conocido como ID3 (*Iterative Dichotomiser*) y presentó más tarde C4.5 (un sucesor de ID3), que se volvió un punto de referencia para algoritmos de aprendizaje

supervisado. Los estadísticos (L. Breiman, J. Friedman, R. Olshen y C. Stone) publicaron el libro CART *Classisication and Regression Trees*, (Árboles de Clasificación y Regresión), que describe la generación de árboles de decisión binarios. ID3 y CART surgieron de forma independiente entre sí aproximadamente al mismo tiempo, pero siguen un enfoque parecido para aprender árboles de decisión a partir de tuplas de entrenamiento. Estos dos algoritmos fundamentales originaron una serie de trabajos sobre la inducción de árboles de decisión. ID3, C4.5, y CART adoptan un enfoque voraz en el que los árboles de decisión se construyen de manera divide y vencerás recursiva descendente.

La mayoría de los algoritmos para la inducción de árboles de decisión de igual manera siguen un enfoque descendente, que comienza con un conjunto de tuplas de entrenamiento y sus etiquetas de clase asociadas. El conjunto de entrenamiento se divide de forma recursiva en subconjuntos más pequeños conforme se construye el árbol. La construcción de clasificadores de árboles de decisión no necesita ningún conocimiento de dominio ni configuración de parámetros, y por este motivo es apropiado para el descubrimiento de conocimiento exploratorio. Los árboles de decisión son capaces de tratar con datos multidimensionales. La representación del conocimiento obtenido en forma de árbol es intuitiva y por lo general fácil de asimilar por las personas. Los pasos de aprendizaje y clasificación de la inducción de árboles de decisión son simples y rápidos. Generalmente, los clasificadores de árboles de decisión presentan buena precisión. No obstante, el éxito de su uso depende de los datos disponibles. Los algoritmos de inducción de árboles de decisión se utilizan para la clasificación en diversas aplicaciones como medicina, fabricación y producción, astronomía, análisis financiero y biología molecular (Han et al., 2012).

1.1.11.1. Algoritmos de inducción de árboles de decisión

La descripción de algunos algoritmos para la construcción de árboles de decisión se muestra enseguida.

Hoeffding Tree: es un algoritmo de inducción de árbol de decisión incremental, con la capacidad de aprender de flujos de datos masivos, asumiendo que los ejemplos de generación de distribución no cambian con el tiempo. Los árboles Hoeffding emplean a su favor el hecho de que es posible que una pequeña muestra a menudo sea suficiente para seleccionar un atributo de división óptimo. Esta idea está basada matemáticamente en el límite de Hoeffding, que cuantifica el número de observaciones que son necesarias para estimar algunas estadísticas dentro de una precisión prescrita (Hulten et al., 2001).

LMT: clasificador para la construcción de árboles de clasificación con funciones de regresión logística en las hojas. El algoritmo trata con variables objetivo binarias y de clases múltiples, atributos numéricos y nominales y valores faltantes (Saeh et al., 2016).

J48: C4.5 Decision Tree es uno de los enfoques más utilizados del mundo real. En C4.5, el clasificador aprendido está representado por un árbol de decisión como conjuntos de reglas

si-entonces para mejorar la legibilidad humana. El árbol de decisión es fácil de entender e interpretar; además, maneja datos nominales y categóricos y funciona bien con grandes conjuntos de datos en poco tiempo. En el entrenamiento C4.5, el árbol de decisión se construye de forma recursiva de arriba hacia abajo (Saeh et al., 2016).

Decision Stump: es un árbol de decisión de un nivel que clasifica las instancias según los valores de las características. Cada nodo representa una característica en una instancia a clasificar y cada rama representa un valor que el nodo toma. Las instancias se clasifican comenzando en el nodo raíz y clasificándolos en función de sus valores de características (Kotsiantis et al., 2005), (Shi et al., 2018).

Random Forest: este algoritmo utiliza métodos de arranque para crear un conjunto de árboles, uno para cada muestra de arranque. Además, las variables elegibles para usarse en la división se varían aleatoriamente para reducir la correlación de las variables. Una vez que se crea el bosque de árboles, estos votan para determinar el valor previsto de los datos de entrada (Dean, 2014).

Random Tree: construye un árbol que considera un número dado de características aleatorias en cada nodo (Witten et al., 2011).

REPTree: construye un árbol de decisión o regresión usando la reducción de la variación de la ganancia de información y lo poda usando la poda de error reducido. Optimizado para la velocidad, solo ordena los valores de los atributos numéricos una vez. Se ocupa de los valores faltantes dividiendo las instancias en partes, como lo hace C4.5. Es posible establecer la proporción mínima de variación del conjunto de entrenamiento para una división y el número de pliegues podados (Witten et al., 2011).

1.1.12. Medidas de selección de atributos

A continuación, se presentan algunas medidas de selección de atributos.

Ganancia de información. ID3 usa la ganancia de información como su atributo de selección de atributos, esta medida se basa en el trabajo pionero de Claude Shannon sobre teoría de información, la cual estudió el valor o contenido informativo de los mensajes. El nodo N representa las tuplas de la partición D. El atributo con la más alta ganancia de información se selecciona como el atributo de división para el nodo N. Este atributo minimiza la información que se necesita para clasificar las tuplas en las particiones resultantes y refleja la menor aleatoriedad o impureza en estas particiones. Este enfoque minimiza el número esperado de pruebas necesarias para clasificar una tupla dada y garantiza que se encuentre un árbol simple (pero no necesariamente el más simple) (Han et al., 2012).

Proporción de ganancia. La medida de la ganancia de información se basa en pruebas con muchos resultados. Es decir, prefiere seleccionar atributos que tengan una gran cantidad de

valores. C4.5, un sucesor de ID3, utiliza una extensión de la ganancia de información conocida como proporción de ganancia, que intenta superar este sesgo. Aplica una especie de normalización a la ganancia de información utilizando un valor de información de la división (Han et al., 2012).

Índice Gini. El índice Gini se utiliza en CART. Considera una división binaria para cada atributo. Considerando el caso donde *A* es un atributo de valor discreto que tiene *v* valores distintivos {$a_1, a_2, ..., a_v$) que ocurren en *D*. Para determinar la mejor división binaria en *A*, se examinan todos los posibles subconjuntos que se forman utilizando valores conocidos de *A* (Han et al., 2012).

1.1.13. Métodos de selección de atributos

Los métodos de selección de atributos permiten seleccionar los mejores atributos de un conjunto de datos para llevar a cabo una tarea de Minería de Datos. Algunos métodos de selección de atributos se describen a continuación.

BestFirst: realiza escalada codiciosa con retroceso; es posible especificar cuántos nodos consecutivos sin mejora deben encontrarse antes de que el sistema retroceda. Es capaz de buscar hacia adelante desde el conjunto vacío de atributos, hacia atrás desde el conjunto completo o comenzar en un punto intermedio (especificado por una lista de índices de atributos) y buscar en ambas direcciones considerando el total de las posibles adiciones y eliminaciones de atributos individuales. Los subconjuntos evaluados se guardan en caché por eficiencia; el tamaño de la caché es un parámetro.

Greedy Stepwise: Hace una búsqueda voraz hacia adelante o hacia atrás a través del espacio de subconjuntos de atributos. Es posible que comience sin/con todos los atributos o desde un punto arbitrario en el espacio. Para cuando la adición/eliminación de cualquier atributo restante da como resultado una disminución en la evaluación. También es capaz de producir una lista clasificada de atributos recorriendo el espacio de un lado al otro y registrando el orden en que se seleccionan los atributos (*Documentation - Weka Wiki*, 2020), (Weka sourceforge, 2022).

CAPÍTULO 2. ESTADO DEL ARTE

En este capítulo, se resumen algunos trabajos relacionados con la fragmentación horizontal de almacenes de datos.

En (Amina & Boukhalfa, 2013), se propuso un enfoque basado en clasificación y elección para seleccionar un esquema de fragmentación horizontal en el caso de grandes cargas de trabajo. Primero, los autores clasificaron las consultas en clases para reducir el tamaño de la carga de trabajo utilizando *k-means*. Luego, eligieron una consulta de cada clase para crear una carga de trabajo más pequeña. Después de eso, analizaron la carga de trabajo, recolectaron metadatos y seleccionaron el esquema de fragmentación utilizando un algoritmo genético (Bellatreche et al., 2008).

Por el contrario, en (Sun et al., 2014) se presentó una técnica con el objetivo de dividir (horizontalmente) los datos en bloques de tamaño fino y equilibrado de manera que las consultas maximicen el salto de bloque. Este es un proceso que se ejecutó en el momento de la carga de datos y también es posible ejecutarlo más tarde para considerar una carga de trabajo más reciente. Los filtros representativos de una carga de trabajo se extraen primero como características mediante la extracción de conjuntos de elementos frecuentes. Con base en estas características, cada tupla se representó como un vector de características. Luego, el problema de bloqueo se formuló como un problema de optimización en los vectores de características, llamado *Balanced MaxSkip Partition*, (Partición Equilibrada Max Skip) que se probó como NP-*hard*. Para encontrar una solución aproximada de manera eficiente, se adoptó el marco de agrupamiento ascendente (Ward, 1963).

Otro enfoque es (Amirat & Boukhalfa, 2014), que se centró en la selección combinada de índices de unión de mapa de bits y fragmentación horizontal. Según los autores, todos los enfoques propuestos utilizaron algoritmos para compartir atributos entre estas dos técnicas. Este trabajo mostró que los enfoques basados en el intercambio de atributos ignoran algunas soluciones alternativas interesantes. Por tanto, se propuso un nuevo enfoque basado en la minería de datos, que consiste en clasificar las consultas entre la partición horizontal y los índices de unión de mapa de bits utilizando *k-means*. Cada subconjunto de consultas se explotó mediante la técnica de optimización más adecuada. El intercambio de consultas permitió podar el espacio de búsqueda y reducir la complejidad de los problemas de selección.

En (Toumi et al., 2015) se presentó una metodología basada en estadística (índice Jaccard), minería de datos (agrupamiento jerárquico) y metaheurística (optimización del enjambre de partículas) para resolver el problema de la fragmentación horizontal en los *data warehouses*

utilizando una carga de trabajo relativamente grande. Primero, los autores calcularon la atracción entre los predicados usando el índice de Jaccard, seguido de una agrupación jerárquica del conjunto de predicados con el algoritmo de Ward (Ward, 1963). En el segundo paso, utilizaron DPSO (Optimización de enjambre de partículas discretas) para seleccionar el mejor esquema de fragmentación.

Los conceptos de Cloud SDW (*Spatial DW*) y OLAP espacial (*On-line Analytical Processing*) como servicio se presentaron en (Mateus et al., 2016). Posteriormente, esos conceptos se utilizaron para describir dos técnicas diferentes de partición de datos basadas en jerarquías para el SDW alojado en la nube: 1) Partición basada en el espacio, que fragmenta las tablas de dimensiones espaciales horizontalmente de acuerdo con una jerarquía espacial y replica las tablas restantes del SDW, y 2) De base convencional, que tiene como objetivo fragmentar tablas de dimensiones convencionales horizontalmente y replicar las tablas restantes del SDW. También se tomó en cuenta la existencia de una jerarquía entre los atributos convencionales en las tablas de dimensiones.

En contraste, el enfoque propuesto en (Ettaoufik & Ouzzif, 2017a) consistió en una técnica de fragmentación horizontal incremental (dinámica) para el DW a través de un servicio web. Esta técnica se basó en actualizar la carga de consultas agregando nuevas consultas frecuentes y eliminando consultas que ya no son frecuentes. El objetivo fue automatizar la implementación de la fragmentación incremental para optimizar una nueva carga de consultas.

Por otro lado, posteriormente en (Ettaoufik & Ouzzif, 2017b) se implementó una técnica de fragmentación horizontal incremental asistida con vistas materializadas temporales por medio de un servicio web. Está técnica consistió en administrar las vistas materializadas temporales con el fin de optimizar las nuevas consultas frecuentes antes de proceder con la implementación de la fragmentación. Los autores usan un servicio web para monitorizar y mejorar automáticamente el desempeño del DW y reducir tanto los esfuerzos manuales y los costos de implementación. El servicio web selecciona e implementa el esquema de fragmentación horizontal, y entonces monitoriza el desempeño del DW para evitar cualquier tipo de degradación.

En (Kechar & Nait-Bahloul, 2017), los autores establecieron que la fragmentación horizontal del almacén de datos se considera una de las técnicas importantes de optimización del rendimiento de las consultas de soporte de decisiones. Esta optimización se logra solo si el gran volumen de datos de la tabla de hechos está fragmentado horizontalmente. Por esa razón, los fragmentos de la tabla de hechos siempre se derivan de los fragmentos de las tablas de dimensiones. Desafortunadamente, en este tipo de fragmentación, es posible que el número de fragmentos aumente drásticamente y su mantenimiento se vuelve bastante difícil y costoso. Por lo tanto, para reducir el número de fragmentos y optimizar aún más el rendimiento de las consultas de apoyo a la decisión, los

autores propusieron fragmentar horizontalmente solo la tabla de hechos explotando conjuntamente: la selectividad de los predicados de selección, sus números de ocurrencia y sus frecuencias de acceso.

Por otro lado,(Barkhordari & Niamanesh, 2018) propusieron un método llamado Chabok, que utilizó Map-Reduce en dos fases para resolver problemas de DW con Big Data. Chabok se utilizó para almacenes de datos de esquema en estrella y fue capaz de calcular medidas distributivas. Este método también se aplicó a grandes dimensiones, que son dimensiones donde el volumen de datos es mayor que el volumen de un nodo. Chabok fragmentó horizontalmente la tabla de hechos. En caso de nodos homogéneos, se asignó el mismo número de registros a cada nodo Fact-Mapper. El método propuesto se implementó en Hadoop y las consultas TPC-DS se ejecutaron para la evaluación comparativa. El tiempo de ejecución de consultas en Chabok superó a los productos de Big Data más importantes para el almacenamiento de datos.

Como parte de su trabajo en curso sobre la partición impulsada por la carga de trabajo, en (Boissier & Kurzynski, 2018), los autores implementaron un enfoque llamado salto de datos agresivo y lo ampliaron para manejar patrones de acceso tanto analíticos como transaccionales. El objetivo principal era determinar un esquema de partición que (i) divida el conjunto de datos en un número determinado de fragmentos y (ii) esté optimizado para una poda de partición eficiente dada una carga de trabajo. Este enfoque se evaluó con la carga de trabajo y los datos de un sistema de producción de una empresa global 2000.

En contraste, el método que se presentó en (Barr et al., 2018) utilizó programación lineal para resolver el problema NP-*hard* de determinar un esquema de fragmentación horizontal en un DW relacional. Además de diseñar y resolver el problema de selección de la técnica de fragmentación horizontal, el problema se consideró en dos objetivos simultáneos, denominados: el número de Entradas / Salidas necesarias para ejecutar la carga de trabajo global, y el número de fragmentos generados para identificar las mejores soluciones en comparación con el concepto de dominancia de Pareto.

Además, en (Nam et al., 2018), se estableció que a medida que aumenta la cantidad de datos a procesar, un método de partición horizontal de base de datos eficiente se vuelve más importante para el procesamiento de consultas OLAP en plataformas de bases de datos paralelas. Los métodos existentes tienen algunas desventajas importantes, como una gran cantidad de redundancia de datos y que no admiten el procesamiento de combinaciones sin reestructuración (mezcla) en muchos casos a pesar de su alta redundancia de datos. Los autores propusieron un método de partición de bases de datos basado en grafos llamado GPT que mejora el rendimiento de las consultas con menos redundancia de datos.

Una técnica basada en la extracción de conjuntos de elementos frecuentes se propuso en (Ramdane et al., 2018) para fragmentar, agrupar y ordenar las tablas (PBST) de un almacén

de Big Data con los atributos de predicados más frecuentes en las consultas. Los autores tomaron en cuenta la densidad de los atributos de las tablas, el sesgo de los datos y las características físicas de los nodos del clúster. Este enfoque utilizó una técnica de partición *hash* y consiste en construir fragmentos horizontales de las tablas de hechos y dimensiones de un gran almacén de datos relacional, utilizando técnicas PBST basadas en las consultas de la carga de trabajo.

Para mejorar su trabajo anterior, en (Nam et al., 2019), los autores integraron su método GPT en un sistema de procesamiento de consultas en paralelo, Spark SQL, en todas las capas y módulos relevantes, incluido el generador de planes de consultas y el operador de búsqueda. A través de extensos experimentos utilizando tres *benchmarks*: TPC-DS, IMDB y BioWarehouse, Nam et al. demostraron que GPT supera significativamente al método de vanguardia en términos de sobrecarga de almacenamiento y rendimiento de consultas.

En (Letrache et al., 2019), los autores afirmaron que la mayoría de las estrategias para la fragmentación de DW se centran en la fragmentación de DW relacional e ignoran los cubos OLAP, aunque son los primeros afectados por consultas multidimensionales de los usuarios. Para abordar este problema, se propuso una estrategia de fragmentación dinámica para cubos OLAP utilizando minería de reglas de asociación. Letrache et al. consideraron que incluso aunque la fragmentación es compatible con la mayoría de los proveedores de OLAP, no es posible realizar la definición de una estrategia de fragmentación eficiente mediante las herramientas disponibles.

Por otro lado, en (Kechar & Nait-Bahloul, 2019), los autores desarrollaron una versión mejorada de su trabajo anterior (Kechar & Nait-Bahloul, 2017); presentaron un enfoque de partición de datos horizontal adaptado a un gran DW, interrogado a través de un gran número de consultas. La idea era fragmentar horizontalmente solo la gran tabla de hechos con base en predicados de partición, elegidos del conjunto de predicados de selección utilizados por las consultas analíticas.

Por el contrario, los autores de (Ramdane et al., 2019a) asumieron que las técnicas de partición horizontal se han utilizado para muchos propósitos en el procesamiento de Big Data, como el equilibrio de carga, omitir cargas de datos innecesarias y guiar el diseño físico de un almacén de datos. Por lo tanto, propusieron una nueva estrategia de colocación de datos en el entorno de Apache Hadoop denominada "Ubicación de almacén de datos inteligente (SDWP)", que permitió realizar la operación de reunión de estrellas en una sola etapa de Spark. Se investigó el problema de la partición y el equilibrio de carga en un grupo de nodos homogéneos. Los experimentos que utilizaron el *benchmark* TPC-DS mostraron que el método propuesto mejora el rendimiento de las consultas OLAP en términos de tiempo de ejecución.

Asimismo, (Ramdane et al., 2019b) combinaron un modelo basado en datos y uno basado en cargas de trabajo para crear un nuevo esquema para almacenes de Big Data distribuidos sobre Hadoop, llamado "SkipSJoin". Primero, SkipSJoin construyó fragmentos horizontales (cubos) de las tablas de hechos y dimensiones del DW utilizando un método de partición *hash* y distribuyó estos cubos de manera uniforme entre los nodos del clúster. Luego, permitió omitir el acceso a algunos bloques de datos innecesarios, mediante la partición *hash* de algunas tablas del DW con atributos frecuentes de los filtros. Con experimentos que utilizaron el *benchmark* TPC-DS, Ramdane et al. demostraron que su propuesta superó a algunos enfoques en términos de tiempo de ejecución de consultas.

Por el contrario, (Hilprecht et al., 2019a) introdujeron que los productos de analítica de datos comerciales como Microsoft Azure SQL Data Warehouse o Amazon Redshift brindan soluciones de bases de datos escalables para cargas de trabajo de estilo OLAP en la nube. Mientras que el aprovisionamiento de un clúster de base de datos es, en general, totalmente automatizado por los proveedores de la nube, los clientes todavía tienen que tomar decisiones de diseño importantes que tradicionalmente tomaba el administrador de la base de datos, como seleccionar los esquemas de fragmentación. Por lo tanto, los autores propusieron un asesor de particiones aprendido para cargas de trabajo analíticas de estilo OLAP basadas en *Deep Reinforcement Learning* (DRL). La idea principal era que un agente de DRL aprende sus decisiones basándose en la experiencia al monitorizar las recompensas para diferentes cargas de trabajo y esquemas de partición.

De manera similar, (Hilprecht et al., 2019b) evaluó el asesor de particiones aprendido en tres esquemas de base de datos y cargas de trabajo diferentes (SSB, TPC-DS y TPC-CH) que varían en complejidad que van desde un esquema en estrella simple hasta un esquema normalizado complejo. En los experimentos, se utilizó PostgreSQL-XL como sistema de bases de datos distribuido. Asimismo, en su siguiente trabajo (Hilprecht et al., 2020), los autores demostraron que su enfoque no solo es capaz de encontrar particiones que superen los enfoques existentes para el diseño de particiones automatizadas, sino que también es capaz de ajustarse a diferentes cargas de trabajo y nuevas consultas.

Finalmente, (Parchas et al., 2020) se centraron en la forma “Dist-Key” de fragmentación horizontal de la mayoría de los sistemas de almacenamiento de datos comerciales. De esta manera se dividen las tuplas de una relación de acuerdo con los valores de un atributo específico conocido como la clave de distribución. Por lo tanto, los autores tenían el propósito de reducir el costo de red de una carga de trabajo dada seleccionando el mejor atributo para distribuir cada tabla del almacén de datos. Propusieron BaW (*Best of All Worlds*), un enfoque híbrido que combinó algoritmos heurísticos y exactos para elegir la clave de distribución óptima para un subconjunto de las relaciones.

Después de un análisis exhaustivo del estado del arte, los métodos de fragmentación horizontal de los DW se clasificaron según sus principales características, como la

validación del método propuesto y el enfoque del método de fragmentación. La Tabla 2.1 muestra las principales características de los métodos descritos anteriormente.

Como se observa en la Tabla 2.1, se propusieron varios enfoques basados en Minería de Datos. Sin embargo, la mayoría aplican las tareas de agrupamiento (Amina & Boukhalfa, 2013), (Sun et al., 2014), (Amirat & Boukhalfa, 2014), (Toumi et al., 2015) o asociación (Sun et al., 2014), (Ramdane et al., 2018), (Letrache et al., 2019), específicamente, agrupamiento basado en partición (k-means) (Amina & Boukhalfa, 2013), (Amirat & Boukhalfa, 2014) o agrupamiento jerárquico aglomerativo (método Ward) (Sun et al., 2014), (Toumi et al., 2015) y técnicas de minería de conjuntos de elementos (*itemset*) frecuentes (algoritmo Apriori) (Letrache et al., 2019). Las referencias (Hilprecht et al., 2019a) y (Hilprecht et al., 2020) usaron redes neuronales implementadas en Keras para el aprendizaje del asesor de particiones. En este trabajo, se desarrolló un método de fragmentación horizontal derivada que utiliza árboles de decisión para dividir la tabla de dimensión previamente seleccionada considerando la selectividad de los predicados y la frecuencia de las consultas OLAP, y divide la tabla de hechos de acuerdo con el esquema de fragmentación horizontal de la tabla de dimensión obtenida por el árbol de decisión.

Más tarde, se realizó un análisis comparativo (Tabla 3.1) de 49 métodos de fragmentación horizontal considerando siete criterios: 1) Tipo de esquema utilizado: estrella, copo de nieve, constelación de hechos, entre otros; 2) Completitud: Si el artículo incluye todo lo necesario para implementar el método; 3) Validación: Forma de evaluación; 4) Facilidad de implementación; 5) Modelo de costo, 6) Tipo y gestor de base de datos usados: relacional, multidimensional, orientado a objetos, XML, objeto-relacional, entre otros, y 7) Pasos del método. Posteriormente, se analizaron los modelos de costo. Los hallazgos de este análisis se presentan en el siguiente capítulo.

Tabla 2.1. Tabla comparativa de trabajos sobre fragmentación horizontal

Trabajo	Clasificación	Validación
(Amina & Boukhalfa, 2013)	Basado en Minería de Datos (Agrupamiento)	*Benchmark* APB-1
(Sun et al., 2014)	Basado en Minería de Datos (Agrupamiento y asociación) Basado en Costo	*Benchmark* TPC-H y carga de trabajo de una empresa de transmisión de video, llamada Conviva.
(Amirat & Boukhalfa, 2014)	Basado en Minería de Datos (Agrupamiento)	*Benchmark* APB-1 *Release* II
(Toumi et al., 2015)	Basado en Minería de Datos (Agrupamiento) y Basado en Meta-heurística.	*Benchmark* APB-1
(Costa et al., 2016)	Basado en Jerarquía	*Benchmark* Data Warehouse Espacial (Spadawan)
(Ettaoufik & Ouzzif, 2017a)	Basado en Costo	*Benchmark* APB-1

Trabajo	Clasificación	Validación
(Kechar & Nait-Bahloul, 2017)	Basado en Costo	*Benchmark* APB-1
(Barkhordari & Niamanesh, 2018)	Basado en Map-Reduce	*Benchmark* TPC-DS
(Boissier & Kurzynski, 2018)	Basado en Costo	*Benchmarks* TPC-C, TPC-CH (CH-benCHmark), y carga de trabajo de un sistema SAP ERP de una empresa Global 2000.
(Barr et al., 2018)	Basado en Meta-heurística	*Benchmark* APB-1
(Nam et al., 2018)	Basado en Costo Basado en Grafo	*Benchmark* TPC-DS, The Internet Movie DataBase (IMDB) y BioWarehouse
(Ramdane et al., 2018)	Basado en Minería de Datos (Asociación)	*Benchmark* TPC-DS
(Nam et al., 2019)	Basado en Costo Basado en Grafo	*Benchmarks* TPC-DS, The Internet Movie Data Base (IMDB) y Bio Warehouse
(Letrache et al., 2019)	Basado en Minería de Datos (Asociación)	*Benchmark* TPC-DS
(Kechar & Nait-Bahloul, 2019)	Basado en Costo	SSB Star Schema Benchmark
(Ramdane et al., 2019a)	Basado en Partición *Hash*	*Benchmark* TPC-DS
(Ramdane et al., 2019b)	Basado en partición *Hash*	*Benchmark* TPC-DS
(Hilprecht et al., 2019a)	Basado en DRL (redes neuronales)	*Benchmarks* SSB, TPC-DS, y TPC-CH
(Hilprecht et al., 2019b)	Basado en DRL	*Benchmarks* SSB, TPC-DS, y TPC-CH
(Hilprecht et al., 2020)	Basado en DRL (redes neuronales)	*Benchmarks* SSB, TPC-DS, y TPC-CH
(Parchas et al., 2020)	Basado en partición *Hash*	Gráfico de unión Real1, Real2 extraído al azar de usuarios reales de Redshift con varios tamaños y densidades y *Benchmark* TPC-DS

CAPÍTULO 3. APLICACIÓN DE LA METODOLOGÍA

En este capítulo se describen los pasos seguidos para el desarrollo del proyecto de tesis. La metodología seguida para el desarrollo de este proyecto de tesis se visualiza en la Figura 3.1 de este documento y muestra las diferentes etapas que constituyeron la realización de todo el proyecto.

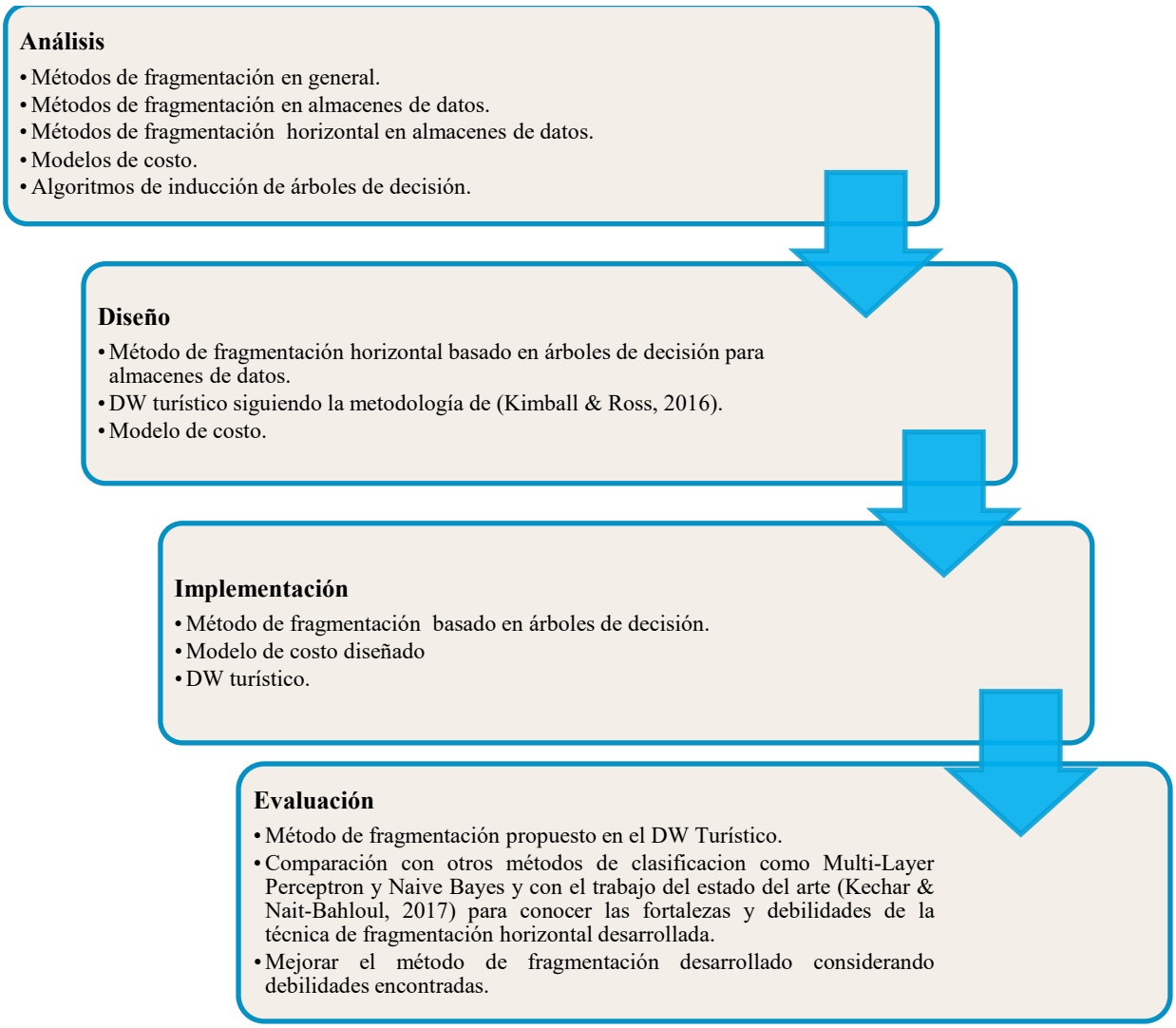


Figura 3.1. Metodología del proyecto de tesis

3.1. Análisis

3.1.1. Análisis del estado del arte

Se llevó a cabo la revisión del estado del arte para conocer las ventajas y desventajas de los métodos de fragmentación para almacenes de datos, además se analizaron los modelos de costo utilizados por los métodos, así como sus características a fin de encontrar los costos más relevantes considerados por los autores al obtener su esquema de fragmentación en el DW.

La Figura 3.2 muestra la metodología que se siguió para analizar los métodos de fragmentación y los modelos de costo. Primero, se consultaron las principales bibliotecas digitales de Ciencias de la Computación como son: 1) ACM Digital Library; 2) Springer Link; 3) Science Direct; 4) IEEE Xplore, y 5) Otras. Las palabras clave utilizadas en la búsqueda fueron: *Fragmentation*, *Data Warehouse*, *Horizontal Fragmentation*, *Partitioning* y *Horizontal Partitioning*. Se encontraron un total de 93 artículos sobre los métodos de fragmentación para DW, posteriormente se clasificaron según el tipo de fragmentación que se realizaba encontrando que, de los artículos obtenidos en la búsqueda, la mayoría se enfocan en fragmentación horizontal, como se aprecia en la Figura 3.3. Los trabajos también se clasificaron por editorial y año de publicación, esto se muestra en las Figuras 3.4 y 3.5.

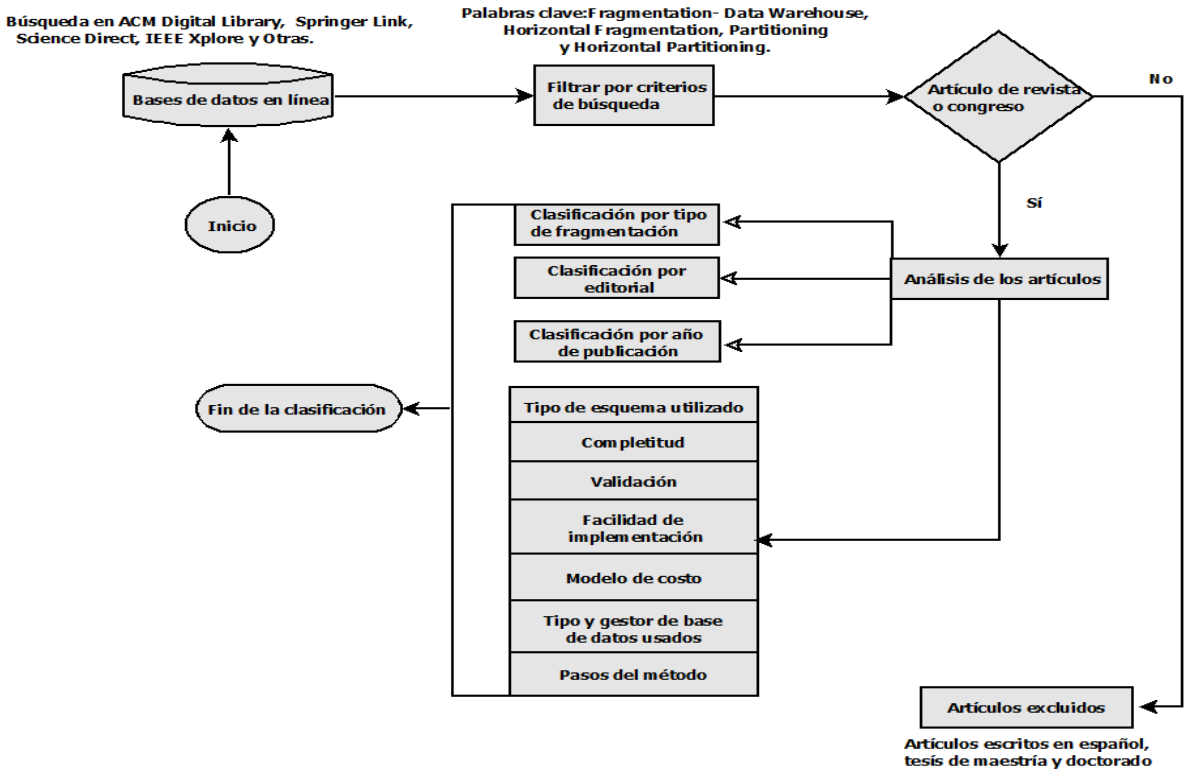


Figura 3.2. Metodología seguida para la investigación sobre los métodos de fragmentación

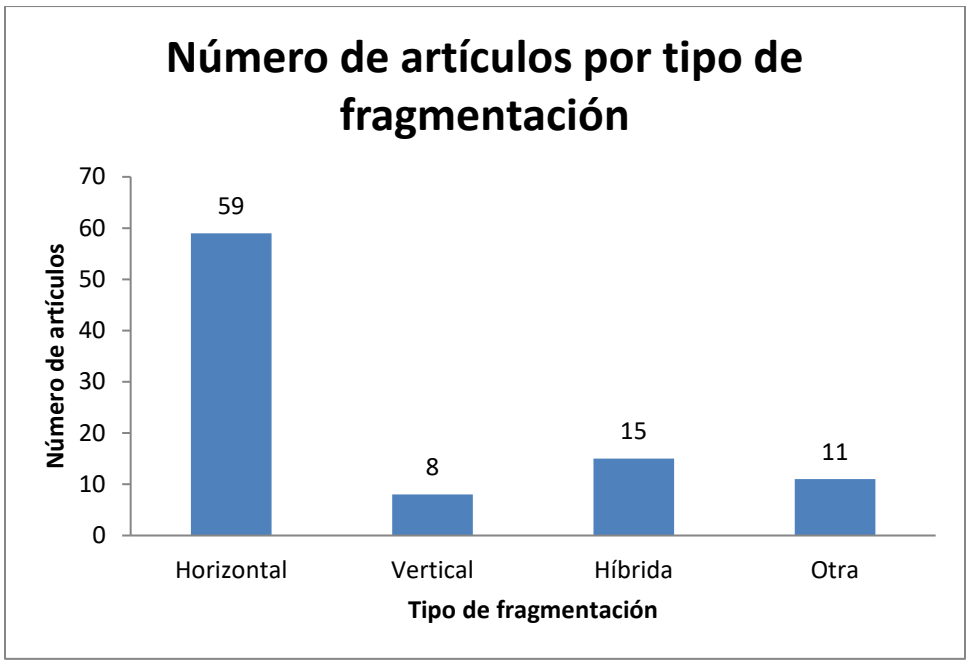


Figura 3.3. Clasificación de artículos encontrados según el tipo de fragmentación

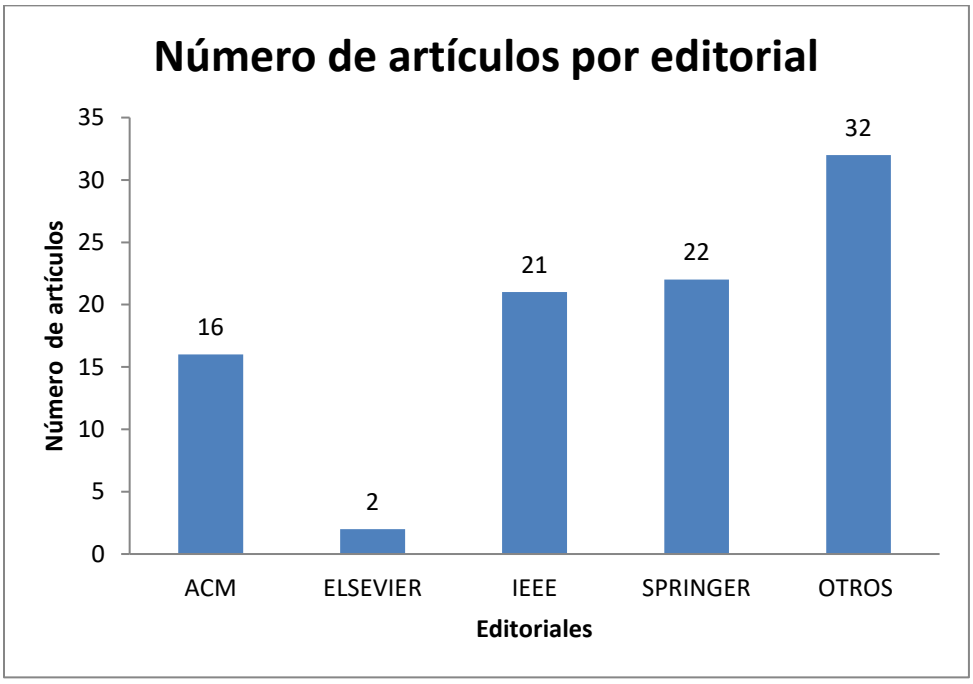


Figura 3.4. Número de artículos encontrados por editorial

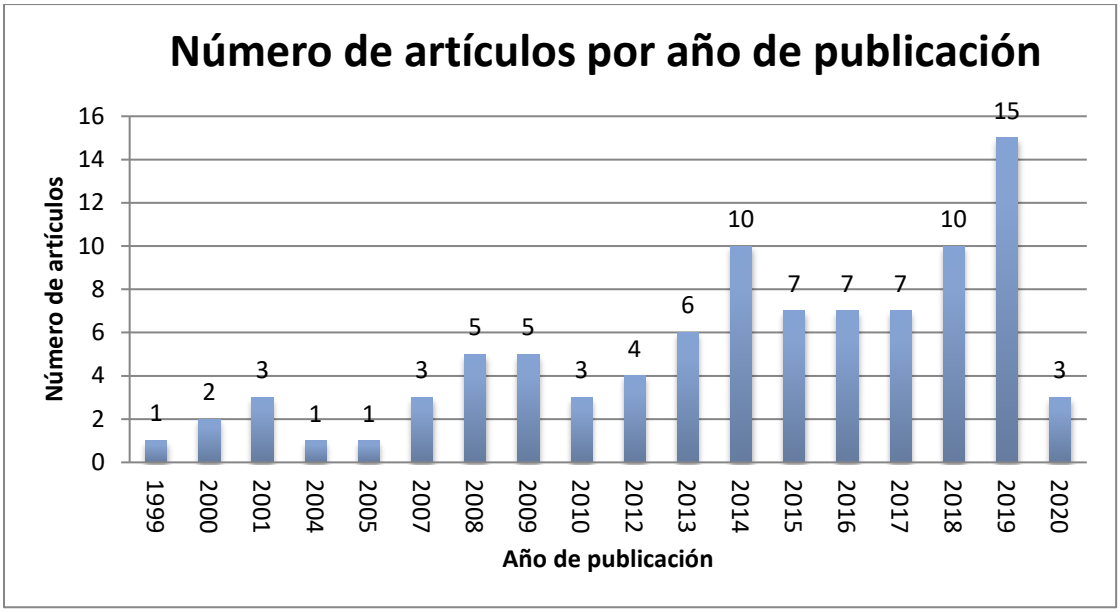


Figura 3.5. Número de artículos por año de publicación.

3.1.2. Análisis Comparativo de trabajos de Fragmentación Horizontal en DW

Después, se realizó un análisis comparativo (Tabla 3.1) de 49 métodos de fragmentación horizontal considerando siete criterios: 1) Tipo de esquema utilizado: estrella, copo de nieve, constelación de hechos, entre otros; 2) Completitud: Si el artículo incluye todo lo necesario para implementar el método; 3) Validación: Forma de evaluación; 4) Facilidad de implementación; 5) Modelo de costo, 6) Tipo y gestor de base de datos usados: relacional, multidimensional, orientado a objetos, XML, objeto-relacional, entre otros, y 7) Pasos del método. Posteriormente, se analizaron los modelos de costo.

Tabla 3.1 Comparación de métodos de fragmentación horizontal para DW.

Trabajo	Esquema	Completo	Validación	Fácil	Costo	Base de datos
(Noaman & Barker, 1999)	Estrella	Sí	Con respecto a las tres reglas de exactitud de la fragmentación: completitud, reconstrucción y disyunción.	Sí	No	Relacional
(Bellatreche et al., 2000)	Estrella	Sí	Conjunto de datos de Informix	Sí	Sí	Relacional
(Gopalkrishnan et al., 2000)	Esquema *Object Relational Data Warehouse*	Sí	Esquema ORDW	No	Sí	Objeto-relacional

Trabajo	Esquema	Completo	Validación	Fácil	Costo	Base de datos
	(ORDW)					
(Gopalkrishnan et al., 2001)	Copo de nieve	Sí	Esquema ORDW	No	Sí	Objeto-relacional
(Ezeife, 2001)	Estrella	Sí	Se realizó un experimento utilizando un DW bancario.	Sí	No	Relacional (Oracle 7.1)
(Furtado, 2004)	Estrella	Sí	*Benchmark* TPC-H	No	Sí	Relacional (Oracle 9i)
(Bellatreche & Boukhalfa, 2005)	Estrella	Sí	*Benchmark* APB-1 Release II	Sí	Sí	Relacional
(Bellatreche et al., 2006)	Estrella	Sí	*Benchmark* APB-1 *Release* II	Sí	Sí	Relacional
(de Aguiar et al., 2007)	Estrella	Sí	*Benchmark* TPC-H	Sí	No	Relacional (Oracle™)
(Benkrid et al., 2008)	Estrella	Sí	*Benchmark* APB-1	No	Sí	Relacional
(Bellatreche et al., 2008)	Estrella	No	*Benchmark* APB-1	No	Sí	Relacional (Oracle 10g)
(Mahboubi & Darmont, 2008)	XML DW	Sí	XWeB (XML *Data Warehouse Benchmark*)	Sí	No	XML (X-Hive XML)
(Bellatreche & Woameno, 2009)	Estrella	Sí	*Benchmark* APB-1	Sí	Sí	Relacional (Oracle 11g)
(Bellatreche & Benkrid, 2009)	Estrella	Sí	*Benchmark* APB-1	No	Sí	Relacional
(Mahboubi & Darmont, 2009)	XML DW	Sí	XWeB (XML Data Warehouse Benchmark)	Sí	No	XML (X-Hive XML)
(Cuzzocrea et al., 2009)	XML DW	Sí	XWeB (XML Data Warehouse Benchmark)	Sí	No	XML (X-Hive XML)
(Karima et al., 2010)	Estrella	Sí	*Benchmark* APB-1 *Release* II	Sí	No	Relacional (Oracle 11g)
(Barr & Bellatreche, 2010)	Estrella	Sí	*Benchmark* APB-1	SÍ	Sí	Relacional (Oracle 10g)
(Dimovski	Estrella	No	Experimentos	No	Sí	Relacional

Trabajo	Esquema	Completo	Validación	Fácil	Costo	Base de datos
et al., 2010)						
(Brki, 2012)	Estrella	No	Probado en un proyecto de la vida real en instituciones educativas de Croacia.	Sí	No	Relacional
(Barr, 2012)	Estrella	Sí	*Benchmark* APB-1	Sí	No	Relacional (Oracle 10g)
(Liu et al., 2013)	Estrella	Sí	*Benchmark* TPC-H	Sí	No	Objeto-Relacional
(Rafid, 2013)	Estrella	Sí	Experimentos con 10 consultas complejas.	Sí	No	Relacional
(Bellatreche et al., 2013)	Estrella	No	Experimentos	No	No	Relacional (Oracle DBMS)
(Barr, 2013)	Estrella	Sí	*Benchmark* APB-1	No	No	Relacional
(Amina & Boukhalfa, 2013)	Estrella	No	Benchmark APB-1	Sí	Sí	Relacional (Oracle 11g)
(Sun et al., 2014)	Estrella	Sí	*Benchmark* TPC-H y carga de trabajo de una compañía de transmisión de video, llamada Conviva.	No	Sí	Relacional (Shark)
(Bouchakri, Bellatreche, Faget, et al., 2014)	Estrella	Sí	*Benchmark* APB-1	No	Sí	Relacional (Oracle)
(Bouchakri, Bellatreche, & Faget, 2014)	Estrella	Sí	*Benchmark* APB-1	No	Sí	Relacional (Oracle 11g)
(Sarka et al., 2014) (Hanane & Kamel, 2014)	Estrella	Sí	*Benchmark* APB-1 *Release* II	No	Sí	Relacional (Oracle 11g)
(Toumi et al., 2015)	Estrella	Sí	*Benchmark* APB-1	Sí	Sí	Relacional (Oracle 11g)
(Mateus et al., 2016)	DW espacial (SDW)	Sí	*Spatial Data Warehouse*	No	No	Espacial en la nube

Trabajo	Esquema	Completo	Validación	Fácil	Costo	Base de datos
			Benchmark (Spadawan)			
(Ettaoufik & Ouzzif, 2017a)	Estrella	Sí	*Benchmark* APB-1	No	Sí	Relacional (Oracle 11g)
(Ettaoufik & Ouzzif, 2017b)	Estrella	Sí	*Bechmark* APB-1	No	Sí	Relacional (Oracle 11g)
(Kechar & Nait-Bahloul, 2017)	Estrella	Sí	*Benchmark* APB-1	Sí	Sí	Relacional (Oracle 10g)
(Barkhordari & Niamanesh, 2018)	Estrella	Sí	*Benchmark* TPC-DS	No	No	Objeto-relacional (PostgreSQL)
(Boissier & Kurzynski, 2018)	Estrella	Sí	*Benchmarks TPC-C, TPC-CH (*CH-benCHmark*)*, datos y carga de trabajo de un sistema SAP ERP de una compañía Global 2000.	No	Sí	Base de datos columnar en memoria
(Barr et al., 2018)	Estrella	Sí	*Benchmark* APB-1	Sí	Sí	Relacional
(Nam et al., 2018) Estático	Constelación	Sí	*Benchmark* TPC-DS, The Internet Movie DataBase (IMDB) y BioWarehouse	No	Sí	Ninguna (Hadoop Distributed File System)
(Ramdane et al., 2018)	Estrella	Sí	*Benchmark* TPC-DS	No	No	Relational (Spark-SQL, Hive y MySQL)
(Kechar & Nait-Bahloul, 2019)	Estrella	Sí	SSB	Sí	Sí	Relacional (Oracle 12c)
(Nam et al., 2019)	Constelación	Sí	*Benchmark* TPC-DS, The Internet Movie DataBase (IMDB) y BioWarehouse	No	Sí	Paralela (Apache Spark SQL)
(Letrache et al., 2019)	Estrella	Sí	*Benchmark* TPC-DS	No	Sí	Relacional (Microsoft SQL Server)
(Ramdane	Star schema	Si	*TPC-DS*	No	No	Relacional

Trabajo	Esquema	Completo	Validación	Fácil	Costo	Base de datos
et al., 2019a)			*benchmark*			
(Ramdane et al., 2019b)	Star esquema	No	*TPC-DS Benchmark*	No	No	Relacional
(Hilprecht et al., 2019a)	Star schema	No	*SSB, TPC-DS y TPC-CH Benchmarks*	No	Si	Objeto-Relacional PostgreSQL-XL System XL
(Hilprecht et al., 2019b)	Diferentes esquemas de bases de datos	Si	*SSB, TPC-DS y TPC-CH Benchmarks*	No	Si	Base de datos distribuida
(Hilprecht et al., 2020)	Diferentes esquemas de bases de datos	No	*SSB, TPC-DS y TPC-CH Benchmarks*	No	Si	Objeto-Relacional PostgreSQL-XL System XL
(Parchas et al., 2020)	Star schema	Si	*Real1, Real2 Gráficos de unión extraído al azar de usuarios reales de Redshift con varios tamaños y densidades y TPC-DS Benchmark.*	No	Si	Relacional Amazon redshift

Cada trabajo se analizó considerando siete criterios:

1. **Tipo de esquema multidimensional utilizado:** Para cada método, se identificó el esquema utilizado para modelar el DW: estrella, copo de nieve, constelación, por mencionar los más comunes.

2. **Completitud:** Se refiere a que el artículo contenga todo lo necesario para la implementación del método.

3. **Validación:** En esta columna se indica si se usaron datos sintéticos (*benchmark*) o datos reales para validar el método de fragmentación.

4. **Facilidad de implementación:** Este criterio considera si la implementación del método es fácil de replicar, es decir, si en el artículo se explican claramente cada uno de los pasos del método.

5. **Modelo de costos:** Usar un modelo de costos provee una estimación de costo de acuerdo con la carga de consultas comúnmente usadas. Este proporciona una estimación del consumo de recursos por una estructura o una técnica. Estos recursos incluyen espacio de almacenamiento, tiempo de respuesta o el número de entrada/salida entre la memoria y el disco (Barr & Bellatreche, 2010).

6. **Modelo de datos y gestor de bases de datos usado:** De acuerdo con (Noaman & Barker, 1999), hay dos técnicas para modelar el DW*:* el modelo de datos multidimensional y el modelo de datos relacional. Estas dos técnicas de modelado proveen una vista multidimensional de datos para soportar y facilitar las operaciones OLAP (*On Line Analytics Processing*, Procesamiento Analítico en Línea).

7. **Pasos del método:** Contiene cada uno de los pasos del método de fragmentación.

Después de haber analizado los 49 métodos de fragmentación horizontal en DW se concluyó lo siguiente:

Esquema multidimensional: Como se aprecia en la Figura 3.6, el esquema de estrella fue el modelo multidimensional más utilizado por los métodos; tres utilizaron el esquema XML DW, el cual está compuesto por documentos XML que representan tanto hechos como dimensiones y que permite modelar esquemas de constelación de hechos sin la necesidad de duplicar información de las dimensiones, por lo tanto, logra dimensiones compartidas (Mahboubi & Darmont, 2008), (Mahboubi & Darmont, 2009) y (Cuzzocrea et al., 2009); dos usaron el esquema de constelación (Nam et al., 2018), (Nam et al., 2019); tres emplearon esquemas ORDW (Gopalkrishnan, Li, & Karlapalem, 2000), copo de nieve (Gopalkrishnan, Li, & Karlapalem, 2001), y DW espacial (Mateus et al., 2016), respectivamente.

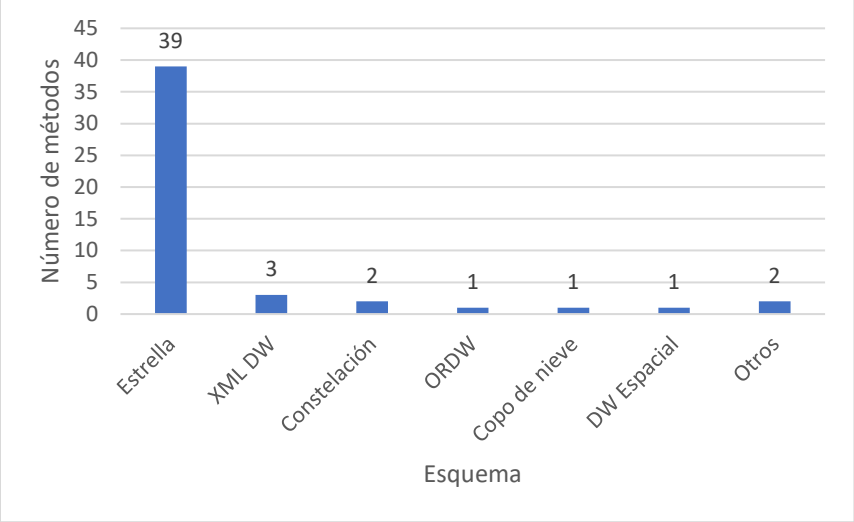


Figura 3.6. Esquemas utilizados por los métodos.

Completitud: La Figura 3.7 muestra que la mayoría de los artículos analizados presentaron la información necesaria para la implementación del método, solo cinco no contaron con completitud, por ejemplo, en (Bellatreche, Boukhalfa, & Richard, 2008), de acuerdo con los autores, no se incluyó el modelo de costos por falta de espacio y aunque se presentó un enlace a un reporte técnico, este ya no estaba disponible; mientras que en (Dimovski et al., 2010) y en (Bellatreche, et al., 2013) no se utilizó un *benchmark* o un DW real para evaluar el método; en (Brki, 2012) no se explicó cómo determinar los predicados que se utilizaron en la fragmentación; en (Amina & Boukhalfa, 2013) utilizaron *k-means* para clasificar consultas, pero no brindaron alguna pauta para la selección del valor de *k*.

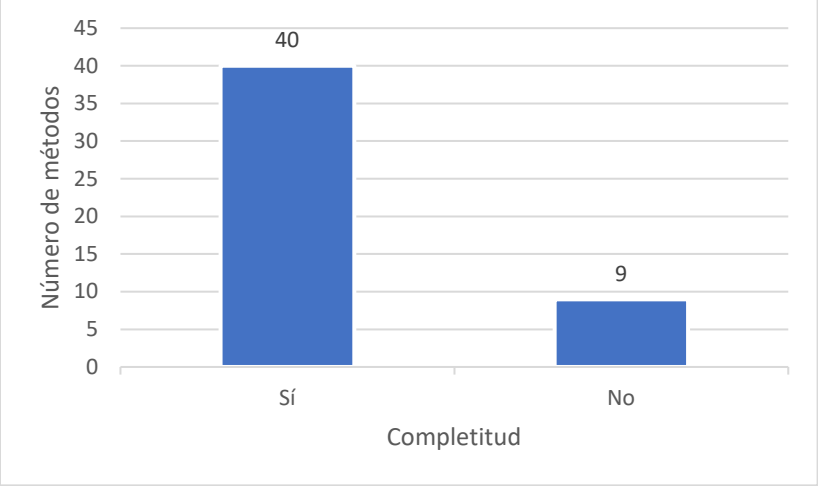


Figura 3.7. Completitud de los métodos.

Validación: 43 de los 49 artículos analizados utilizaron *benchmarks* para la evaluación del método de fragmentación. La Figura 3.8 muestra que los *benchmarks* más usados fueron APB-1, TPC-DS y TPC-H.

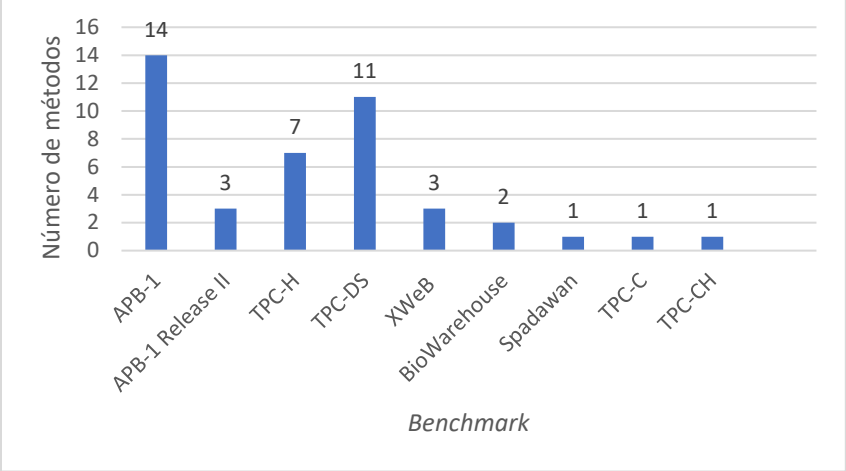


Figura 3.8. *Benchmarks* empleados para validar los métodos.

Facilidad de implementación: Más de la mitad de los métodos se clasificaron como difíciles de implementar como se observa en la Figura 3.9, debido a que algunos se enfocaron en la fragmentación de bases de datos complejas, como orientadas a objetos (Gopalkrishnan et al., 2000), (Gopalkrishnan et al., 2001) o espaciales (Mateus et al., 2016), mientras que otros realizaron fragmentación incremental (Bellatreche et al., 2013), (Bouchakri, Bellatreche, & Faget, 2014), (Bouchakri, Bellatreche, Faget, et al., 2014), (Ettaoufik & Ouzzif, 2017a), (Ettaoufik & Ouzzif, 2017b), (Letrache et al., 2019), la cual se adapta a los cambios en los patrones de acceso al DW; otros se enfocaron en la selección combinada de fragmentación horizontal e índices (Hanane & Kamel, 2014), tomando en cuenta predicados compuestos (Dimovski et al., 2010) o consideraron el problema de diseñar DW paralelos (Furtado, 2004), (Benkrid et al., 2008), (Bellatreche & Benkrid, 2009), (Barkhordari & Niamanesh, 2018), (Nam et al., 2018).

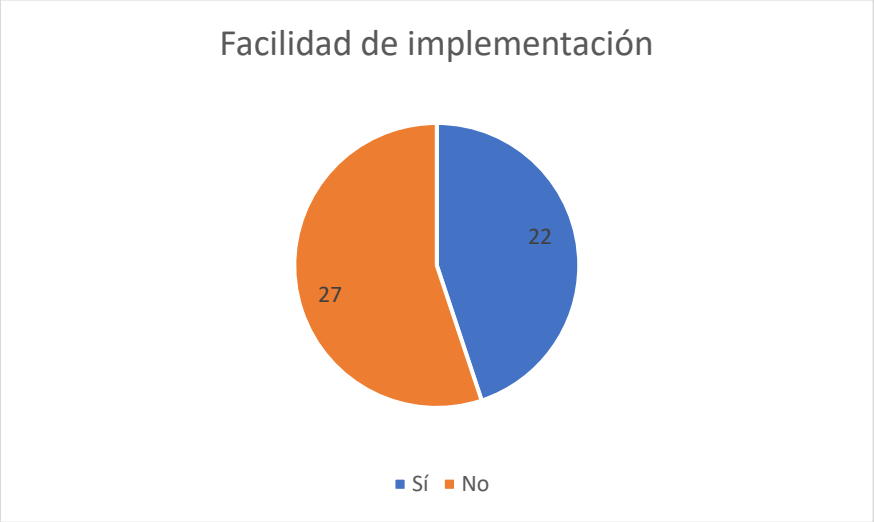


Figura 3.9. Facilidad de implementación de los métodos.

Modelo de costos: Como se observa en la Figura 3.10, 30 métodos incluyen un modelo de costos, los cuales se comparan en la siguiente sección.

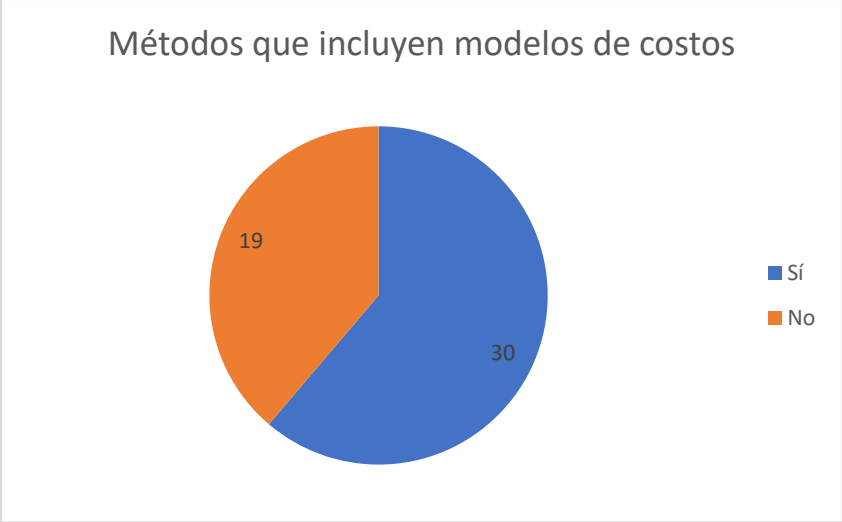


Figura 3.10. Tipos de SGBDs utilizados por los métodos

Modelo de datos y gestor de bases de datos usado: La Figura 3.11 muestra que el modelo de datos más utilizado para la implementación de los métodos fue el relacional. El SGBD más usado fue Oracle 11g como se aprecia en la Figura 3.12, debido a que Oracle 11g ofrece una gran evolución de la fragmentación horizontal, ya que soporta varios métodos de fragmentación: los métodos de fragmentación compuestos incluyen todas las posibles combinaciones de métodos básicos (rango, *hash* y lista) y soporta fragmentación de columnas, donde una tabla se descompone usando un atributo virtual definido por una expresión, usando una o más columnas existentes de una tabla, y almacenando esta expresión como metadatos, además fue el primer gestor capaz de realizar fragmentación referencial para dividir una tabla usando una relación padre-hijo, la cual es similar a la fragmentación derivada (Bellatreche et al., 2008), (Bouchakri, Bellatreche, & Faget, 2014).

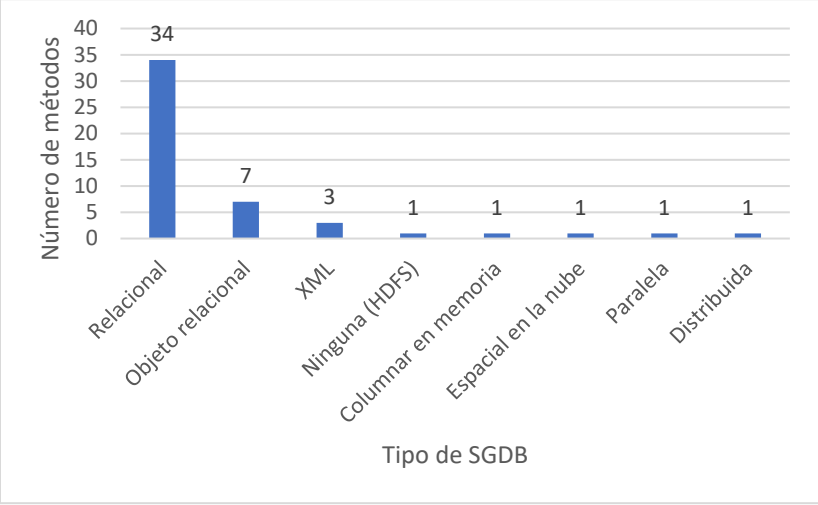


Figura 3.11.Modelo de Datos usados para la implementación de los métodos.

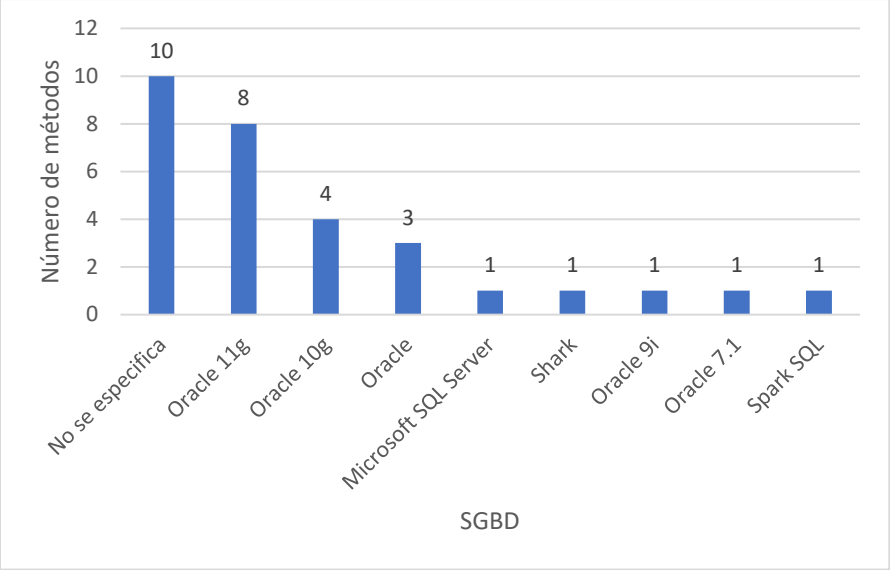


Figura 3.12. SGBDs usados para la implementación de los métodos.

3.1.3. Análisis de modelos de costo

En cuanto a los modelos de costo se revisaron un total de 24, para los cuales se tomaron en cuenta los criterios como costos considerados y si estos correspondían a procesamiento local y comunicación.

Como se observa en la Tabla 3.2, casi todos los modelos de costo solo toman en cuenta el costo de procesamiento local, solo (Furtado, 2004) considera el costo de comunicación.

Tabla 3.2 Análisis de los modelos de costo

Modelo de costo	Costos considerados	Procesamiento local	Comunicación
(Bellatreche et al., 2000)	• Carga de tablas de dimensiones • Selección de tablas de dimensiones • Carga de tuplas útiles de las tablas de dimensiones. • Ejecución de operaciones de selección. • Acceso de tuplas de la tabla de hechos. • Carga de tuplas útiles de la tabla de hechos.	Sí	No
(Gopalkrishnan et al., 2000)	• Almacenamiento • Recuperación • Mantenimiento	Sí	No
(Gopalkrishnan et al., 2001)	• Almacenamiento • Recuperación • Mantenimiento	Sí	No
(Furtado, 2004)	• Fragmentación	Sí	Sí

Modelo de costo	Costos considerados	Procesamiento local	Comunicación
	• Refragmentación • Comunicación • Procesamiento local • Fusión		
(Bellatreche & Boukhalfa, 2005)	• Número de operaciones de E/S	Sí	No
(Bellatreche, Boukhalfa, & Abdalla, 2006)	• Número de operaciones de E/S (accesos a páginas)	Sí	No
(Benkrid et al., 2008)	• Número de operaciones de E/S	Sí	No
(Bellatreche et al., 2008)	• Número de operaciones de E/S	Sí	No
(Bellatreche & Woameno, 2009)	• Número de operaciones de E/S (tamaño de resultados intermedios de reuniones y el tamaño del *buffer*)	Sí	No
(Bellatreche & Benkrid, 2009)	• Número de operaciones de E/S	Sí	No
(Barr & Bellatreche, 2010)	• Número de operaciones de E/S	Sí	No
(Dimovski et al., 2010)	• Número total de filas del fragmento	Sí	No
(Amina & Boukhalfa, 2013)	• Número de operaciones de E/S	Sí	No
(Sun et al., 2014)	• Número de tuplas que es posible saltarse si se ejecutan todas las consultas en la carga de trabajo.	Sí	No
(Bouchakri et al., 2014)	• Número de operaciones de E/S (en términos de páginas)	Sí	No
(Bouchakri, Bellatreche, & Faget, 2014)	• Número de operaciones de E/S (en términos de páginas)	Sí	No
(Hanane & Kamel, 2014)	• Cardinalidad de los atributos de selección (CAR) • Selectividad de las consultas (SEL) • Número de operaciones de E/S	Sí	No
(Toumi et al., 2015)	• Número de accesos a páginas de disco	Sí	No
(Ettaoufik & Ouzzif, 2017a)	• Selección del nuevo esquema • Implementación del esquema	Sí	No
(Ettaoufik & Ouzzif, 2017b)	• Selección del nuevo esquema • Implementación del esquema	Sí	No

Modelo de costo	Costos considerados	Procesamiento local	Comunicación
(Boissier & Kurzynski, 2018)	• Número de tuplas que es posible saltarse si se ejecutan todas las consultas en la carga de trabajo.	Sí	No
(Kechar & Nait-Bahloul, 2017)	• Número de operaciones de E/S	Sí	No
(Nam et al., 2018)	• Costo total de operaciones de E/S (*equi-joins* entre tablas de partición y replicación).	Sí	No
(Letrache et al., 2019)	• Costo de E/S antes y después de la fragmentación (número de páginas necesarias para cargar el cubo o las particiones en memoria).	Sí	No

Solo siete métodos cumplieron con completitud, modelo de costo y facilidad de implementación. La Tabla 3.3 compara los métodos considerando el enfoque utilizado para la fragmentación de las tablas de dimensiones, la estrategia empleada para la selección del mejor esquema de fragmentación, además se observa que todos consideran la restricción del máximo número de fragmentos que se deben generar, también se incluyen algunas características positivas y negativas de cada método; se agregó el método de (Bouchakri et al., 2014) porque aunque el artículo trata sobre fragmentación incremental, primero presentan un método de fragmentación estática.

Tabla 3.3. Características de los métodos que cumplieron con los criterios de comparación.

Método	Fragmentación	Selección	Restricción	Características
(Bellatreche et al., 2000)	Afinidad	Algoritmo voraz	Sí	-Modelo de costo complejo.
(Bellatreche & Boukhalfa, 2005)	Predicados mintérmino	Algoritmo genético (GA)	Sí	-Utilizado en métodos más recientes.
(Bellatreche et al., 2006)	Predicados mintérmino	GA con recocido simulado	Sí	+Genético mejorado
(Bellatreche & Woameno, 2009)	Cualquier método de fragmentación primaria	Reducir tamaño de resultados intermedios	Sí	+Fácil de implementar
(Barr & Bellatreche, 2010)	Predicados mintérmino	Optimización de Colonia de Hormigas (ACO)	Sí	-Menos eficiente que (Barr et al., 2018) .
(Bouchakri, Bellatreche, Faget, et al., 2014)	Predicados mintérmino	GA	Sí	+Completo
(Toumi et al.,	Atracción y	PSO	Sí	+Más rápido y más

Método	Fragmentación	Selección	Restricción	Características
2015)	agrupamiento jerárquico			efectivo que el algoritmo genético.
(Barr et al., 2018)	No se especifica	Programación lineal	Sí	+Más eficiente que el basado en ACO.

3.1.4 Clasificación de los métodos de fragmentación según su enfoque

Se han propuesto varios enfoques para resolver el problema de la fragmentación horizontal tanto en las bases de datos tradicionales como en los almacenes de datos, este trabajo se enfoca en la fragmentación del almacén de datos que se clasifica en tres ejes: el enfoque basado en predicados, el enfoque basado en afinidad, y el enfoque basado en costos, y se consideró si la fragmentación era dinámica o estática en cada caso.

3.1.5 Análisis comparativo de métodos de fragmentación horizontal según el enfoque para seleccionar el HFS

La fragmentación horizontal (HF) es una técnica de optimización ampliamente utilizada para reducir el costo de ejecución de las consultas de unión en estrella en DW relacionales. Estas consultas contienen combinaciones múltiples y operaciones de selección complejas que involucran tablas de hechos y tablas de dimensiones múltiples. Las consultas de unión en estrella son las consultas OLAP más costosas y, a menudo, implican un alto nivel de costo de comunicación (Ramdane et al., 2019a). En esta sección se presentan las principales características de las 49 técnicas de HF analizadas agrupándolas en 1) Métodos estáticos sin restricción en el número de fragmentos; 2) Métodos estáticos con esa restricción, y 3) Métodos dinámicos.

3.1.6 Métodos de Fragmentación Horizontal Estática sin Restricción

La mayoría de los artículos (40 de 49) se centran en una selección estática de un esquema de fragmentación. 21 son métodos sin restricciones. Se clasificaron estos 21 artículos teniendo en cuenta la estrategia de seleccionar un esquema de fragmentación, lo que resultó en cinco grupos: 1) Predicados minterm (Ezeife, 2001; Liu et al., 2013; Noaman & Barker, 1999); 2) Costo (Furtado, 2004; Kechar & Nait-Bahloul, 2019; Liu et al., 2013); 3) Minería de datos (Amina & Boukhalfa, 2013; Boissier & Kurzynski, 2018; Ramdane et al., 2018; Sun et al., 2014); 4) Grafo (de Aguiar et al., 2007; Parchas et al., 2020), y 5) Otro (Barkhordari & Niamanesh, 2018; Barr, 2012; Brki, 2012; Mateus et al., 2016). La Figura 3.13 muestra esta clasificación.

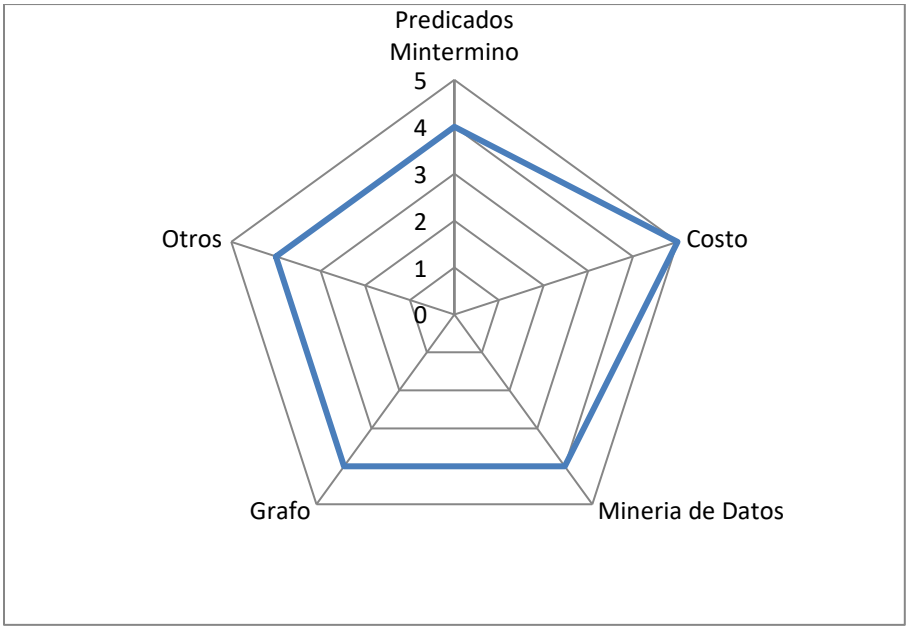


Figura 3.13. Número de métodos estáticos no restringidos por la estrategia para obtener el HFS

3.1.7 Fragmentación Horizontal Estática con Restricción

Solo 19 de 40 métodos estáticos están limitados en el número de fragmentos. Como se ve en la Figura 3.14, estas 19 técnicas se agruparon en cuatro categorías: 1) algoritmos basados en afinidad (Bellatreche et al., 2000, 2008); 2) Algoritmos basados en modelos de costos (Barr & Bellatreche, 2010; Bellatreche et al., 2006; Bellatreche & Benkrid, 2009; Bellatreche & Boukhalfa, 2005; Bellatreche & Woameno, 2009; Benkrid et al., 2008; Dimovski et al., 2010); 3) Algoritmos basados en minería de datos (Hanane & Kamel, 2014, 2014; Karima et al., 2010; Ramdane et al., 2019a, 2019b; Toumi et al., 2015) y 4) Enfoques basados en metaheurísticas (Barr, 2013; Rafid, 2013; Ramdane et al., 2019b).

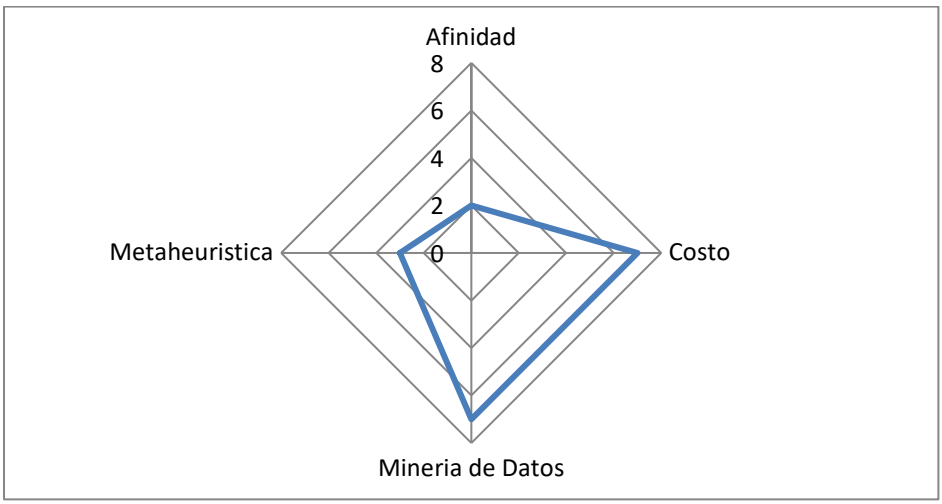


Figura 3.14. Número de métodos estáticos restringidos por la estrategia para obtener el HFS

3.1.8 Fragmentación Horizontal Dinámica

La evolución de los almacenes de datos y la naturaleza *ad hoc* de las consultas OLAP contribuyeron al desarrollo de algoritmos HF incrementales o dinámicos (Bouchakri, Bellatreche, & Faget, 2014). La Figura 3.15 muestra que los 9 métodos dinámicos de HF se basan principalmente en metaheurísticas (Bellatreche et al., 2013; Bouchakri, Bellatreche, & Faget, 2014; Bouchakri, Bellatreche, Faget, et al., 2014) o aprendizaje de refuerzo profundo (DRL) (Hilprecht et al., 2020).

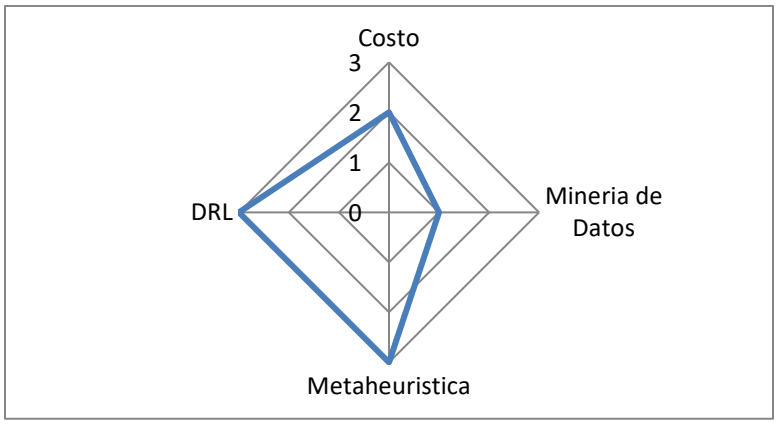


Figura 3.15. Número de métodos dinámicos por estrategia para obtener el HFS

3.2. Estudio de Algoritmos de Árboles de Decisión

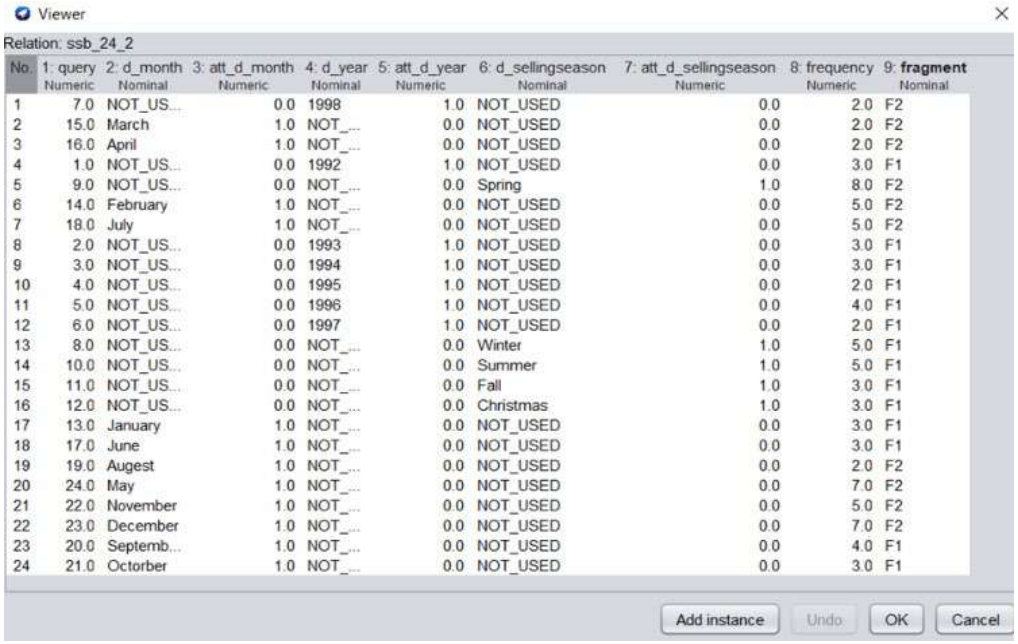
Se realizó el estudio de algoritmos de árboles de decisión para seleccionar el mejor para utilizarlo en el método de fragmentación horizontal propuesto, para esto fue necesario en primer lugar la instalación del *benchmark* SSB (*Star Schema Benchmark*) que está construido a partir de TPC-H (*Transaction Performance Council Benchmark*™H). Después de la implementación del *benchmark* en PostgreSQL se generaron diversos conjuntos de datos utilizando los Algoritmos 1 y 2, los cuales son una modificación de los algoritmos presentados en (Rodríguez et al., 2014). El Algoritmo 1 toma como entrada una matriz de uso de predicados (PUM) y el número máximo de fragmentos (W) y usa el Algoritmo 2 para generar W-1 conjuntos de datos. El conjunto de datos resultante considerando 24 consultas y 2 fragmentos se ilustra en la Figura 3.16, como se observa, el atributo utilizado como etiqueta de clase fue *fragment*.

Data: PUM of the fact table FT (a set of predicates $Pr=\{p_1, p_2, ..., p_r\}$, the selectivity sel_i of each predicate p_i, a set of queries $Q=\{q_1, q_2, ..., q_s\}$, the frequency f_k of each query q_k}, W
Result: D={data_set$_2$, data_set$_3$,..., data_set$_W$}
for each step$_i$ ∈PT |2≤i≤W **do**
    getPPM(PUM, PPM);
    select two nodes with máximum partitioning profit;
    divide the nodes;
    generate data_set$_i$;
end;

Algoritmo 1. Generar conjuntos de datos

```
Data: PUM
Result: PPM: Partitioning Profit Matrix
for each p_i ∈Pr | 1≤i≤r-1 do
    for each p_j ∈Pr | i+1≤j≤r do
        IRT=0;
        DIT=0;
        partitioning_profit=0;
        for each q_k ∈Q | 1≤k≤s do
            if PUM(q_k,p_i)=1 & PUM(q_k,p_j)=1 then
                IRT=IRT+f_k*(sel_i+sel_j);
            else
                if PUM(q_k,p_i)=1 then
                    DIT=DIT+ f_k*sel_j
                else
                    if PUM(q_k,p_j)=1 then
                        DIT=DIT+ f_k*sel_i
                    end
                end
            end
        end
        partitioning_profit=IRT-DIT;
        PPM(p_i,p_j)=partitoning_profit;
    end
end
```

Algoritmo 2. getPPM



No.	1: query Numeric	2: d_month Nominal	3: att_d_month Numeric	4: d_year Nominal	5: att_d_year Numeric	6: d_sellingseason Nominal	7: att_d_sellingseason Numeric	8: frequency Numeric	9: **fragment** Nominal
1	7.0	NOT_US...	0.0	1998	1.0	NOT_USED	0.0	2.0	F2
2	15.0	March	1.0	NOT_...	0.0	NOT_USED	0.0	2.0	F2
3	16.0	April	1.0	NOT_...	0.0	NOT_USED	0.0	2.0	F2
4	1.0	NOT_US...	0.0	1992	1.0	NOT_USED	0.0	3.0	F1
5	9.0	NOT_US...	0.0	NOT_...	0.0	Spring	1.0	8.0	F2
6	14.0	February	1.0	NOT_...	0.0	NOT_USED	0.0	5.0	F2
7	18.0	July	1.0	NOT_...	0.0	NOT_USED	0.0	5.0	F2
8	2.0	NOT_US...	0.0	1993	1.0	NOT_USED	0.0	3.0	F1
9	3.0	NOT_US...	0.0	1994	1.0	NOT_USED	0.0	3.0	F1
10	4.0	NOT_US...	0.0	1995	1.0	NOT_USED	0.0	2.0	F1
11	5.0	NOT_US...	0.0	1996	1.0	NOT_USED	0.0	4.0	F1
12	6.0	NOT_US...	0.0	1997	1.0	NOT_USED	0.0	2.0	F1
13	8.0	NOT_US...	0.0	NOT_...	0.0	Winter	1.0	5.0	F1
14	10.0	NOT_US...	0.0	NOT_...	0.0	Summer	1.0	5.0	F1
15	11.0	NOT_US...	0.0	NOT_...	0.0	Fall	1.0	3.0	F1
16	12.0	NOT_US...	0.0	NOT_...	0.0	Christmas	1.0	3.0	F1
17	13.0	January	1.0	NOT_...	0.0	NOT_USED	0.0	3.0	F1
18	17.0	June	1.0	NOT_...	0.0	NOT_USED	0.0	3.0	F1
19	19.0	Augest	1.0	NOT_...	0.0	NOT_USED	0.0	2.0	F2
20	24.0	May	1.0	NOT_...	0.0	NOT_USED	0.0	7.0	F2
21	22.0	November	1.0	NOT_...	0.0	NOT_USED	0.0	5.0	F2
22	23.0	December	1.0	NOT_...	0.0	NOT_USED	0.0	7.0	F2
23	20.0	Septemb...	1.0	NOT_...	0.0	NOT_USED	0.0	4.0	F1
24	21.0	Octorber	1.0	NOT_...	0.0	NOT_USED	0.0	3.0	F1

Figura 3.16. Conjunto de datos con 24 consultas y 2 fragmentos

Posteriormente, se trasladaron los datos a Weka 3.9.4 para que a través de esta se aplicaran los diferentes algoritmos para inducción de árboles de decisión que están disponibles en la versión (DecisionStump, HoeffdingTree, J48, Logistic Model Tree, RandomForest, RandomTree y REPTree). La evaluación de los algoritmos de árboles se realizó usando validación cruzada de 10 pliegues y se consideraron cuatro métricas: precisión, *recall*, *F-measure* y área ROC. Se hicieron 32 experimentos usando los 7 algoritmos de árboles de

decisión, con 12, 24 y 50 consultas, considerando de 2 a 5 fragmentos, por razones de espacio solo se muestran los resultados para las 24 y 50 consultas, mismos que se aprecian en las Figuras 3.17 a 3.24 de este documento. Cabe mencionar que en dichas gráficas cuando los algoritmos no obtuvieron un resultado para todas las clases del conjunto de datos en Weka se muestra como cero ese resultado. Finalmente se determinó que los dos mejores algoritmos fueron RandomForest y J48 por lo que se decide seleccionar a J48, ya que es más eficiente en la construcción del modelo, debido a que la complejidad computacional de J48 dado un conjunto de datos D es $O(n \times |D| \times log(|D|))$ donde n es el número de atributos que describen las tuplas en D y $|D|$ es el número de tuplas de entrenamiento en D (Han et al., 2012). Por el contrario, la complejidad computacional para construir un bosque de M árboles aleatorios es $O(M \times K \times \tilde{N}^2 \times \log(\tilde{N}))$, donde K es el número de variables extraídas aleatoriamente en cada nodo y $\tilde{N}=0.632|D|$ (Louppe, 2015).

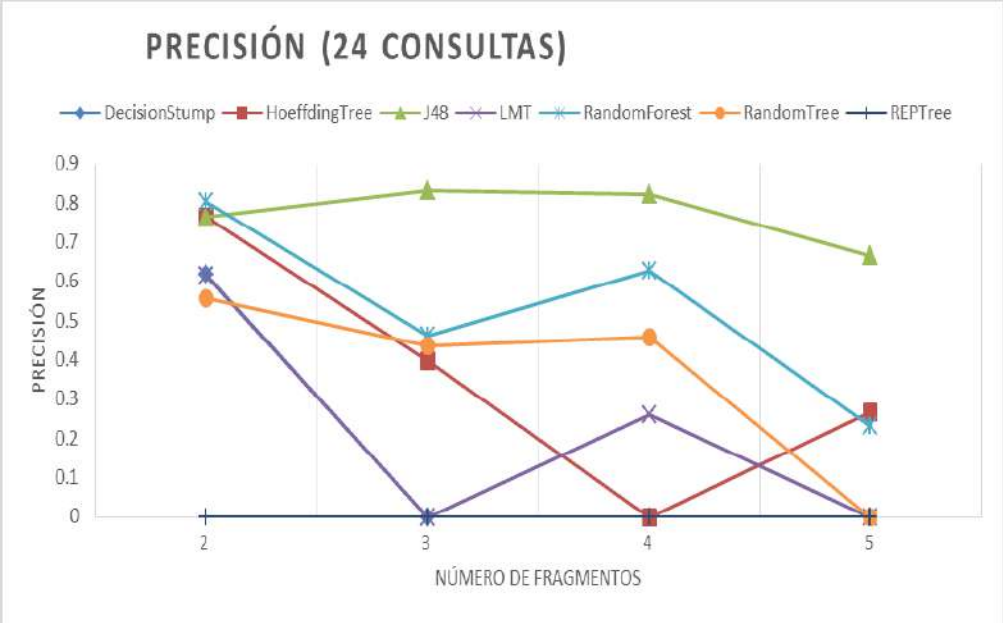


Figura 3.17. Resultados para precisión con data set de 24 consultas

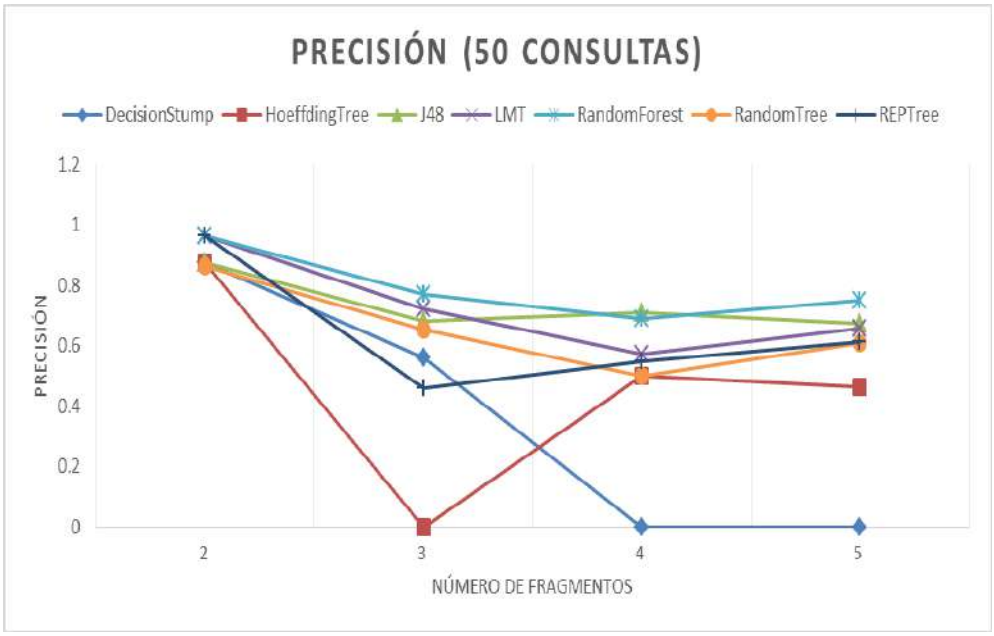


Figura 3.18. Resultados para precisión con data set de 50 consultas

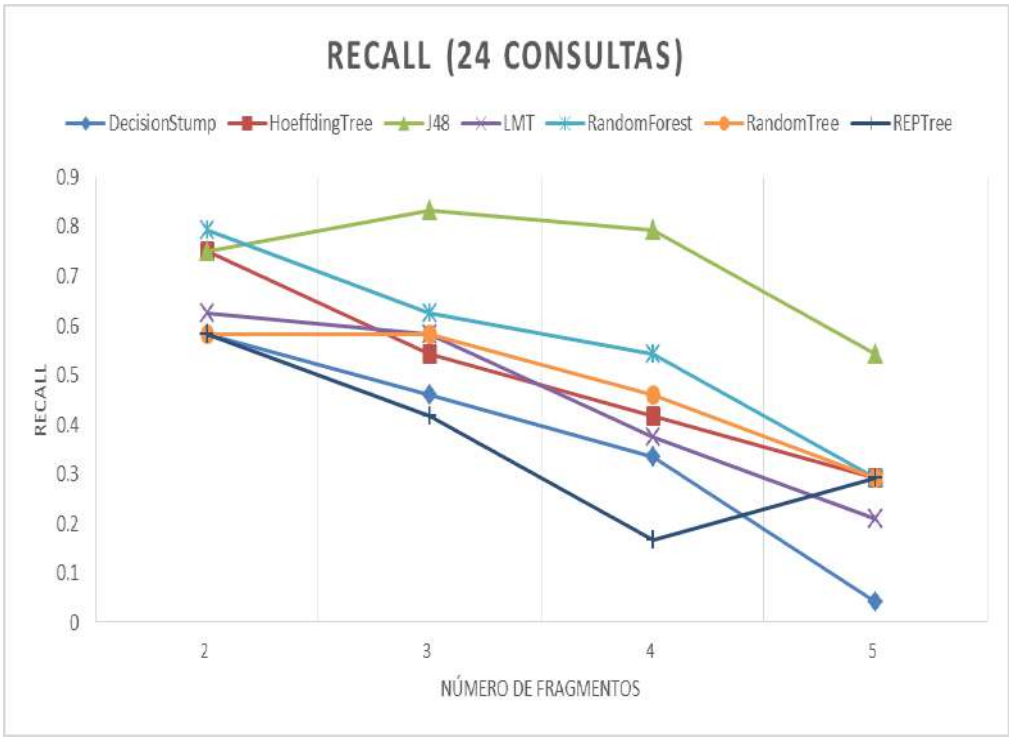


Figura 3.19. Resultados para Recall con data set de 24 consultas

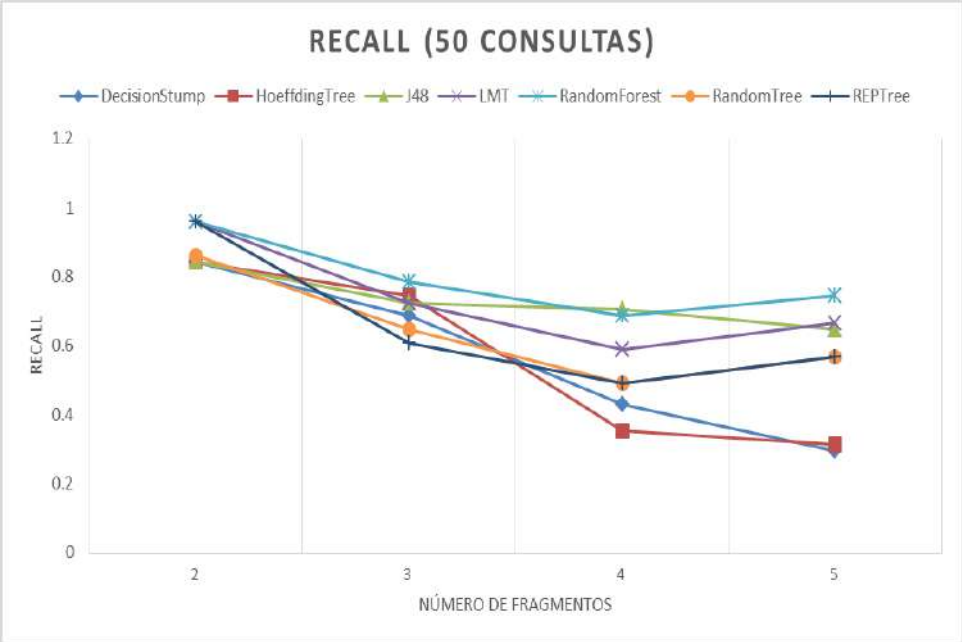


Figura 3.20. Resultados para Recall con data set de 50 consultas

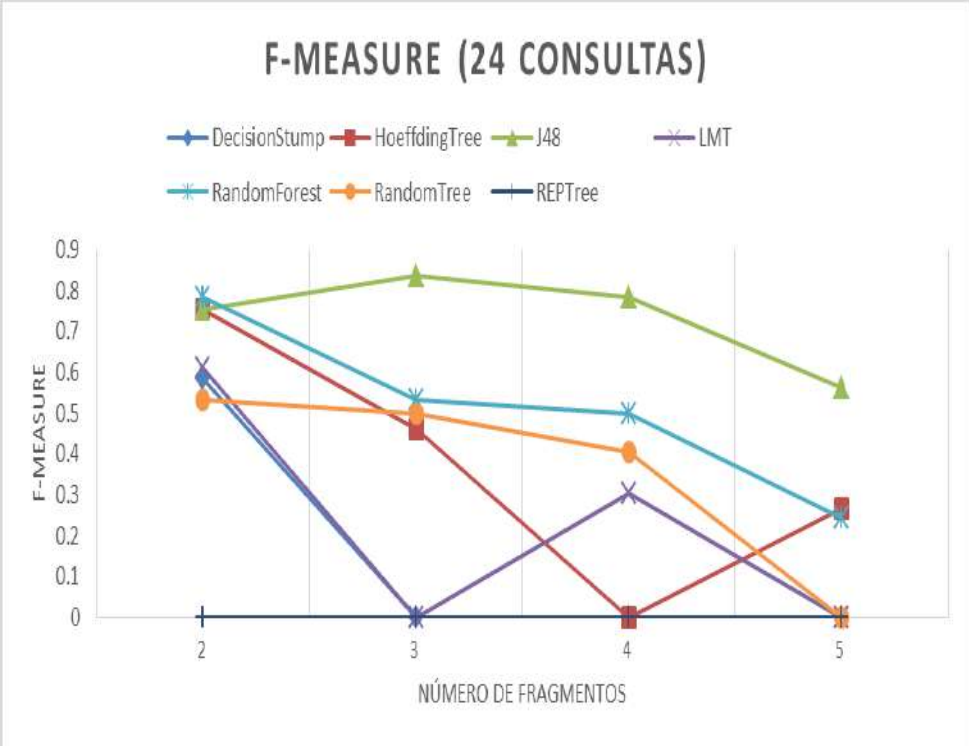


Figura 3.21.Resultados para *F-measure* con data set de 24 consultas

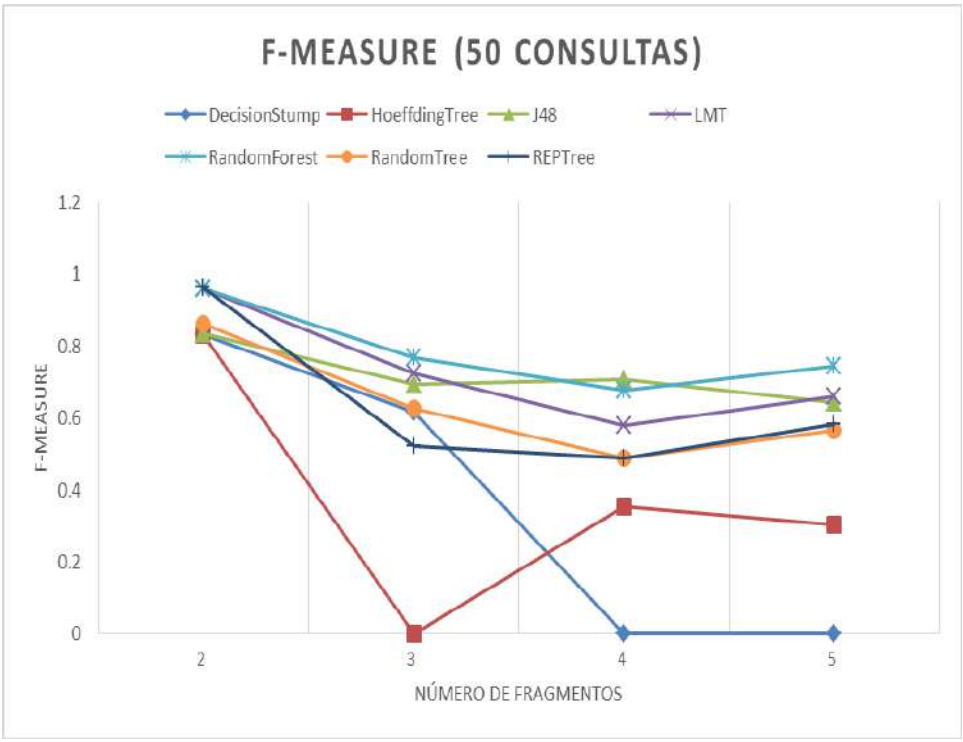


Figura 3.22.Resultados para *F-measure* con data set de 50 consultas

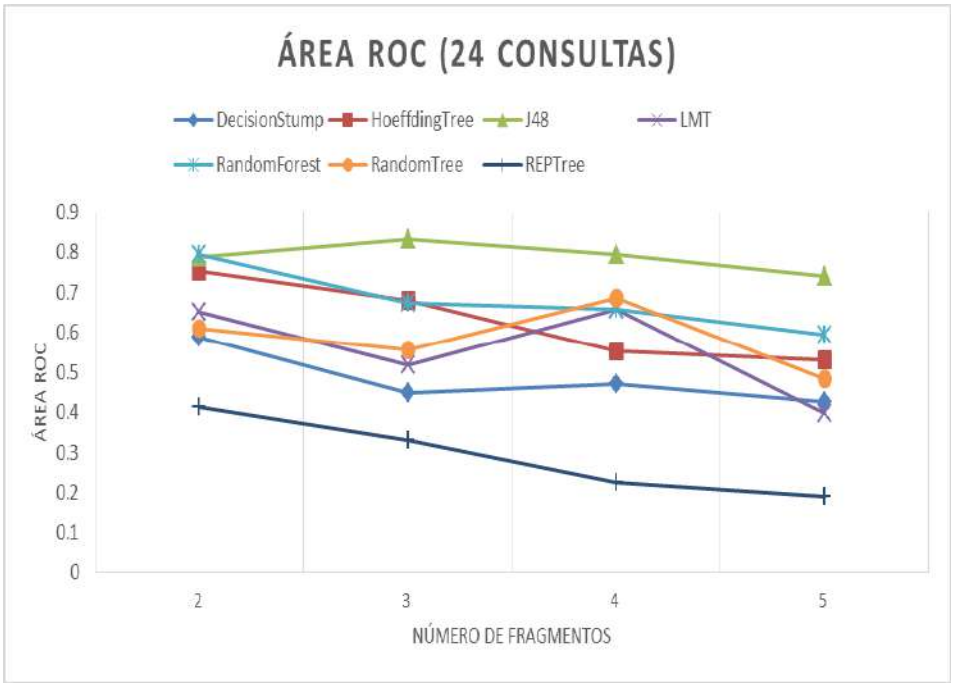


Figura 3.23.Resultados para Área Roc con data set de 24 consultas

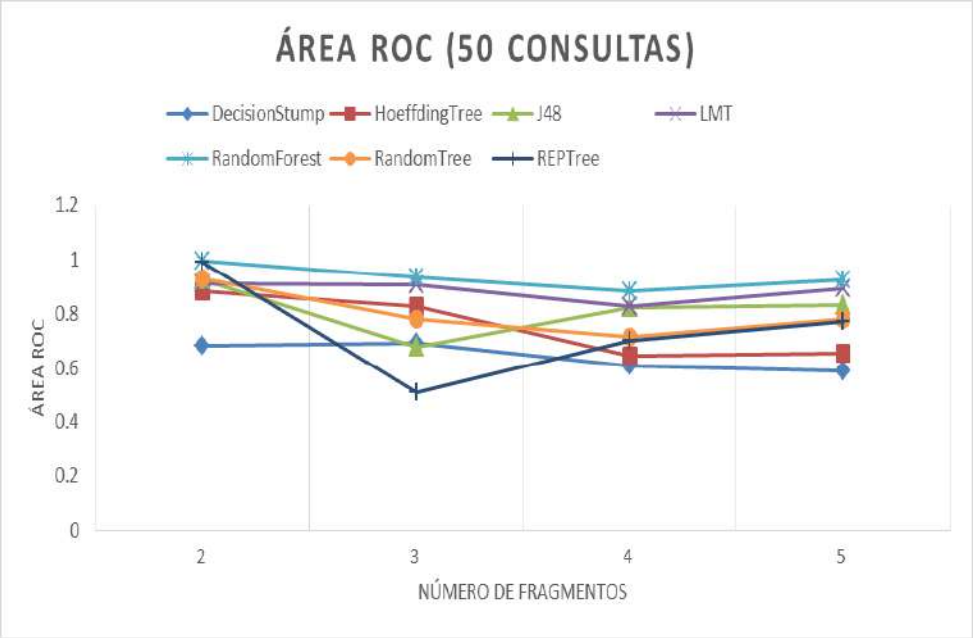


Figura 3.24.Resultados para Área Roc con data set de 50 consultas

Un ejemplo del árbol generado por J48 se visualiza en la Figura 3.25.

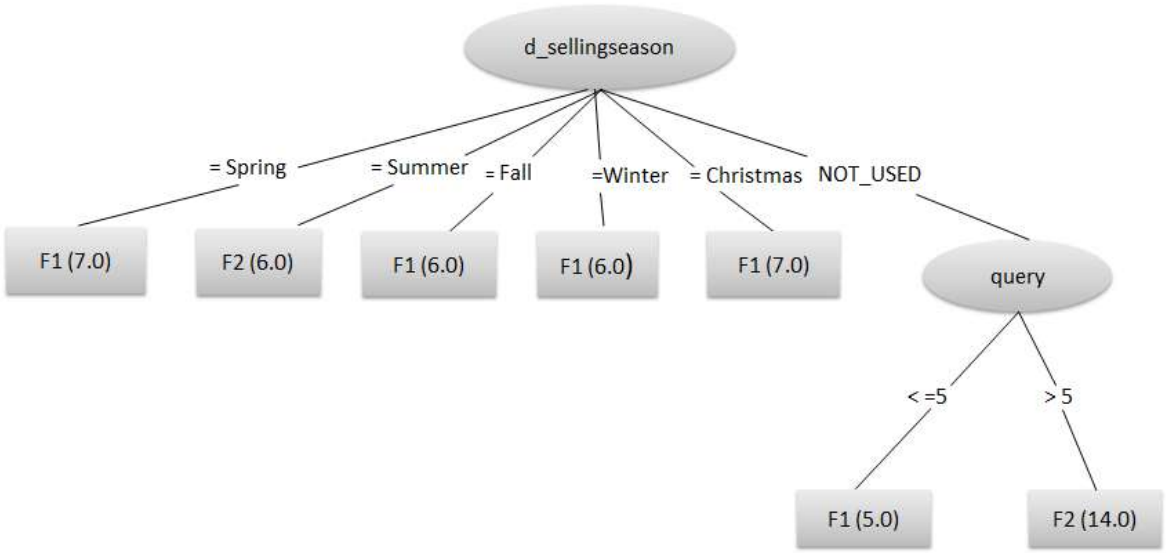


Figura 3.25. Árbol de decisión generado por J48

3.3. Diseño del Data Warehouse Turístico

Por otro lado, se trabajó en el diseño del DW siguiendo la metodología de (Kimball & Ross, 2016), esta se visualiza en la Figura 3.26. Las etapas que se llevaron a cabo fueron el Diseño de la arquitectura técnica, la Selección de productos e instalación, el Modelado

dimensional, el Diseño físico, el Diseño ETL (Extracción, Transformación y Carga) y el Diseño de aplicaciones *Businnes Intelligence*, mismas que se describen en esta sección.

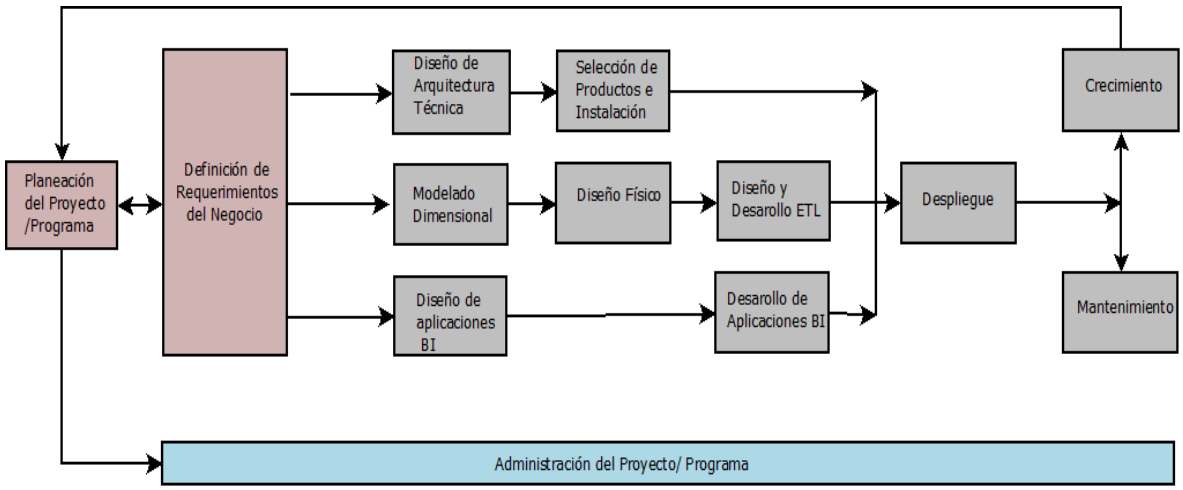


Figura 3.26. Metodología Kimball & Ross para el diseño de un DW.

3.3.1. Planificación del DW Turístico

Como parte de la fase de planeación del DW Turístico (DWT) se estudió el turismo para conocer el área de este donde finalmente se desarrolló el DWT, inicialmente se contempló el turismo doméstico (turismo que se realiza en México por habitantes con residencia mexicana) como el área ideal donde se aplicaría el proyecto debido a que es una área de gran interés para SECTUR (Secretaría de Turismo), quien en sus requerimientos del sector del Fondo Sectorial para la Investigación, el Desarrollo y la Innovación Tecnológica en Turismo 2017, en conjunto con CONACYT, marcó el turismo doméstico como un área donde busca desarrollar herramientas que ayuden a su análisis, sin embargo, al investigar las fuentes de datos disponibles en esta área se encontraron muy pocos datos abiertos disponibles para su descarga en las principales fuentes turísticas consultadas: SECTUR (Secretaría de turismo); CEDOC (Centro de Documentación Turística); ICTUR (Instituto de Competitividad Turística); INEGI (Instituto Nacional de Estadística y Geografía); BANXICO (Banco de México); DATATUR (Análisis Integral del Turismo); SIIMT (Sistema Integral de Información de Mercados Turísticos); CONAPO (Consejo Nacional de Población); SEPOMEX (Servicio Postal Mexicano), así como empresas privadas, por lo cual se decidió utilizar en este proyecto de tesis datos abiertos existentes del turismo internacional que llega a México debido principalmente a que en este rubro existen bases de datos descargables en diferentes organizaciones que regulan la actividad turística en México.

Posteriormente y una vez teniendo claro que el área turística en la que se enfocaría esta tesis sería el turismo internacional o receptor, se procedió a analizar las fuentes de datos

disponibles, para esto se revisaron todos los datos abiertos disponibles en los diferentes organismos que regulan la actividad turística como son: SECTUR, ICTUR, INEGI, BANXICO, DATATUR, OMT y el SIMMT. La Tabla 3.4 muestra las diferentes fuentes consultadas.

Tabla 3.4. Fuentes turísticas consultadas

Organismo	Limitaciones	Disponibilidad de datos	Ventajas	Estandarización de datos
DATATUR	Algunos reportes no se encuentran estandarizados, ni existen para todos los años. Existe duplicidad de datos, datos faltantes y la información de ciertos años no aparece.	Sí	Aborda varias áreas del turismo, tiene información tanto de accesos, atractivos y facilidades turísticas.	No en todos los casos.
CEDOC	No cuenta con bases de datos abiertas descargables.	No	Cuenta con información turística detallada.	No
INEGI	Contiene datos repetidos o faltantes.	Sí	Contiene archivos en Excel, csv, entre otros, actualizados por año y aborda diferentes áreas del turismo.	No
BANXICO	No cuenta con información de la actividad turística desde 2008, debido a la implementación de la cuenta satélite del turismo del INEGI.	Sí	Hasta 2008 contaba con información relevante en cuanto a gasto turístico generado y formas de pago realizadas en el territorio mexicano.	No
OMT	Los archivos solo están disponibles bajo pago o solicitud debidamente requisitada, sin embargo, el trámite de esto no siempre es garantía de obtener los archivos.	No	Cuenta con información detallada y actualizada en todas las áreas turísticas incluyendo bases de datos de turismo doméstico.	Sí
SIMMT	Contempla aspectos cualitativos de materia turística como proyectos y publicaciones, pero no	No	Cuenta con información importante en materia turística	No

Organismo	Limitaciones	Disponibilidad de datos	Ventajas	Estandarización de datos
	contiene información estadística descargable.		para consulta, además permite descargar artículos.	

Posterior a lo antes descrito, se determinaron los datos que se utilizaron para el desarrollo del DW Turístico quedando como fuentes principales los de la OMT, SECTUR E INEGI, debido a que son los que contienen datos turísticos abiertos, contienen pocos datos faltantes, permiten analizar diferentes áreas del turismo y además están actualizados.

3.3.2. Definición de requerimientos

Los principales interesados en este Data Warehouse son los especialistas en turismo (trabajadores de la industria turística, organismos gubernamentales, organismos turísticos, empresarios, profesores y alumnos).

Requerimientos funcionales

- Existe un administrador que cuenta con todos los permisos y el usuario final que solo tiene acceso a la realización de consultas.
- Se consideró que el Data Warehouse guardaría información desde 2007 a 2018 sobre turistas internacionales y actividad hotelera extraída del compendio.
- La obtención de reportes y gráficas que muestren el número de turistas internacionales fronterizos y al interior, el gasto total y el gasto promedio por año, seleccionar un año en específico y mostrar por trimestre o por mes. En cuanto a la actividad hotelera, es posible analizar el número de cuartos ocupados, porcentaje de ocupación, el número de noches, el número de llegadas, y la estadía de turistas nacionales y extranjeros, con los mismos niveles en la jerarquía de tiempo.
- Es posible seleccionar alguna de las medidas: para Turistas internacionales: número de turistas, gasto o gasto medio, así como las dimensiones que se deseen incluir en el reporte: tiempo o turismo; mientras que para Actividad hotelera: número de cuartos ocupados, porcentaje de ocupación, número de noches, llegadas y estadía, además de las dimensiones deseadas: tiempo, lugar y turista; es posible que el reporte tenga categorías cuando se seleccionan varios niveles de una jerarquía, por ejemplo, si se desea ver el número de turistas por año, trimestre y mes.
- La información obtenida abarca:

  Turistas internacionales

  1. Número de turistas internacionales por año/trimestre/mes.
  2. Gasto total de turistas internacionales por año/trimestre/mes.
  3. Gasto medio de turistas internacionales por año/trimestre/mes.
  4. Número de turistas de internación por año/trimestre/mes.
  5. Número de turistas fronterizos por año/trimestre/mes.
  6. Gasto total de turistas de internación por año/trimestre/mes.

7. Gasto total de turistas fronterizos por año/trimestre/mes.
8. Gasto medio de turistas de internación por año/trimestre/mes.
9. Gasto medio de turistas fronterizos por año/trimestre/mes.
10. Número de turistas de internación por vía aérea por año/trimestre/mes.
11. Número de turistas de internación por vía terrestre por año/trimestre/mes.
12. Número de turistas fronterizos peatones por año/trimestre/mes.
13. Número de turistas fronterizos automovilistas por año/trimestre/mes.

Actividad hotelera

1. Número de cuartos totales por año/trimestre/mes.
2. Porcentaje de ocupación total por año/trimestre/mes.
3. Número de noches totales por año/trimestre/mes.
4. Número de llegadas totales por año/trimestre/mes.
5. Estadía total por año/trimestre/mes.
6. Número de cuartos totales por corredor turístico/centro turístico/estado.
7. Porcentaje de ocupación total por corredor turístico/centro turístico/estado.
8. Número de noches totales por corredor turístico/centro turístico/estado.
9. Número de llegadas totales por corredor turístico/centro turístico/estado.
10. Estadía total por corredor turístico/centro turístico/estado.
11. Número de cuartos nacionales/extranjeros.
12. Porcentaje de ocupación nacionales/extranjeros.
13. Número de noches nacionales/extranjeros.
14. Número de llegadas nacionales/extranjeros.
15. Estadía total por nacionales/extranjeros.

Requerimientos no funcionales

- Se tiene considerado utilizar herramientas OLAP y ETL convenientes para la carga de datos.
- Se pretende utilizar el sistema gestor de bases de datos que represente más beneficios para el desarrollo del Data Warehouse. Teniendo en consideración PostgreSQL, MySQL, Microsoft SQL Server y Oracle Database.
- Se considera representar la información utilizando un esquema de constelación de hechos, con una tabla de hechos Turistas internacionales que guarde tres medidas: número de turistas internacionales (en miles), gasto total (en millones de dólares) y gasto medio (en dólares). Se tendrán dos dimensiones: Tiempo, la cual permitirá visualizar cada medida por mes, trimestre y año; Turismo, para obtener información por tipo de turismo (Interior o Fronterizo) y vía de acceso (Aéreo o Terrestre, Peatones o Automovilistas). Además, se tendrá una segunda tabla de hechos Actividad hotelera con cinco medidas: cuartos ocupados (en números exactos), ocupación (en porcentaje), número de noches (cantidad total de noches), número de llegadas (cantidad total de llegadas de turistas) y estadía (estadía promedio por turista). Para esta tabla de hechos se tendrán tres dimensiones: Tiempo, con los mismos niveles jerárquicos especificados para la primera tabla de hechos; Lugar, la

cual permitirá visualizar la actividad hotelera por corredor turístico, centro turístico y estado, y finalmente, Turista, con la cual será posible analizar el tipo de turista (Nacional y Extranjero). La Tabla 3.5 describe la matriz colectiva preliminar que usa el DW turístico.

Tabla 3.5 Matriz colectiva empresarial preliminar

Procesos	Dimensiones			
	Tiempo	Lugar	Turista	Turismo
Visitantes internacionales a México mensual	X			X
Cuartos ocupados mensual	X	X	X	X
Llegada de turistas mensual	X	X	X	X
Turistas noche mensual	X	X	X	
Porcentaje de ocupación mensual	X	X	X	X
Estadía mensual	X	X	X	X

3.3.3. Diseño de arquitectura técnica

La arquitectura técnica establecida para la construcción del DWT está conformada por tres grandes capas: datos, *back room* y *front room*.

Datos: Las fuentes que se utilizaron para la construcción del DWT corresponden a los reportes generados por SEGOB (Secretaria de Gobierno), SCT (Secretaria de Comunicaciones y Transporte), UWTO (Organización Mundial del Turismo) e INEGI (Instituto Nacional de Estadística y Geografía) en México para el 2018 y que están disponible en la página de la SECTUR (Secretaria de Turismo) (DATATUR, 2018). Para la construcción del DWT se tomaron en cuenta los datos que se visualizan en las Tablas 3.7 y 3.8.

Back room: Es el área del DW encargado de extraer y preparar los datos, es aquí donde se describe cómo se realizó el proceso ETL (Extracción, Transformación y Carga) para el desarrollo del DWT. Se parte de los datos fuentes que manejan los diferentes organismos turísticos antes mencionados. En la Tabla 3.8 se observan los datos correspondientes a cada tabla establecida y la fuente de la que los datos se extrajeron. La transformación implica la selección de los datos relevantes y estructurarlos de tal forma que facilite la carga en la base de datos. Finalmente, los datos quedan listos para utilizarse por herramientas OLAP.

Front room: El Data Warehouse está estructurado de tal forma que se visualice la información en el modelo dimensional correspondiente a visitantes internacionales y actividad hotelera, con respecto a los requerimientos realizados por turismo, tiempo, turista y lugar.

Tabla 3.6. Datos disponibles (SECTUR)

Visitantes internacionales	Actividad hotelera
Número de turistas internacionales	Cuartos ocupados
Gasto total	Porcentaje de ocupación
Gasto medio	Turistas noches
	Llegada turistas
	Estadía

Tabla 3.7. Datos disponibles (SECTUR)

Turista	Lugar	Tiempo	Turismo
Tipo de turista	Corredor turístico	Mes	Tipo de turismo
	Centro turístico	Trimestre	Vía de acceso
	Estado	Año	

Tabla 3.8. Mapeo de datos turísticos

Tabla	Fuente de datos
Turismo	Visitantes internacionales a México mensual.
Lugar	Actividad hotelera en centros turísticos seleccionados.
Turista	Actividad hotelera en centros turísticos seleccionados.
Tiempo	Resumen mensual.
Actividad hotelera	Actividad hotelera en centros turísticos seleccionados.
Visitantes internacionales	Visitantes internacionales a México mensual.

3.3.4. Selección de productos e instalación

Se llevó a cabo el análisis de diferentes SGBD para determinar el más conveniente para implementar el DWT. La Tabla 3.9 muestra las características más sobresalientes del análisis realizado. Una vez realizado el estudio comparativo, se llegó a la conclusión de utilizar el SGBD PostgreSQL, ya que es de código abierto, además de ser multiplataforma, robusto y compatible con la mayor parte de las herramientas seleccionadas para la

construcción del DWT. También, se analizaron diferentes herramientas ETL y OLAP *open source* para encontrar las más adecuadas para desarrollar el Data Warehouse. Finalizado el estudio comparativo, se concluyó utilizar Pentaho, ya que la edición comunitaria cuenta con una plataforma de integración y analítica de datos, así como herramientas para el diseño de informes, de agregación y de cubos OLAP, además de un editor de metadatos. Por otra parte, maneja distintos archivos como csv, xsl, txt, entre otros, y es compatible con los SGBD que se analizaron.

Tabla 3.9. Análisis de SGBD

SGBD	Licencia	Sistema operativo	Características	Tipos de partición
PostgreSQL 12	PostgreSQL (*open source*)	BSD (*Berkeley Software Distribution*), Linux, macOS, Solaris, Windows	Fiabilidad, robustez y buen rendimiento.	Range, Hash y List
MySQL 8.0	Licencia de producto de Oracle (comercial) y *General Public Licence* versión 2, (GPLv2)	FreeBSD, Linux, macOS, Solaris, Windows	Facilidad de uso, fiabilidad y buen rendimiento.	Range, Hash, List y Key.
Microsoft SQL Server 2019	Propietaria, aunque cuenta con versiones *Developer* y *Express* gratuitas.	Windows y Linux	Seguridad, mejor rendimiento y compatibilidad con BI y análisis	Range
Oracle Database 19c	Licencia de producto de Oracle (comercial)	Solaris, Windows y Linux	Escalabilidad, fiabilidad, seguridad y buen rendimiento.	Range, Hash, List, Compuesta, Intervalos, Referencial, Basada en columnas virtuales.

3.3.5. Modelado dimensional

Con los requerimientos obtenidos, se procede a la elaboración del modelado dimensional de acuerdo a las medidas, dimensiones y jerarquías establecidas (Figura 3.27).

Dimensiones

- Turista
- Lugar

- Tiempo
- Turismo

Medidas

- Número de turistas internacionales
- Gasto total
- Gasto medio
- Cuartos ocupados
- Porcentaje ocupación
- Turistas noches
- Llegada turistas
- Estadía

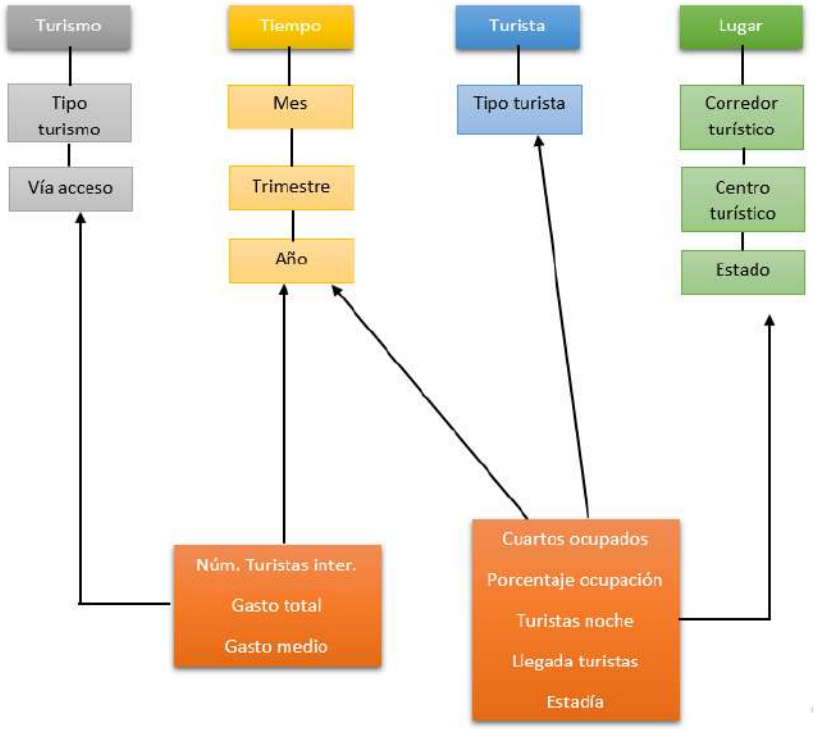


Figura 3.27. Medidas, dimensiones y jerarquías.

3.3.6. Diseño físico

El modelo físico se muestra en la Figura 28 y se utiliza para representar todas las estructuras de las tablas que conforman el DWT.

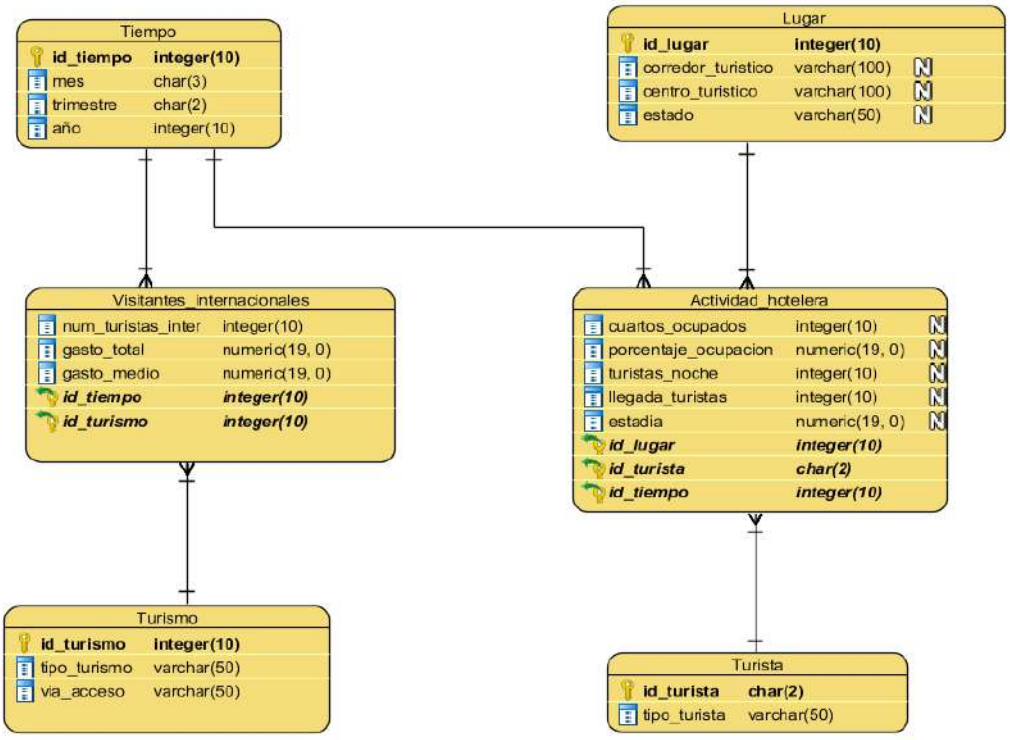


Figura 3.28. Modelo físico del DWT

3.3.7. Diseño del ETL

El Sistema ETL para la construcción del DWT consiste en una serie de pasos para cada fase de extracción, transformación y carga.

Extracción: Esta fase se centra en extraer los datos de origen. En este caso los datos de origen se encontraban almacenados en archivos Excel, por lo tanto, se creó un paso *Microsoft Excel input* para su extracción. Dentro del cual se realizaron tres procedimientos: el primero se encargó de seleccionar el archivo Excel a transformar, el segundo extrajo la hoja del archivo en donde se encuentran los datos y el tercero obtuvo los datos.

Transformación: La fase de transformación consta de dos pasos, *Add sequence* y *Select values,* mismos que se visualizan en la Figura 3.29. *Add sequence* es el que genera el ID de las tablas, mientras que *Select values* permite seleccionar los campos requeridos para la tabla añadiendo el ID creado, en este paso también es posible renombrar los atributos.

Carga: En la fase carga se creó un paso *Table output* para exportar los datos a la base de datos, como se aprecia en la Figura 3.29. En este procedimiento es donde se genera el código SQL para crear las tablas de hechos y de dimensiones en la base de datos.

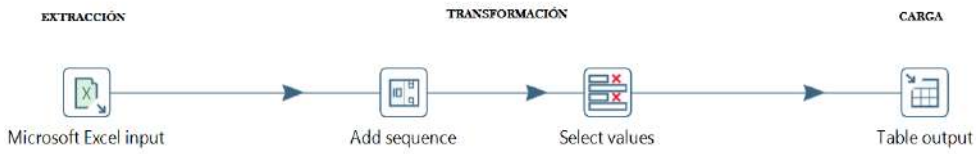


Figura 3.29. Sistema ETL del proyecto

3.3.8. Diseño OLAP

En el proceso OLAP es donde se generaron los cubos que concentran toda la información obtenida en la base de datos para su posterior creación de informes. La Figura 3.30 describe la estructura que sigue el diseño OLAP, dentro de *New Cube* se asigna el nombre al cubo a crear y se relaciona con la tabla de hechos correspondiente. En *New Dimension* se da el nombre de la dimensión correspondiente a la tabla de hechos aunado con su ID. *New Hierarchy* se relaciona con su tabla de dimensión correspondiente y se da el nombre de la misma dimensión aunado al ID de su tabla. Dentro de *New Level* se crean los niveles de la jerarquía para cada dimensión, dando el nombre correspondiente aunado con el atributo correspondiente de la dimensión. Dentro de *New Measure* se da el nombre de la medida aunado al campo correspondiente de la tabla de hechos. Como se determinó anteriormente, el desarrollo OLAP se llevó a cabo mediante Pentaho.

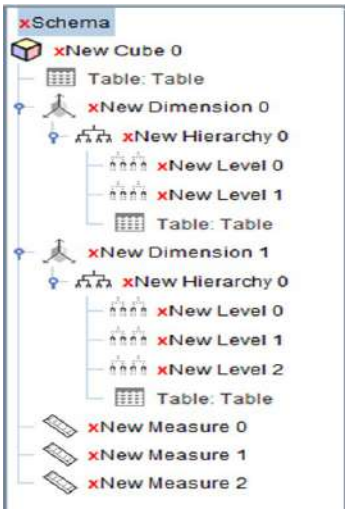


Figura 3.30. Diseño OLAP

3.4. Implementación del Data Warehouse

3.4.1. Desarrollo ETL

Se realizó una transformación para cada tabla de dimensión y tabla de hechos con la herramienta Pentaho Data Integration, en las Figuras 3.31, 3.32 y 3.33 se observan las transformaciones para las dimensiones lugar y tiempo y para la tabla de hechos visitantes internacionales.

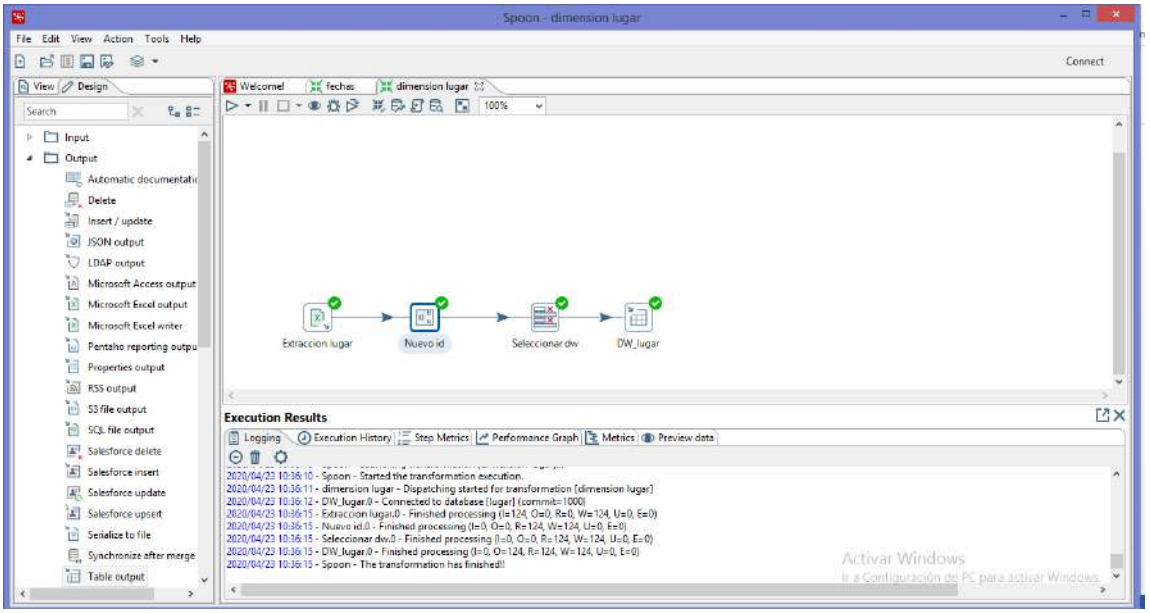


Figura 3.31. ETL para la dimensión Lugar

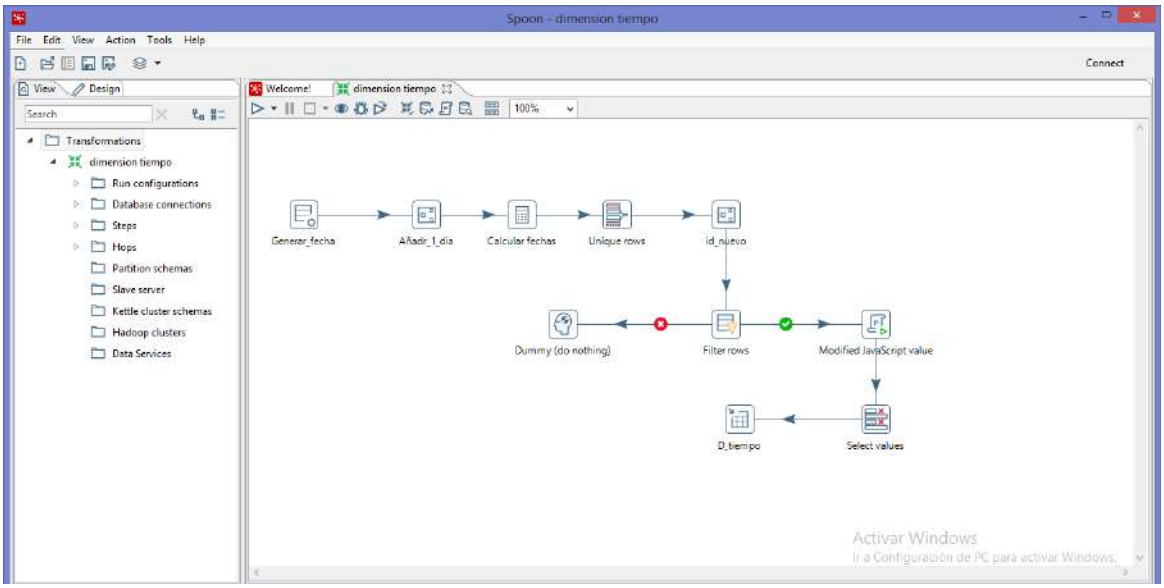


Figura 3.32. ETL para la dimensión Tiempo

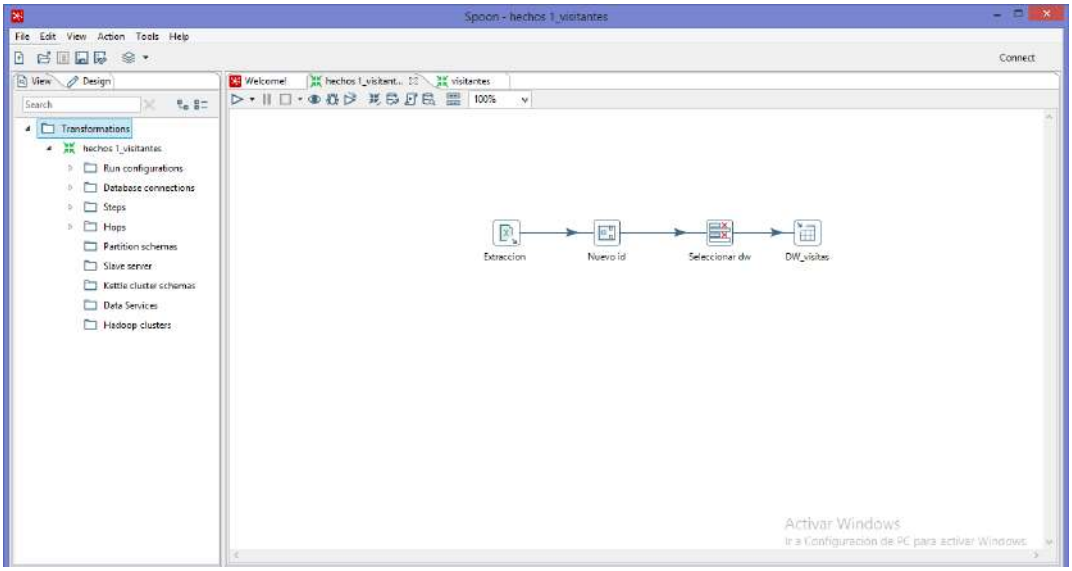


Figura 3.33. ETL para la tabla de hechos Visitantes internacionales

En las Figuras 3.34 a 3.36 se muestra la base de datos después de ejecutar las transformaciones anteriores.

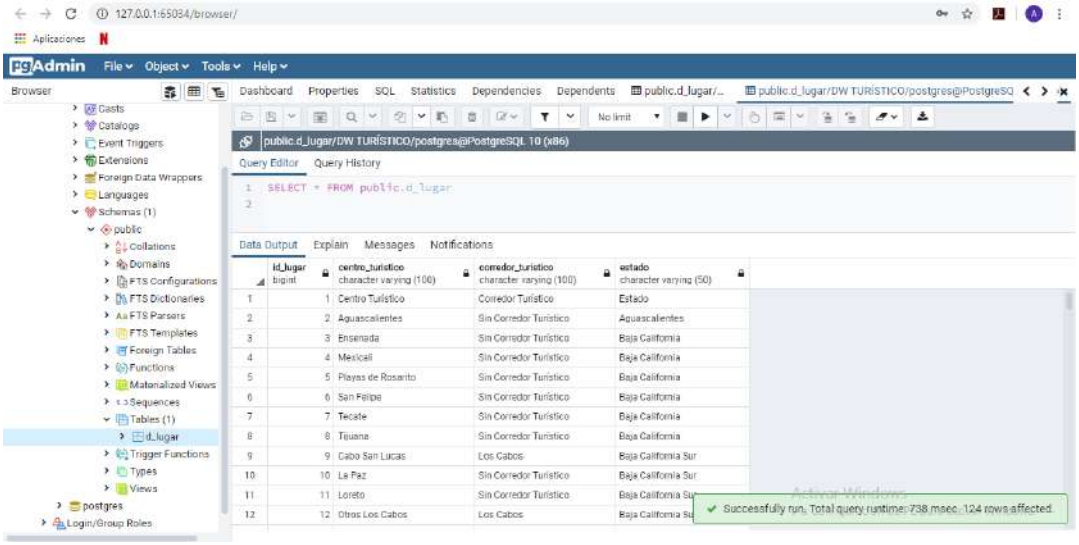


Figura 3.34. Base de datos después de la ejecución de la transformación Lugar

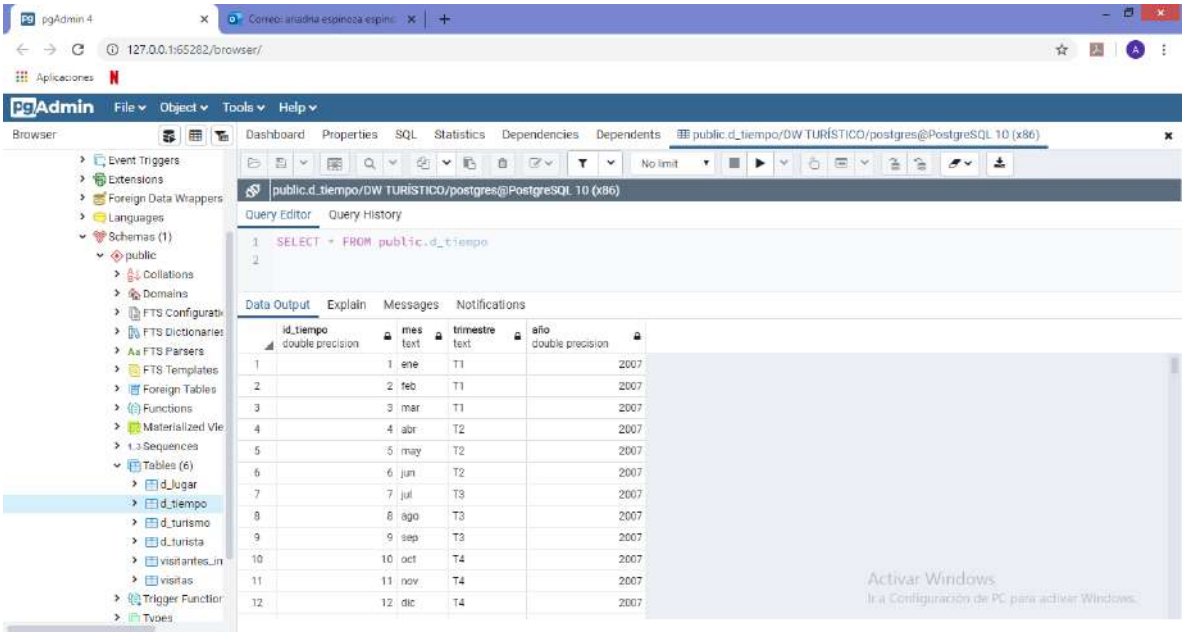


Figura 3.35. Base de datos después de ejecutar la transformación Tiempo

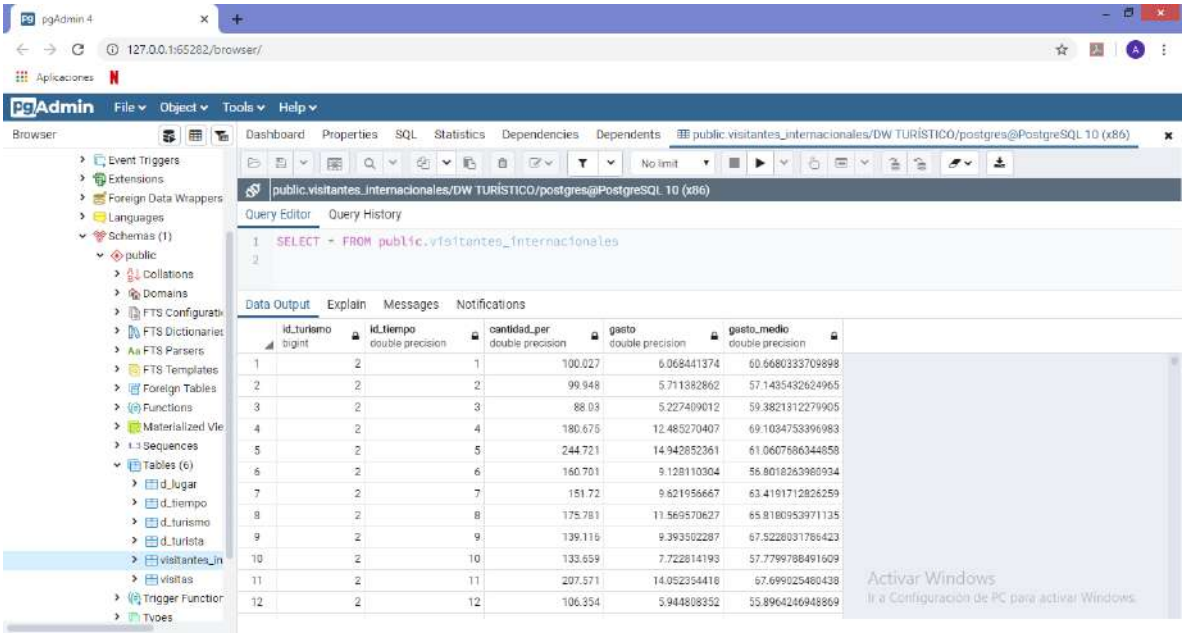


Figura 3.36. Base de datos después de ejecutar la transformación de Visitantes internacionales

3.4.2. Desarrollo aplicación BI

Se realizó la conexión con la base de datos *dw_turistico*. A través de la herramienta Schema Workbench de Pentaho como se visualiza en la Figura 3.37.

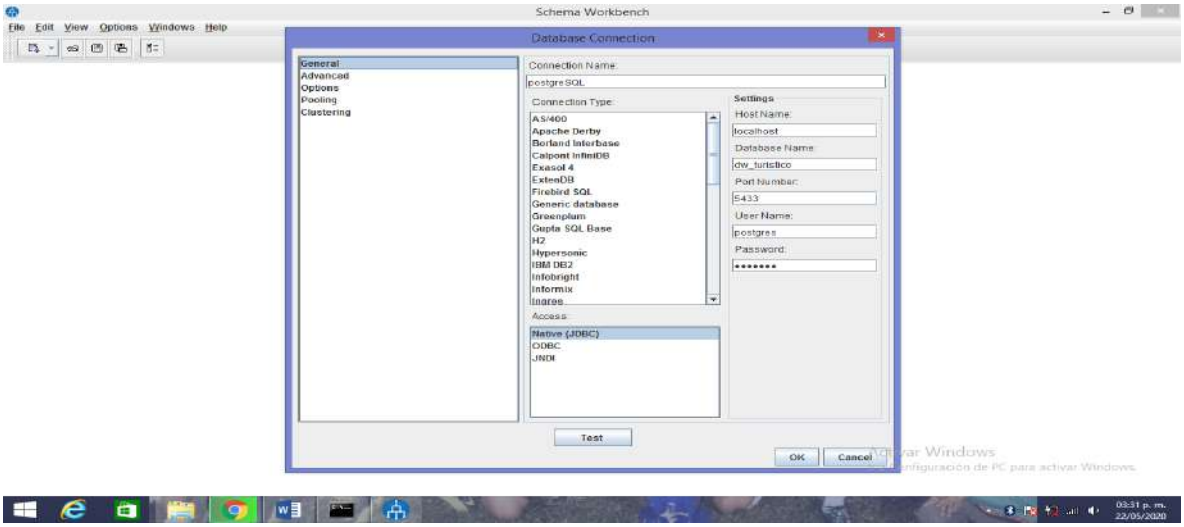


Figura 3.37. Conexión DB

A continuación, se crearon dos cubos OLAP, los cuales se muestran en las Figuras 3.38 y 3.39.

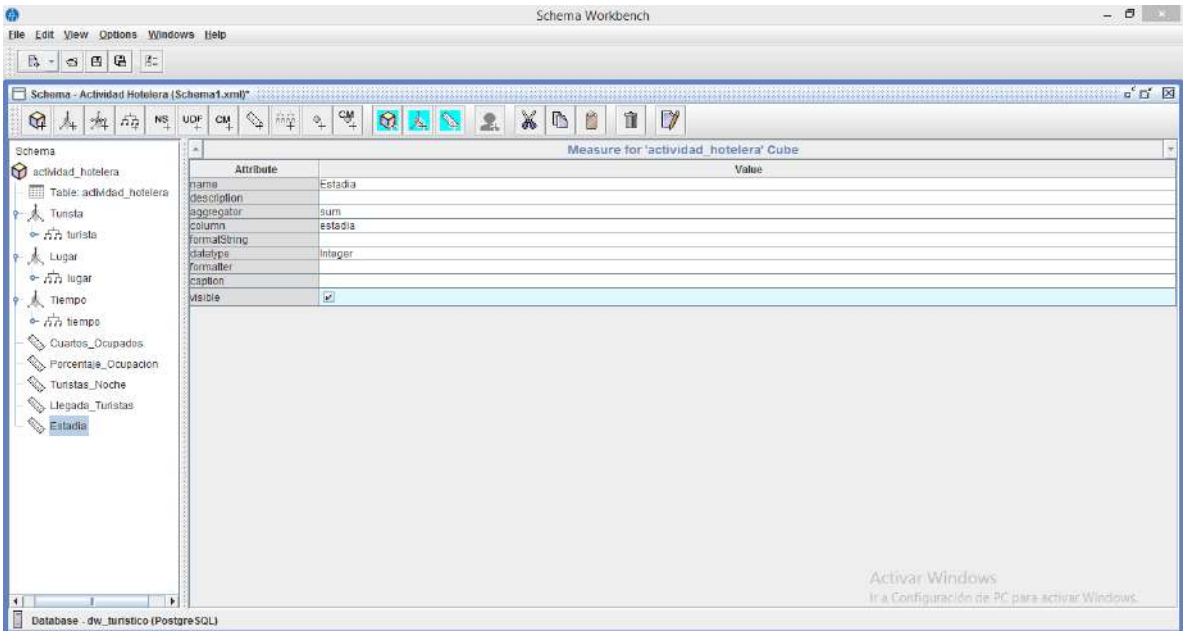


Figura 3.38. Cubo OLAP actividad_hotelera

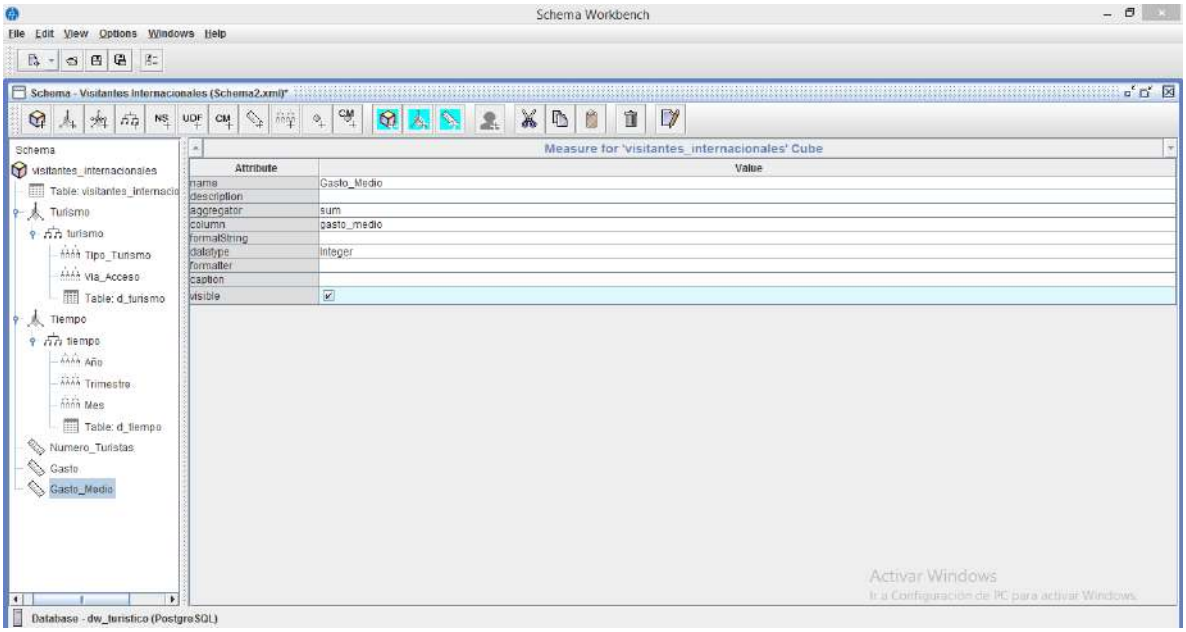


Figura 3.39. Cubo OLAP visitantes_internacionales

El siguiente paso fue importar los cubos al servidor Pentaho, para esto, se realizó la conexión a la base de datos y se importaron los cubos creados. En la Figura 3.40 se muestra la importación del cubo OLAP *actividad_hotelera*.

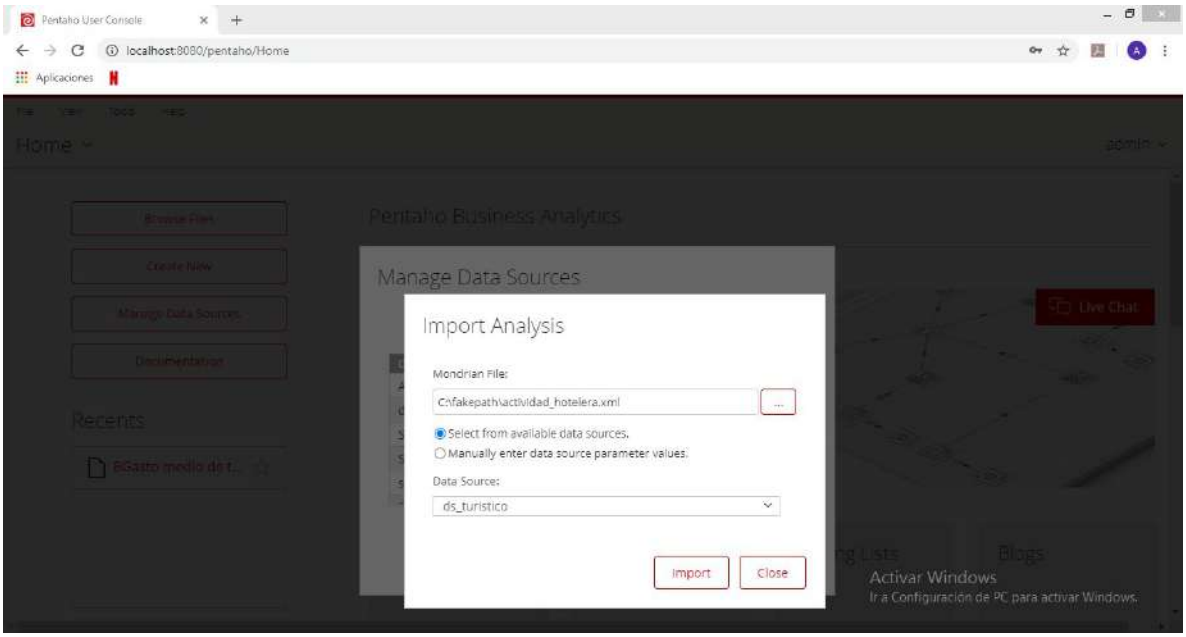


Figura 3.40. Importación del cubo OLAP actividad_hotelera

Lo mismo se realizó para el cubo de visitantes internacionales.

A continuación, se elaboraron distintos reportes utilizando cada uno de los cubos OLAP por medio de *JPivot*. En la Figura 3.41 se visualiza un reporte obtenido con el cubo

visitantes_internacionales, en este caso se elaboró el reporte Número de turistas internacionales por año/trimestre/mes.

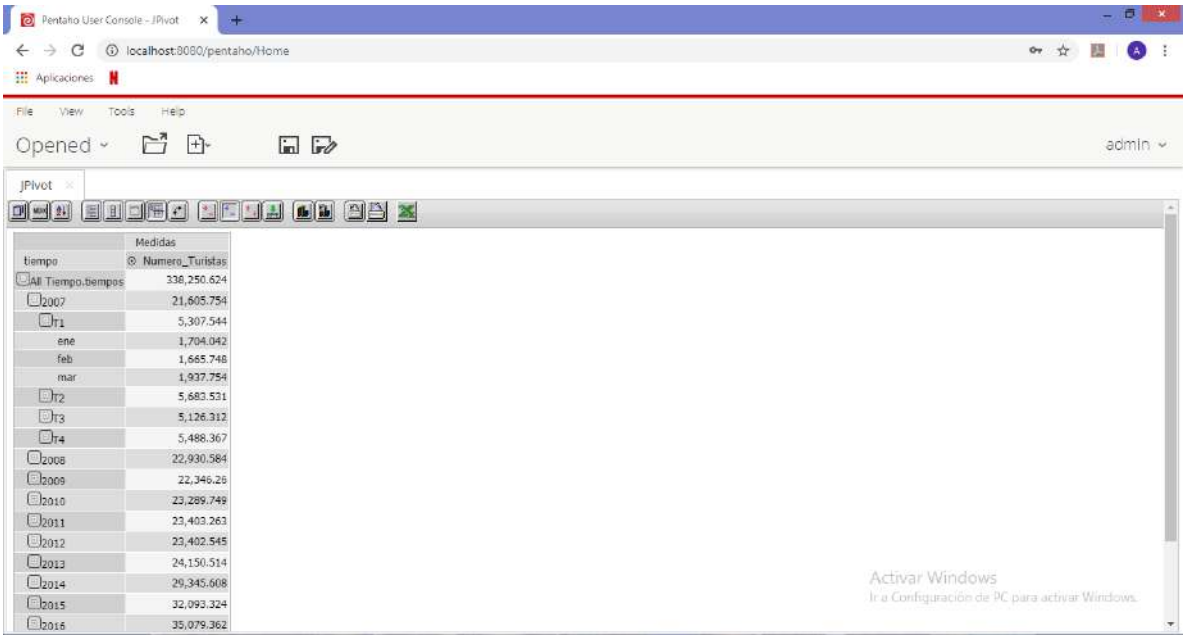


Figura 3.41. Vista de jPivot del cubo OLAP visitantes internacionales

Se realizó el mismo procedimiento para *Gasto total de turistas internacionales por año/trimestre/mes*, en este caso se agregó una gráfica, la cual se observa en la Figura 3.42.

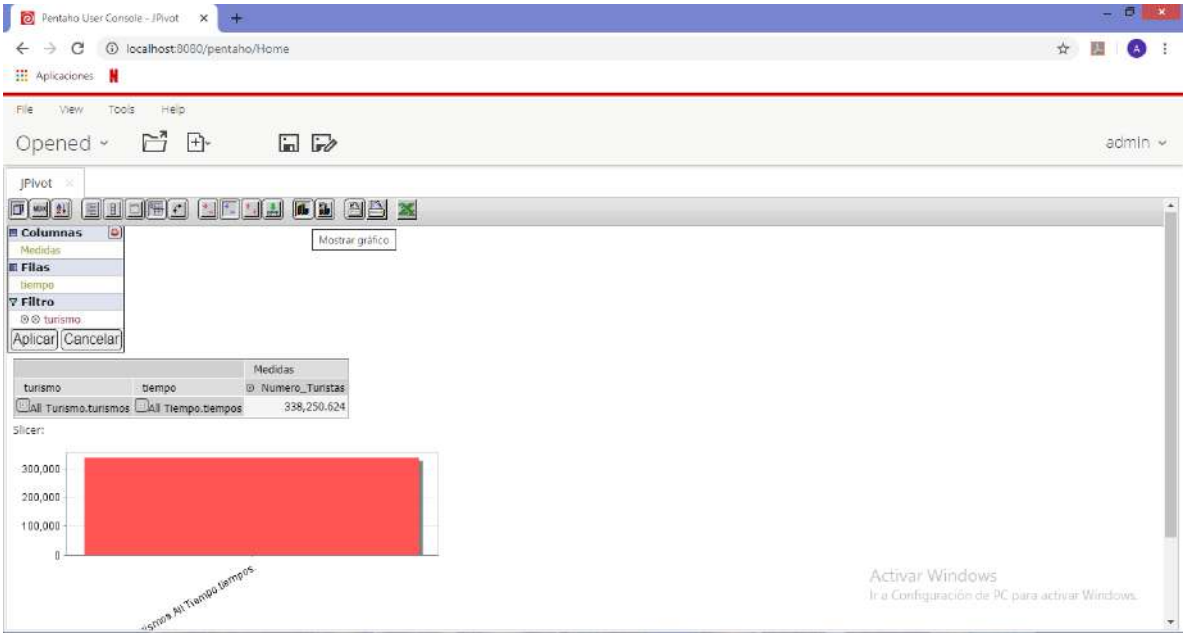


Figura 3.42. Gráfica del gasto total de turistas internacionales

3.5 Diseño del método de fragmentación horizontal basado en árboles de decisión para almacenes de datos

En la Figura 3.43 se muestra el método de fragmentación propuesto, el cual está compuesto por cuatro pasos.

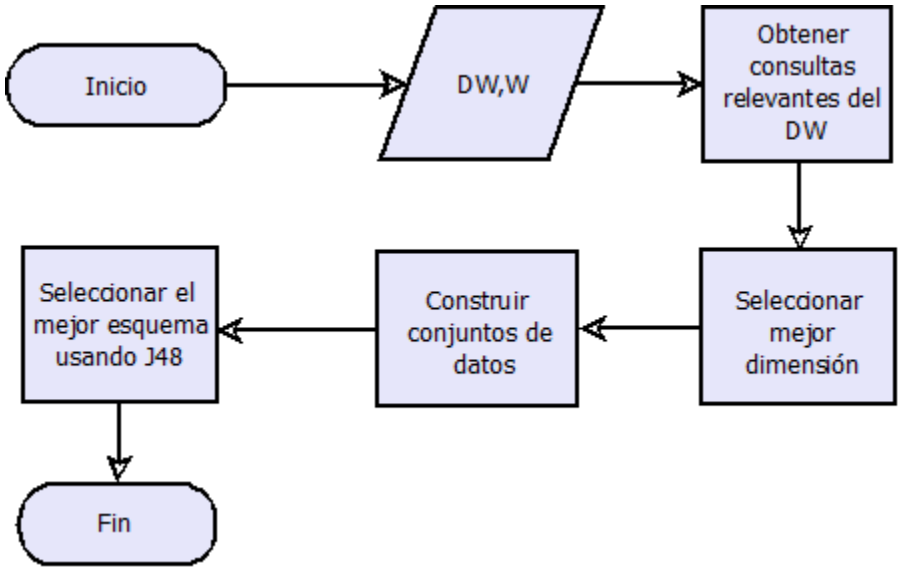


Figura 3.43. Método de fragmentación horizontal basado en árboles de decisiones

Con el fin de clarificar el enfoque propuesto, se presenta el siguiente escenario creado a partir de SSB (Star Schema Benchmark) (O'Neil et al., 2007). SSB contiene un data warehouse compuesto por una tabla de hechos **lineorder** y cuatro tablas de dimensiones **customer**, **supplier**, **part** y **date**. Se consideran las siguientes consultas:

q_1: select d_mes, sum (lo_quantity) from date, lineorder where date.d_datekey=lineorder.lo_orderdate and d_año=1992 group by d_mes;

q_2: select d_mes, sum (lo_quantity) from date, lineorder where date.d_datekey=lineorder.lo_orderdate and date.d_year=1992 and d_sellingseason='Spring' group by d_mes;

q_3: select d_year, sum (lo_quantity) from date, lineorder where date.d_datekey=lineorder.lo_orderdate and date.d_mes='January' group by d_year;

q_4: select d_year, sum (lo_quantity) from date, lineorder where date.d_datekey=lineorder.lo_orderdate and date.d_sellingseason='Winter' group by d_year;

q_5: select d_mes, sum (lo_quantity) from date, lineorder where date.d_datekey=lineorder.lo_orderdate and d_year=1993 group by d_mes;

El data warehouse (DW) está formado por una tabla de hechos F y varias tablas de dimensión $D = \{d_1, d_2, \ldots, d_m\}$. Para llevar a cabo la fragmentación se realizaron los siguientes pasos:

- **Obtener consultas relevantes del DW.** Para esto es necesario quedarse solo con las consultas realizadas en el DW que implican reuniones de la tabla de hechos con las tablas de dimensiones, cada tabla de dimensión d_i tendrá un conjunto de consultas $Q_i=\{q_{i1}, q_{i2}, \ldots, q_{in}\}$, también se obtiene la frecuencia de las consultas $F_i=\{f_{i1}, f_{i2}, \ldots, f_{in}\}$, cada consulta q_{ij} tiene un conjunto de predicados $Pr_{ij}=\{p_{ij1}, p_{ij2}, \ldots, p_{ijq}\}$ y por último, se obtiene la selectividad de los predicados $S_{ij}=\{sel_{ij1}, sel_{ij2}, \ldots, sel_{ijq}\}$. En el siguiente ejemplo, las cinco consultas corresponden a la dimensión **date**. La Tabla 3.10 muestra los predicados utilizados en las consultas.

Tabla 3.10. Predicados utilizados por las consultas.

Q	Pr
q_1	p_1=d_year=1992
q_2	p_1=d_year=1992, p_2=d_sellingseason='Spring'
q_3	p_3=d_sellingseason='Winter'
q_4	p_4=d_mes='January'
q_5	p_5=d_year=1993

- **Seleccionar mejor dimensión.** Se selecciona la dimensión con mayor importancia, para calcular la importancia de cada dimensión se utiliza la Ecuación 1, para esto, se realiza la sumatoria de la multiplicación de la frecuencia de la consulta por la sumatoria de la selectividad de sus predicados. A la tabla seleccionada se le llama *dim*.

$$I_i=\sum_{j=1}^{n} f_{ij}\left(\sum_{k=1}^{q} sel_{ijk}\right) \quad (1)$$

- **Construir conjuntos de datos.** Primero, se generan W esquemas de fragmentación con las consultas realizadas en *dim*, donde W se refiere al máximo número de fragmentos ingresado por el administrador del *data warehouse*, para esto se realizan los pasos mostrados en los Algoritmos 1 y 2. El Algoritmo 1 recibe como entrada una Matriz de Uso de Predicados (PUM, por sus siglas en inglés) de las consultas realizadas tanto a la tabla de hechos como a *dim* y W. La Tabla 3.11 muestra la PUM para el ejemplo.
  La salida del Algoritmo 1 son los conjuntos de datos. Primero se genera un árbol de partición (PT) (Son & Kim, 2004a), donde en el primer paso se considera que cada predicado está en un fragmento distinto (ver Figura 44); del paso 1 hasta el paso r-1, donde r es el número de predicados, se obtiene una Matriz de Beneficio de Fusión (MPM, por sus siglas en inglés), la cual mide el beneficio que se obtiene al unir un par de predicados. Cuando se une un par de predicados, la cantidad de tuplas remotas accedidas se reduce, ya que algunas consultas que antes requerían tuplas de

varios fragmentos, ahora solo tendrán que acceder un fragmento, por ejemplo, en la Figura 3.44, cuando se fusionan los predicados p_1 y p_2, q_2 solo tiene que acceder un fragmento, ya que utiliza ambos predicados; mientras que el número de tuplas irrelevantes recuperadas se incrementa porque algunas consultas ahora tendrán que recuperar todas las tuplas del nuevo fragmento aunque solo requieran las de uno de los fragmentos fusionados.

En la Figura 3.44, q_1 solo requiere las tuplas que satisfacen el predicado p_1, pero al fusionar los dos predicados, también tiene que acceder las tuplas del predicado p_2.

La MPM se obtiene con el Algoritmo 2 (L. Rodríguez-Mazahua et al., 2014), el cual toma como entrada la PUM y calcula la cantidad reducida de tuplas remotas accedidas (DRT) y la cantidad incrementada de tuplas irrelevantes accedidas (IIT). El beneficio de fusión de dos predicados es igual a la diferencia entre DRT e IIT. Se fusiona el par de predicados con el mayor beneficio de fusión.

La Tabla 3.12 muestra la MPM del ejemplo para el paso 1 del PT. Cuando se llega al paso W, se obtiene el primer conjunto de datos $data\_set_{W\text{-}c}$, el cual tiene el máximo número de fragmentos requerido por el administrador del DW, con cada fusión se genera un nuevo conjunto de datos con un fragmento menos, hasta encontrar el último conjunto generado formado por dos fragmentos.

Data: PUM of the fact table FT (a set of predicates $Pr=\{p_1, p_2,\ldots, p_r\}$, the selectivity sel_i of each predicate p_i, a set of queries $Q=\{q_1, q_2, \ldots, q_s\}$, the frequency f_k of each query q_k), W
Result: data sets $D=\{data\_set_2, data\_set_3, \ldots, data\_set_W\}$

```
getMPM(PUM, MPM);
c=0;
for each step_i ∈PT |1≤i≤r-1 do
        getMPM(PUM, MPM);
        select two nodes with maximum merging profit;
        merge the nodes;
        if i>=W
                generate data_set_W-c;
                c=c+1;
        end
end;
```

Algoritmo 1. Generar data sets.

Data: PUM
Result: MPM: Merging Profit Matrix

```
for each p_i ∈Pr | 1≤i≤r-1 do
        for each p_j ∈Pr | i+1≤j≤r do
                DRT=0;
                IIT=0;
                merging_profit=0;
                for each q_k ∈Q | 1≤k≤s do
                        if PUM(q_k,p_i)=1 & PUM(q_k,p_j)=1 then
                                DRT=DRT+f_k*(sel_i+sel_j);
```

```
                    else
                        if PUM(qk,pi)=1 then
                            IIT=IIT+ fk*selj
                        else
                            if PUM(qk,pJ)=1 then
                                IIT=IIT+ fk*selI
                            end
                        end
                    end
                end
                merging_profit=DRT-IIT;
                MPM(pi,pj)=merging_profit;
            end
        end
```

Algoritmo 2. Obtener MPM

Tabla 3.11 Matriz de uso de predicados

Q/Pr	p_1	p_2	p_3	p_4	p_5	f_i
q_1	1	0	0	0	0	5
q_2	1	1	0	0	0	4
q_3	0	0	1	0	0	3
q_4	0	0	0	1	0	10
q_5	0	0	0	0	1	7
sel_j	907987	524713	1577160	541483	908288	

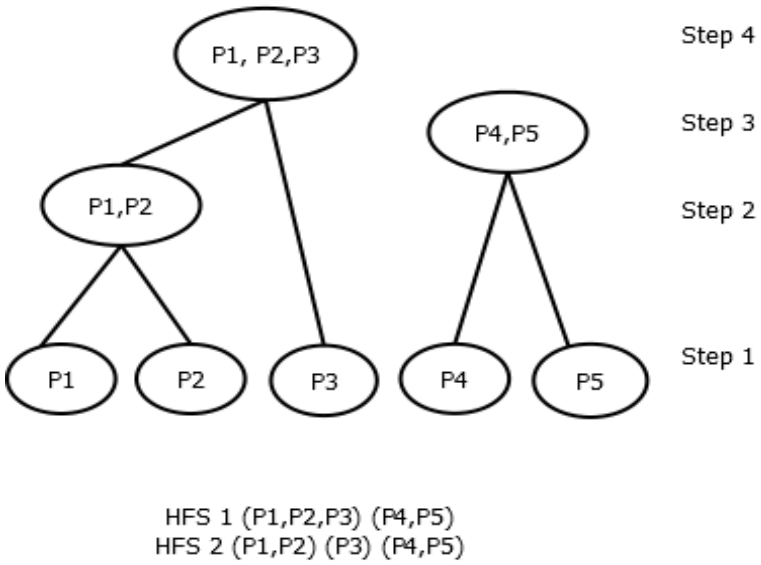


Figura 3.44. Árbol de Partición.

Tabla 3.12. Matriz de beneficio de fusión

Pr	p_1	p_2	p_3	p_4	p_5
p_1		3107235	-16918401	-13953217	-14530501
p_2			-7882779	-7413062	-7306143
p_3				-17396049	-13764984
p_4					-12873261
p_5					

- **Aplicar J48 a los conjuntos de datos.** Se utilizó la API de Weka para aplicar J48 a los W-1 conjuntos de datos. El conjunto de datos con el que se lograron los mejores resultados para las cuatro métricas de evaluación (*Precision*, *Recall*, ROC *area* y F-*measure*) fue el seleccionado. Debido a que se reduce considerablemente el desempeño del algoritmo a mayor número de fragmentos, es necesario mantener la cardinalidad del primer conjunto de datos para evitar que se obtengan mejores resultados cuando se tienen dos fragmentos, por lo tanto, se obtiene la cardinalidad de los fragmentos del primer conjunto de datos de datos y se generan consultas sintéticas en el conjunto de datos para mantener la cardinalidad del último fragmento, como se aprecia en la Figura 3.45

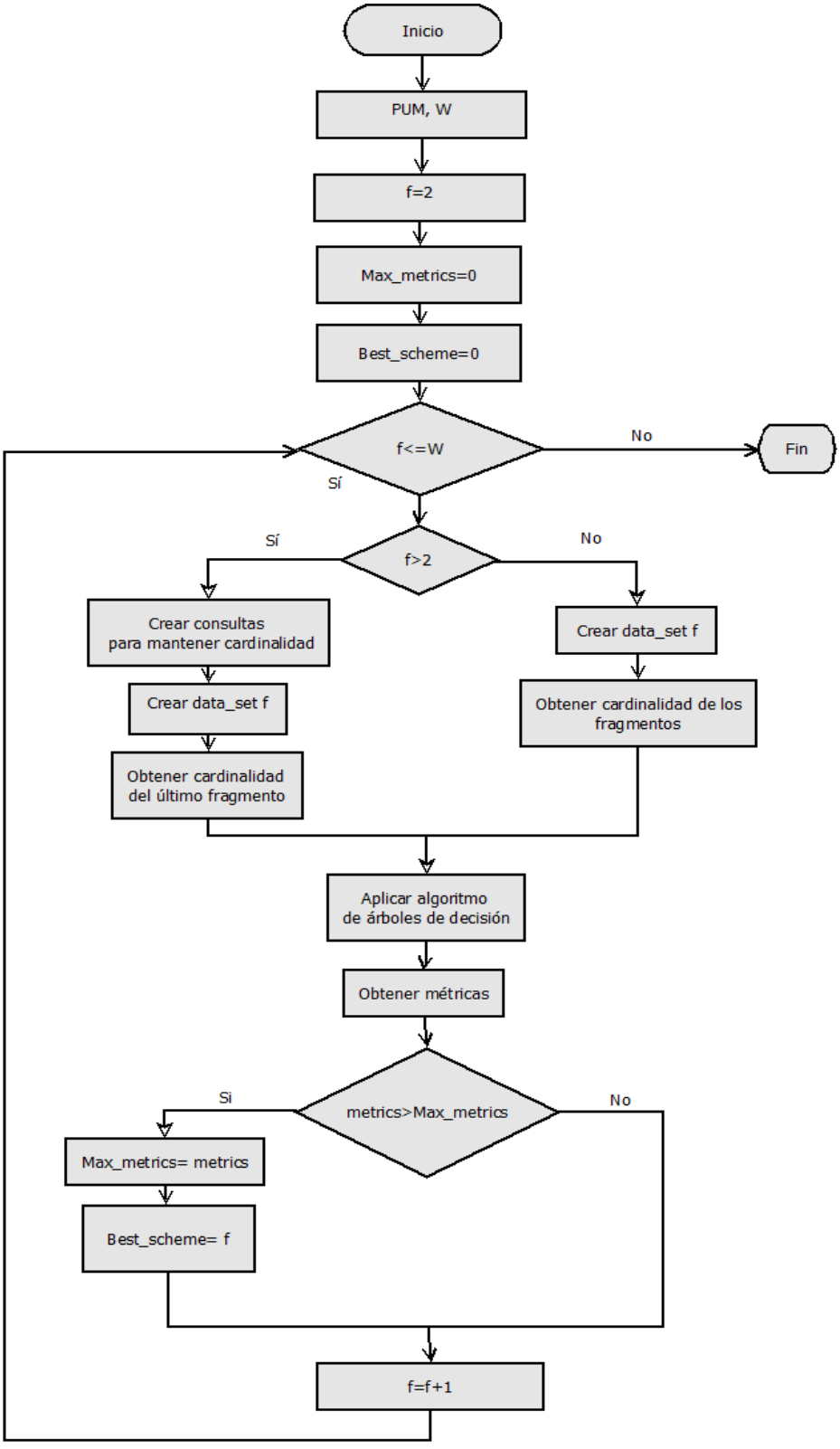


Figura 3.45. Método de fragmentación de la tabla de dimensión seleccionada.

3.6 Implementación del método de fragmentación

El método de fragmentación horizontal diseñado se implementó y probó en el DW turístico formado por dos tablas de hechos y cuatro de dimensiones, como se observa en la Tabla 3.13. Para esto se siguieron los pasos descritos a continuación.

Tabla 3.13. Tablas del DW turístico

Nombre	Tipo	Atributos	Tuplas
actividad_hotelera	Hechos	9	35424
visitantes_internacionales	Hechos	5	576
d_lugar	Dimensión	4	123
d_tiempo	Dimensión	4	144
d_turismo	Dimensión	3	4
d_turista	Dimensión	2	2

3.6.1 Obtener consultas relevantes del DW.

La tabla de hechos que se fragmentó fue *actividad_hotelera*. Se ejecutaron las consultas que se muestran en la Tabla 3.14:

Tabla 3.14. Consultas OLAP ejecutadas en el

Reporte	Consulta	Frecuencia
Número de cuartos totales por año/trimestre/mes.	SELECT mes, sum(cuartos_ocupados) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2007 GROUP BY mes;	3
	SELECT mes, sum(cuartos_ocupados) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2008 GROUP BY mes;	4
	SELECT mes, sum(cuartos_ocupados) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.trimestre='T1' GROUP BY mes;	2
Número de noches totales por año/trimestre/mes.	SELECT mes, sum(turistas_noche) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2009 GROUP BY mes;	5
	SELECT trimestre, sum(turistas_noche) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2010 GROUP BY trimestre;	2
	SELECT anio, sum(turistas_noche) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.trimestre='T2' group by anio;	3
Número de llegadas totales por año/trimestre/mes.	SELECT mes, sum(llegada_turistas) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND	4

Reporte	Consulta	Frecuencia
	d_tiempo.anio=2011 group by mes;	
	SELECT anio, sum(llegada_turistas) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.mes='dic' group by anio;	7
	SELECT trimestre, sum(llegada_turistas) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2012 group by trimestre;	4
Estadía total por año/trimestre/mes.	SELECT anio, avg(estadia) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.mes='jul' group by anio;	10
	SELECT trimestre, avg(estadia) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2013 group by trimestre;	3
	SELECT mes, avg(estadia) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2013 AND d_tiempo.trimestre='T4' group by mes;	5

Para iniciar el método de fragmentación es necesario ejecutar el script *FTree.sql* en el DW turístico. *FTree.sql* crea el esquema *ftree* en el *data warehouse*, así como cinco funciones, cinco tablas, nueve disparadores y una vista dentro del esquema *ftree*. La descripción de cada uno de estos se presenta en la Tabla 3.14.

Tabla 3.14. Contenido del esquema *ftree* creado por *FTree.sql*.

Nombre	Tipo	Función
postgres_log	Tabla	Almacena todas las consultas que se encuentran en el archivo log.
current_database	Disparador	Antes de insertar en la tabla *postgres_log* verifica que solo se guarden las consultas que se realizaron al DW turístico.
copy_log()	Función	Copia el contenido del log de transacciones del día actual en la tabla *postgres_log*.
queries	Tabla	Almacena el identificador, la descripción y la frecuencia de las consultas.
delete_queries	Disparador	Después de eliminar una consulta en la tabla *queries*, reduce el valor del identificador en 1.
insert_queries	Disparador	Antes de insertar una consulta en la tabla *queries*, si la consulta ya está almacenada en esta tabla, entonces no se inserta y su frecuencia se incrementa en 1.
ai_queries	Disparador	Después de insertar una consulta en la tabla *queries,*

Nombre	Tipo	Función
		actualiza su identificador si es mayor que el número máximo de filas.
statistic_collector(text)	Función	Analiza las consultas almacenadas en *postgres_log* para insertar en la tabla *queries* solo las ejecutadas en la tabla que se desea fragmentar.
attributes	Vista	Almacena los nombres de las dimensiones, así como de los atributos de cada dimensión y el orden en que aparecen en la tabla de dimensión.
attribute	Tabla	Guarda los datos de la vista *attributes* y agrega un atributo de orden global.
fact_attributes	Vista	Almacena el nombre de la tabla de hechos además de los nombres y el orden de cada uno de sus atributos.
predicates	Tabla	Guarda el identificador, la descripción y la selectividad de los predicados.
insert_predicates	Disparador	Antes de insertar en la tabla *predicates*, analiza que solo se tomen los predicados de las consultas que involucren a la tabla de hechos con la tabla de dimensión seleccionada.
insert_predicates2	Disparador	Antes de insertar en la tabla *predicates*, valida que solo se inserten predicados diferentes.
ai_predicates	Disparador	Después de insertar un predicado en la tabla *predicates*, actualiza su identificador si es mayor que el número máximo de filas.
delete_predicates	Disparador	Después de eliminar un predicado en la tabla *predicates*, reduce el valor del identificador en 1.
get_predicates()	Función	Obtiene los predicados de las consultas almacenadas en *queries* y los guarda en *predicates*.
query_predicate	Tabla	Guarda la relación entre las consultas y los predicados.
delete_query_predicate	Disparador	Después de eliminar en la tabla *query_predicate*, reduce el valor de los identificadores de la consulta y del predicado en 1.
query_pred()	Función	Analiza qué predicados aparecen en qué consultas y guarda esta relación en *query_predicate*.
get_dataset()	Función	Crea los conjuntos de datos que se utilizan para entrenar el árbol de decisión.

Posteriormente, *FTree.sql* ejecuta la función *copy_log()*. El contenido de la tabla *postgres_log* se muestra en la Figura 3.46.

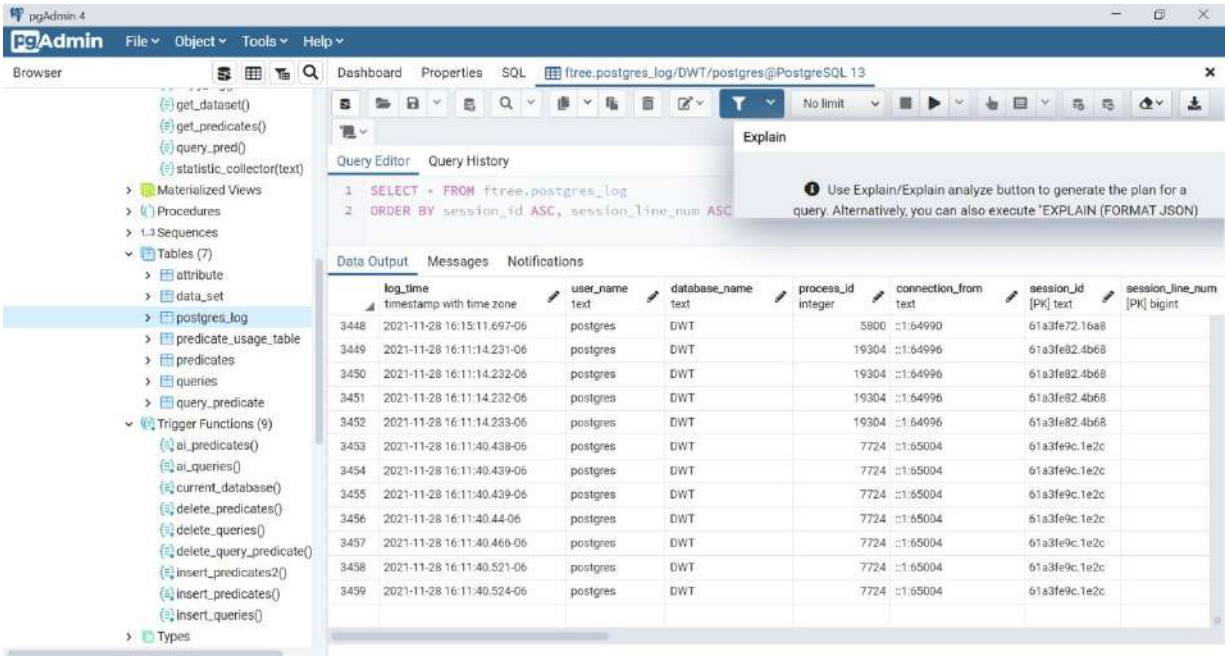


Figura 3.46. Ejecución de función copy_log() en el DW turístico.

A continuación, el ADW (Administrador del Data Warehouse) ejecuta la función *statistic_collector('actividad_hotelera')*, el contenido de la tabla *queries* se presenta en la Figura 3.47.

	id [PK] integer	description text	frequency integer
1	1	SELECT mes, sum(cuartos_ocupados) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2007 GROUP BY mes;	3
2	2	SELECT mes, sum(cuartos_ocupados) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2008 GROUP BY mes;	4
3	3	SELECT mes, sum(cuartos_ocupados) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.trimestre='T1' GROUP BY mes;	2
4	4	SELECT mes, sum(turistas_noche) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2009 GROUP BY mes;	5
5	5	SELECT trimestre, sum(turistas_noche) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2010 GROUP BY trimestre;	2
6	6	SELECT anio, sum(turistas_noche) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.trimestre='T2' group by anio;	3
7	7	SELECT mes, sum(llegada_turistas) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2011 group by mes;	4
8	8	SELECT anio, sum(llegada_turistas) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.mes='dic' group by anio;	7
9	9	SELECT trimestre, sum(llegada_turistas) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio='2012' group by trimestre;	4
10	10	SELECT anio, avg(estadia) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.mes='jul' group by anio;	10
11	11	SELECT trimestre, avg(estadia) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2013 group by trimestre;	3
12	12	SELECT mes, avg(estadia) FROM d_tiempo, actividad_hotelera WHERE d_tiempo.id_tiempo=actividad_hotelera.id_tiempo AND d_tiempo.anio=2013 AND d_tiempo.trimestre='T4' group by mes;	4

Figura 3.47. Tabla queries.

Posteriormente, se ejecuta la función *query_pred()* que obtiene los predicados de las consultas. El contenido de la tabla *predicates* se muestra en la Figura 3.48.

	id_pred [PK] integer	description_pred text	selectividad integer
1	1	anio=2007	2952
2	2	anio=2008	2952
3	3	trimestre='T1'	8856
4	4	anio=2009	2952
5	5	anio=2010	2952
6	6	trimestre='T2'	8856
7	7	anio=2011	2952
8	8	mes='dic'	2952
9	9	anio=2012	2952
10	10	mes='jul'	2952
11	11	anio=2013	2952
12	12	trimestre='T4'	8856

Figura 3.48. Tabla predicates

A continuación, se ejecuta la función *query_pred()* que permite relacionar a las consultas con sus predicados. El contenido de la tabla *query_predicate* se presenta en la Figura 3.49.

	id_query [PK] integer	id_pred [PK] integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	10	10
10	11	11
11	12	11
12	12	12

Figura 3.49. Tabla query_predicate

3.6.2 Construir conjuntos de datos

Al ejecutar la función *get_dataset*(), se crean dos tablas:

predicate_usage_table: Almacena el identificador de la consulta, los predicados que utiliza y su frecuencia.

data_set: Guarda los datos necesarios para entrenar el árbol de decisión (consultas, predicados, frecuencia de consultas y fragmento al que pertenecen las consultas).

Estas tablas se muestran en las Figuras 3.50 y 3.51.

	query [PK] integer	p1 bit	p2 bit	p3 bit	p4 bit	p5 bit	p6 bit	p7 bit	p8 bit	p9 bit	p10 bit	p11 bit	p12 bit	frequency integer
1	1	1	0	0	0	0	0	0	0	0	0	0	0	3
2	2	0	1	0	0	0	0	0	0	0	0	0	0	4
3	3	0	0	1	0	0	0	0	0	0	0	0	0	2
4	4	0	0	0	1	0	0	0	0	0	0	0	0	5
5	5	0	0	0	0	1	0	0	0	0	0	0	0	2
6	6	0	0	0	0	0	1	0	0	0	0	0	0	3
7	7	0	0	0	0	0	0	1	0	0	0	0	0	4
8	8	0	0	0	0	0	0	0	1	0	0	0	0	7
9	9	0	0	0	0	0	0	0	0	0	0	0	0	4
10	10	0	0	0	0	0	0	0	0	0	1	0	0	10
11	11	0	0	0	0	0	0	0	0	0	0	1	0	3
12	12	0	0	0	0	0	0	0	0	0	0	1	1	4

Figura 3.50. Tabla predicate_usage_table

	query [PK] integer	mes character varying (50)	att_mes bit	trimestre character varying (50)	att_trimestre bit	anio character varying (50)	att_anio bit	frequency integer	fragment character varying (10)
1	1	NOT_USED	0	NOT_USED	0	2007	1	3	F1
2	2	NOT_USED	0	NOT_USED	0	2008	1	4	F1
3	3	NOT_USED	0	T1	1	NOT_USED	0	2	F1
4	4	NOT_USED	0	NOT_USED	0	2009	1	5	F1
5	5	NOT_USED	0	NOT_USED	0	2010	1	2	F1
6	6	NOT_USED	0	T2	1	NOT_USED	0	3	F1
7	7	NOT_USED	0	NOT_USED	0	2011	1	4	F1
8	8	dic	1	NOT_USED	0	NOT_USED	0	7	F1
9	9	NOT_USED	0	NOT_USED	0	NOT_USED	0	4	F1
10	10	jul	1	NOT_USED	0	NOT_USED	0	10	F1
11	11	NOT_USED	0	NOT_USED	0	2013	1	3	F1
12	12	NOT_USED	0	T4	1	2013	1	4	F1

Figura 3.51. Conjunto de datos con un solo fragmento

A partir de *predicate_usage_table* se ejecutó el algoritmo MHPA (*Multimedia Horizontal Partititoning Algorithm*) propuesto por Rodríguez-Mazahua et al. (2014) para obtener los

W-2 esquemas de fragmentación, donde *W* es el número máximo de fragmentos ingresado por el ADW, en este caso *W*=4. Por lo tanto, es necesario obtener dos esquemas. Los esquemas que se utilizaron para entrenar el árbol de decisión fueron los del paso 9 y 10 del árbol de partición mostrado en la Figura 3.52.

Partition tree

Step0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	[047232]
Step1	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11,p12	*	[26568/0]
Step2	p1	p2	p3,p9	p4	p5	p6	p7	p8	*	p10	p11,p12	*	[32472/0]
Step3	p1,p5	p2	p3,p9	p4	*	p6	p7	p8	*	p10	p11,p12	*	[47232/0]
Step4	p1,p5	p2,p7	p3,p9	p4	*	p6	*	p8	*	p10	p11,p12	*	[70848/0]
Step5	p1,p5	p2,p7	p3,p9	p4,p8	*	p6	*	*	*	p10	p11,p12	*	[106272/0]
Step6	p1,p5	p2,p7	p3,p9,p6	p4,p8	*	*	*	*	*	p10	p11,p12	*	[159408/0]
Step7	p1,p5,p10	p2,p7	p3,p9,p6	p4,p8	*	*	*	*	*	*	p11,p12	*	[233208/0]
Step8	p1,p5,p10	p2,p7,p4,p8	p3,p9,p6	*	*	*	*	*	*	*	p11,p12	*	[351288/0]
Step9	p1,p5,p10	p2,p7,p4,p8	p3,p9,p6,p11,p12	*	*	*	*	*	*	*	*	*	[554976/0]
Step10	p5,p10,p2,p7,p4,p8	*	p3,p9,p6,p11,p12	*	*	*	*	*	*	*	*	*	[909216/0]
Step11	3,p3,p9,p6,p11,p12	*	*	*	*	*	*	*	*	*	*	*	[2293704/0]

Figura 3.52. Árbol de partición obtenido por el algoritmo MHPA

Finalmente, se utilizó el algoritmo J48 para obtener el esquema de fragmentación horizontal óptimo. Las Figuras 3.53 y 3.54 muestran los árboles de decisión generados por J48 para el conjunto de datos del paso 9 y 10, respectivamente, y la Tabla 3.15 los resultados obtenidos por Weka 3.9.

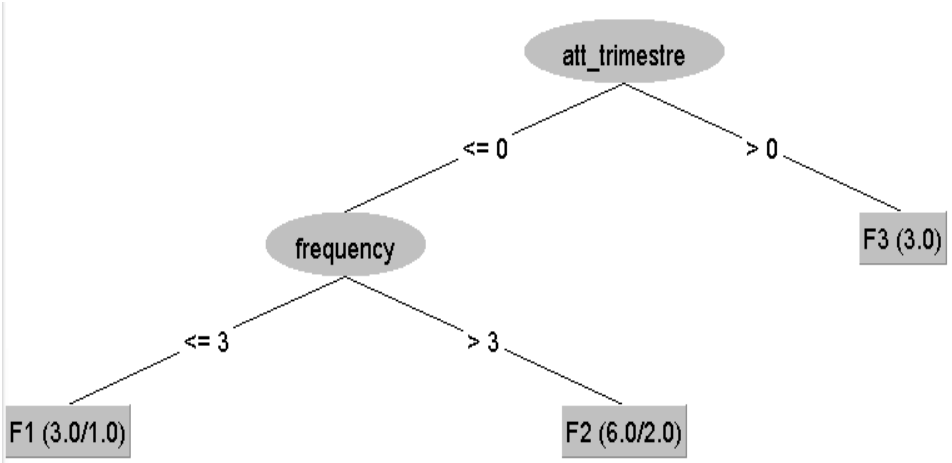


Figura 3.53. Árbol de decisión obtenido por J48 para 3 fragmentos

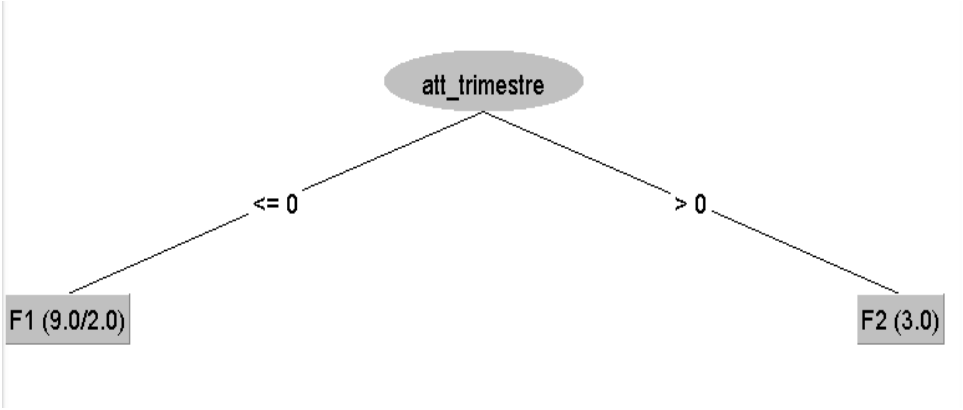


Figura 3.54 . Árbol de decisión producido por J48 para 2 fragmentos

Tabla 3.15. Comparación de los esquemas de fragmentación con 2 y 3 fragmentos.

Fragmentos	Precisión	Exhaustividad	Medida F	Área ROC
2	0.788	0.667	0.593	0.729
3	-	0.417	-	0.533

El mejor esquema de fragmentación obtenido por FTree es el de dos fragmentos fr_1= {p_1, p_2, p_4, p_5, p_7, p_8, p_{10}}, fr_2={p_3, p_6, p_9, p_{11}, p_{12}}.

3.7 Prueba del método de fragmentación que usa árboles de decisión en el DW desarrollado

Para evaluar el método de fragmentación desarrollado, llamado FTree, se comparó con (Kechar & Nait-Bahloul, 2017), para esto se utilizó la matriz de uso de predicados (MUP) de la Figura 3.55.

Q_i/P_j	P_1	P_2	P_3	P_4	P_5	P_6	P_7	F_i
Q_1	0	1	0	0	1	0	1	20
Q_2	1	1	0	0	0	1	0	15
Q_3	0	0	1	0	1	0	1	20
Q_4	0	1	0	1	0	1	0	15
Q_5	0	0	1	0	1	0	1	15
Q_6	1	0	0	1	0	1	1	10
Q_7	0	1	0	1	0	1	0	15
s_j	30	50	80	65	20	35	40	

Figura 3.55. Matriz de uso de predicados

El mejor esquema de fragmentación de acuerdo con (Kechar & Nait-Bahloul, 2017) es el de tres fragmentos $fr_1=\{P_1,P_2,P_3,P_4,P_5\}$, $fr_2=\{P_6\}$, $fr_3=\{P_7\}$.

Con FTree, en este caso, se considera que el número máximo de fragmentos permitido por el administrador del almacén de datos (W) es 4, por lo tanto, ingresando la matriz de la Figura 3.55 se obtienen el árbol de partición de la Figura 3.56.

Se evaluaron los pasos 4 y 5 en Weka para determinar el mejor esquema de fragmentación.

Los predicados considerados se presentan en la Figura 3.57. Los conjuntos de datos con dos y tres fragmentos se muestran en las Figuras 3.58 y 3.59.

Partition tree

Step0	p1	p2	p3	p4	p5	p6	p7	[0/124875]
Step1	p1	p2	p3	p4	p5,p7	p6	'	[200/120675]
Step2	p1	p2	p3	p4,p6	p5,p7	'	'	[1175/101625]
Step3	p1	p2,p4,p6	p3	'	p5,p7	'	'	[3675/71250]
Step4	p1,p2,p4,p6	'	p3	'	p5,p7	'	'	[5175/61500]
Step5	p1,p2,p4,p6	'	p3,p5,p7	'	'	'	'	[7575/24000]
Step6	p1,p2,p4,p6,p3,p5,p7	'	'	'	'	'	'	[20175/0]

Figura 3.56. Árbol de partición generado por FTree

	id_pred [PK] integer	description_pred text	selectividad integer
1	1	anio=2007	30
2	2	mes='dic'	50
3	3	anio=2012	80
4	4	anio=2008	65
5	5	trimestre='T1'	20
6	6	anio=2009	35
7	7	anio=2010	40

Figura 3.57. Predicados

	query [PK] integer	mes character varying (50)	att_mes bit	trimestre character varying (50)	att_trimestre bit	anio character varying (50)	att_anio bit	frequency integer	fragment character varying (10)
1	1	dic	1	T1	1	2010	1	20	F2
2	2	dic	1	NOT_USED	0	2007	1	15	F1
3	3	NOT_USED	0	T1	1	2012	1	20	F2
4	4	dic	1	NOT_USED	0	2008	1	15	F1
5	5	NOT_USED	0	T1	1	2012	1	15	F2
6	6	NOT_USED	0	NOT_USED	0	2007	1	10	F1
7	7	dic	1	NOT_USED	0	2008	1	15	F1
8	8	dic	1	NOT_USED	0	2009	1	15	F1
9	9	NOT USED	0	T1	1	2010	1	20	F2
10	10	dic	1	NOT USED	0	2009	1	15	F1
11	11	NOT USED	0	T1	1	2010	1	15	F2
12	12	NOT USED	0	NOT USED	0	2008	1	10	F1
13	13	NOT USED	0	NOT USED	0	2009	1	10	F1
14	14	NOT USED	0	NOT USED	0	2010	1	10	F1
15	15	dic	1	NOT USED	0	2009	1	15	F1

Figura 3.58. Conjunto de datos con dos fragmentos.

	query [PK] integer	mes character varying (50)	att_mes bit	trimestre character varying (50)	att_trimestre bit	anio character varying (50)	att_anio bit	frequency integer	fragment character varying (10)
1	1	dic	1	T1	1	2010	1	20	F3
2	2	dic	1	NOT_USED	0	2007	1	15	F1
3	3	NOT_USED	0	T1	1	2012	1	20	F2
4	4	dic	1	NOT_USED	0	2008	1	15	F1
5	5	NOT_USED	0	T1	1	2012	1	15	F2
6	6	NOT_USED	0	NOT_USED	0	2007	1	10	F1
7	7	dic	1	NOT_USED	0	2008	1	15	F1
8	8	dic	1	NOT_USED	0	2009	1	15	F1
9	9	NOT USED	0	T1	1	2010	1	20	F2
10	10	dic	1	NOT USED	0	2009	1	15	F1
11	11	NOT USED	0	T1	1	2010	1	15	F2
12	12	NOT USED	0	NOT USED	0	2008	1	10	F1
13	13	NOT USED	0	NOT USED	0	2009	1	10	F1
14	14	NOT USED	0	NOT USED	0	2010	1	10	F1
15	15	dic	1	NOT USED	0	2009	1	15	F1

Figura 3.59. Conjunto de datos con tres fragmentos.

La etiqueta de clase $fr(Q_i)$ se asignó de acuerdo con la selectividad, una consulta pertenece al fragmento que tenga a los predicados con la mayor selectividad que utiliza, por ejemplo, de acuerdo con la Figura 3.55, la consulta Q_1 utiliza los predicados P_2, P_5 y P_7, la cardinalidad de estos predicados es 50, 20 y 40, respectivamente. De acuerdo con la Figura 3.56, en el esquema de dos fragmentos, P_2 se encuentra en el primer fragmento (*F1*), mientras que P_5 y P_7 están en el segundo fragmento (*F2*), por lo que se asigna la etiqueta *F1* a Q_1, ya que la selectividad de $s_2<s_5+s_7$.

Los árboles de decisión obtenidos con J48 con ambos conjuntos de datos se visualizan en las Figuras 3.60 y 3.61.

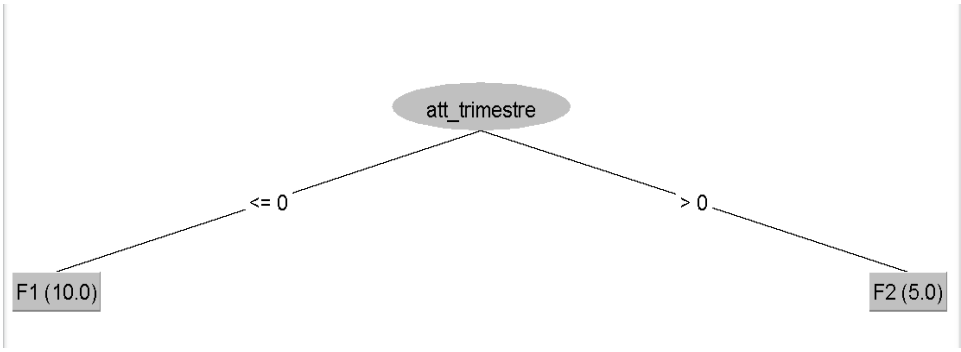


Figura 3.60. Árbol obtenido por J48 con dos fragmentos.

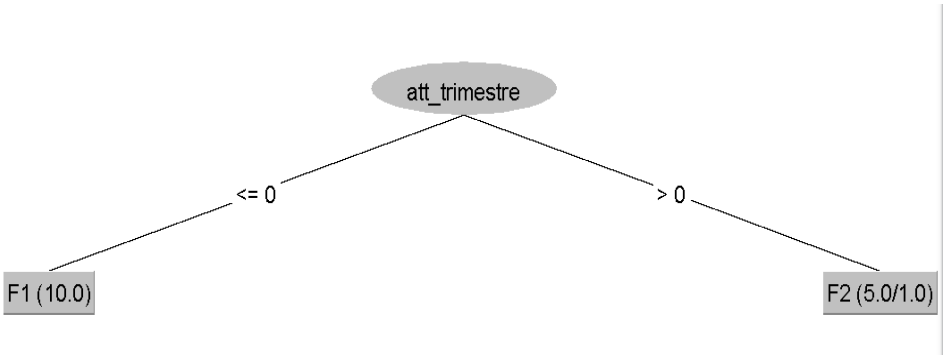


Figura 3.61. Árbol obtenido por J48 con tres fragmentos.

La Tabla 3.16 muestra la comparación entre los dos árboles de decisión.

Tabla 3.16. Resultados de las métricas de evaluación para los dos esquemas.

Fragmentos	Precisión	Exhaustividad	Medida F	Área ROC
2	1	1	1	1
3	-	0.933	-	0.933

El mejor esquema es por lo tanto el de dos fragmentos.

Para comparar los dos esquemas se utilizó un modelo de costos que considera que el costo de una consulta OLAP Q_i está dado por la cantidad de datos irrelevantes accedidos (TIC) y la cantidad de datos relevantes ubicados en otros fragmentos (TRC).

$$Costo(Q_i) = TIC(Q_i) + TRC(Q_i)$$

La cantidad de datos irrelevantes accedidos de una consulta se calcula multiplicando el número de tuplas irrelevantes accedidos por la consulta en cada fragmento fr_k por la frecuencia de la consulta.

$$TIC(Q_i) = \sum_{k=1}^{m} (card(fr_k) - \sum_{P_j | MUP(Q_i,P_j)=1 \wedge P_j \in fr_k} s_j) * F_i$$

La cantidad de datos relevantes ubicados en otros fragmentos se obtiene multiplicando el número de tuplas relevantes que están en otros fragmentos multiplicados por su frecuencia al cuadrado.

$$TRC(Q_i) = \sum_{P_j || MUP(Q_i,P_j)=1 \wedge P_j \notin fr(Q_i)}^{n} s_j * F_i^2$$

La Tabla 3.17 muestra el costo de las consultas en los dos esquemas de fragmentación: el obtenido por (Kechar & Nait-Bahloul, 2017) y el de FTree. En el caso del esquema generado por FTree, las cardinalidades del primer y segundo fragmento son card (fr_1)=168, card (fr_2)=128. El TIC de Q_1 se calculó de la siguiente forma:

$$\begin{aligned} TIC(Q_1) &= \big(card(fr_1) - s_2 + card(fr_2) - (s_5 + s_7)\big) * F_1 \\ &= \big(168 - 50 + 128 - (20 + 40)\big) * 20 = 3720 \end{aligned}$$

El TRC de la misma consulta se obtuvo como se muestra a continuación:

$$TRC(Q_1) = s_2 * F_1^2 = 50 * 20^2 = 20000.$$

Para el esquema producido por (Kechar & Nait-Bahloul, 2017), las cardinalidades de los fragmentos son card(fr_1)=228, card(fr_2)=32, card(fr_3)=36. Por lo tanto, los costos de la primera consulta en este esquema son:

$$TIC(Q_1) = \big(card(fr_1) - (s_2 + s_5)\big) * F_1 = (228 - 70) * 20 = 3160$$

$$TRC(Q_1) = s_7 * F_1^2 = 36 * 20^2 = 14400$$

Tabla 3.17. Comparación de costos de las consultas en los dos esquemas de fragmentación

Consulta	FTree			(Kechar & Nait-Bahloul, 2017)		
	TIC	TRC	Costo	TIC	TRC	Costo
Q_1	3720	20000	23720	3160	14400	**17560**
Q_2	795	0	**795**	2220	7200	9420
Q_3	0	0	**0**	2560	14400	16960
Q_4	270	0	**270**	1695	7200	8895
Q_5	0	0	**0**	1920	8100	10020
Q_6	1260	4000	**5260**	1330	6800	8130
Q_7	270	0	**270**	1695	7200	8895

Como se observa en la Tabla 3.17, el esquema generado por FTree obtuvo menor costo en la mayoría de los casos. Solo la primera consulta fue más eficiente en el esquema encontrado por (Kechar & Nait-Bahloul, 2017).

Se utilizaron otras técnicas de clasificación para seleccionar el esquema de fragmentación. Las Tablas 3.18 y 3.19 presentan los resultados de las métricas de evaluación con Naive Bayes y MultiLayerPerceptron.

Tabla 3.18. Resultados de las métricas de evaluación para los dos esquemas con Naive Bayes.

Fragmentos	Precisión	Exhaustividad	Medida F	Área ROC
2	1	1	1	1
3	-	0.933	-	0.938

Tabla 3.19. Resultados de las métricas de evaluación para los dos esquemas con MultiLayerPerceptron.

Fragmentos	Precisión	Exhaustividad	Medida F	Área ROC
2	1	1	1	1
3	-	0.933	-	0.957

Como se observa en las Tablas 20 y 21, los resultados son ligeramente mejores en cuanto a área ROC para los dos algoritmos, pero ambos concuerdan en que el esquema más adecuado es el de dos fragmentos. Además, el modelo obtenido por J48 (Figura 54) tiene mayor facilidad de interpretación, por lo que permite fragmentar el *data warehouse* con base en el atributo trimestre.

Se llevó a cabo un segundo experimento, ahora considerando la matriz de uso de atributos de la Figura 3.62

	query [PK] integer	p1 bit	p2 bit	p3 bit	p4 bit	p5 bit	p6 bit	p7 bit	p8 bit	p9 bit	p10 bit	p11 bit	p12 bit	frequency integer
1	1	1	0	0	0	0	0	0	0	0	0	0	0	3
2	2	0	1	0	0	0	0	0	0	0	0	0	0	4
3	3	0	0	1	0	0	0	0	0	0	0	0	0	2
4	4	0	0	0	1	0	0	0	0	0	0	0	0	5
5	5	0	0	0	0	1	0	0	0	0	0	0	0	2
6	6	0	0	0	0	0	1	0	0	0	0	0	0	3
7	7	0	0	0	0	0	0	1	0	0	0	0	0	4
8	8	0	0	0	0	0	0	0	1	0	0	0	0	7
9	9	0	0	0	0	0	0	0	0	0	0	0	0	4
10	10	0	0	0	0	0	0	0	0	0	1	0	0	10
11	11	0	0	0	0	0	0	0	0	0	0	1	0	3
12	12	0	0	0	0	0	0	0	0	0	0	1	1	4

Figura 3.62. Matriz de uso de atributos para el segundo experimento.

Los predicados y sus selectividades se muestran en la Figura 3.63.

	id_pred [PK] integer	description_pred text	selectividad integer
1	1	anio=2007	2952
2	2	anio=2008	2952
3	3	trimestre='T1'	8856
4	4	anio=2009	2952
5	5	anio=2010	2952
6	6	trimestre='T2'	8856
7	7	anio=2011	2952
8	8	mes='dic'	2952
9	9	anio=2012	2952
10	10	mes='jul'	2952
11	11	anio=2013	2952
12	12	trimestre='T4'	8856

Figura 3.63. Predicados con sus selectividades.

Para este caso, el valor de *W* es 5, por lo que se evaluaron los pasos 8, 9 y 10 del árbol de partición de la Figura 3.64.

Partition tree

Step0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	[0/271584]
Step1	p1	p2	p3	p4,p6	p5	*	p7	p8	p9	p10	p11	p12	[20664/197784]
Step2	p1	p2,p4,p6	p3	*	p5	*	p7	p8	p9	p10	p11	p12	[44280/150552]
Step3	p1,p3	p2,p4,p6	*	*	p5	*	p7	p8	p9	p10	p11	p12	[50184/123984]
Step4	p1,p3,p5	p2,p4,p6	*	*	*	*	p7	p8	p9	p10	p11	p12	[82656/112176]
Step5	p1,p3,p5	p2,p4,p6,p9	*	*	*	*	p7	p8	*	p10	p11	p12	[168264/85608]
Step6	p1,p3,p5	p2,p4,p6,p9	*	*	*	*	p7,p8	*	*	p10	p11	p12	[200736/85608]
Step7	p1,p3,p5	p2,p4,p6,p9	*	*	*	*	p7,p8	*	*	p10	p11,p12	*	[321768/11808]
Step8	p1,p3,p5	p2,p4,p6,p9	*	*	*	*	p7,p8,p10	*	*	*	p11,p12	*	[413280/11808]
Step9	p1,p3,p5,p11,p12	p2,p4,p6,p9	*	*	*	*	p7,p8,p10	*	*	*	*	*	[738000/0]
Step10	p1,p3,p5,p11,p12	p4,p6,p9,p7,p8,p10	*	*	*	*	*	*	*	*	*	*	[1251648/0]
Step11	p4,p6,p9,p7,p8,p10	*	*	*	*	*	*	*	*	*	*	*	[2898864/0]

Figura 3.64. Árbol de partición.

Las Figuras 3.65, 3.66 y 3.67 muestran los árboles de decisión obtenidos por J48 para los esquemas con dos, tres y cuatro fragmentos. El mejor esquema de fragmentación obtenido por FTree es $fr_1=\{p_1, p_3, p_5,p_{11},p_{12}\}$, $fr_2=\{p_2,p_4, p_6, p_9, p_7, p_8, p_{10}\}$ y un tercer fragmento

almacenaría el complemento. La cardinalidad del primer fragmento es 22,140. La cardinalidad del segundo fragmento es 10,824 y la del complemento es 2,460.

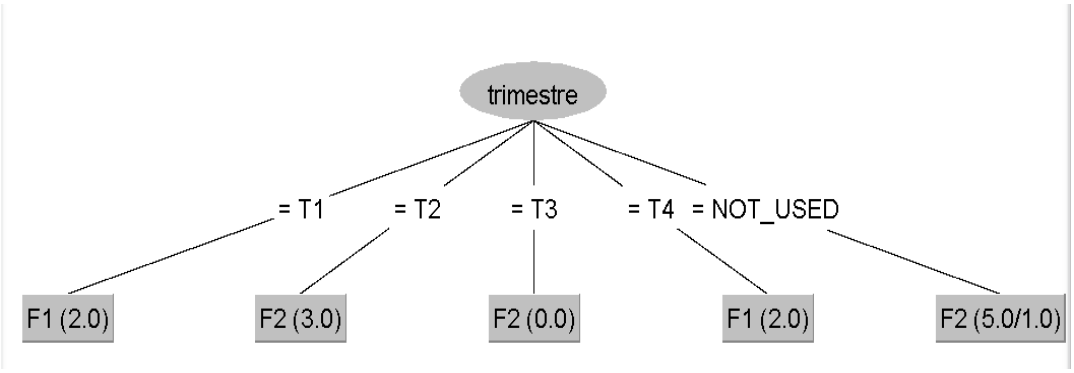


Figura 3.65. Árbol de decisión obtenido por J48 para dos fragmentos.

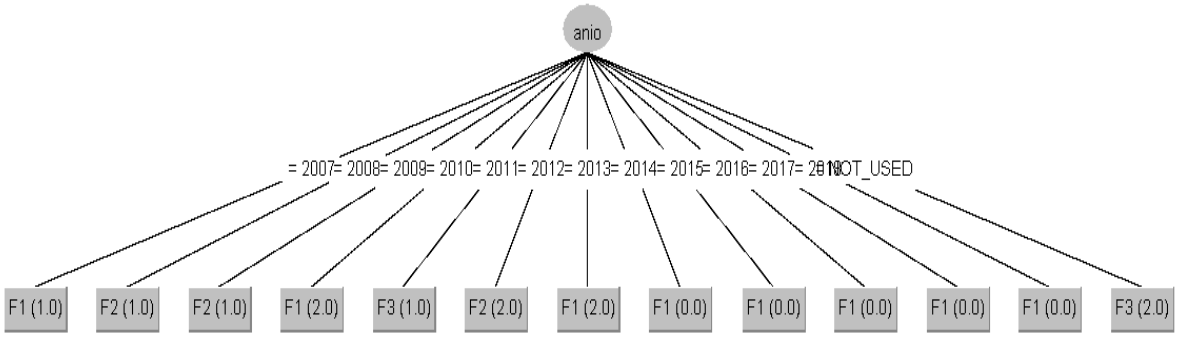


Figura 3.66. Árbol de decisión para tres fragmentos.

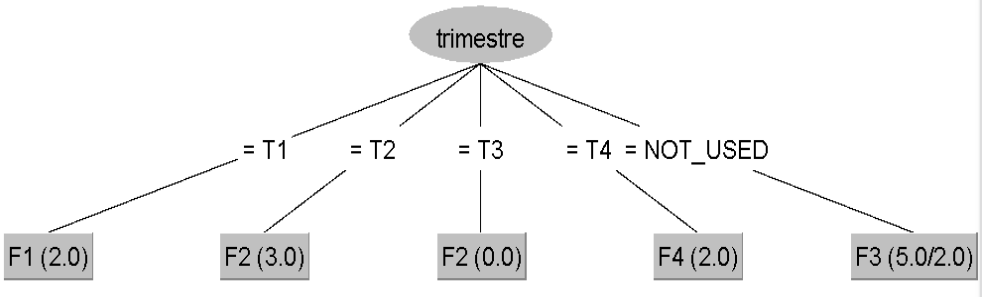


Figura 3.67. Árbol de decisión para cuatro fragmentos.

Las Tablas 3.20, 3.21 y 3.22 contienen los resultados de las métricas de evaluación para los esquemas con dos, tres y cuatro fragmentos utilizando J48, NaiveBayes y MultiLayerPerceptron**.**

Tabla 3.20. Resultados de las métricas de evaluación para los tres esquemas con J48.

Fragmentos	Precisión	Exhaustividad	Medida F	Área ROC
2	**0.825**	**0.750**	0.718	0.657
3	-	0.500	-	0.712
4	0.792	**0.750**	**0.760**	**0.789**

Tabla 3.21. Resultados de las métricas de evaluación para los tres esquemas con Naive Bayes.

Fragmentos	Precisión	Exhaustividad	Medida F	Área ROC
2	**0.764**	**0.750**	**0.752**	**0.800**
3	0.679	0.667	0.667	0.748
4	0.333	0.417	0.367	0.582

Tabla 3.22. Resultados de las métricas de evaluación para los tres esquemas con MultiLayerPerceptron.

Fragmentos	Precisión	Exhaustividad	Medida F	Área ROC
2	**0.927**	**0.917**	**0.915**	0.886
3	0.792	0.750	0.763	0.947
4	0.736	0.667	0.651	**0.934**

Como se observa en las Tablas 20, 21 y 22, J48 obtuvo mejores resultados con el esquema de cuatro fragmentos, mientras que NaiveBayes y MultiLayerPerceptron con los de dos fragmentos. La Tabla 3.23 presenta la comparación de los costos de las 12 consultas en los dos esquemas.

Tabla 3.23. Comparación de costos de las consultas en los dos esquemas de fragmentación

Consulta	Esquema con dos fragmentos			Esquema con cuatro fragmentos		
	TIC	**TRC**	**Costo**	**TIC**	**TRC**	**Costo**
Q_1	30996	0	30996	22140	0	**22104**
Q_2	0	0	**0**	0	0	**0**
Q_3	20664	0	20664	2952	0	**2952**
Q_4	0	0	**0**	0	0	**0**
Q_5	20664	0	**20664**	20664	11808	32472
Q_6	0	0	**0**	0	0	**0**
Q_7	31488	0	31488	23616	0	**23616**
Q_8	55104	0	55104	41328	0	**41328**
Q_9	31488	0	**31488**	35424	0	35424
Q_{10}	78720	0	78720	14760	0	**14760**
Q_{11}	249444	0	249444	6396	0	**6396**
Q_{12}	51660	0	51660	0	0	**0**

Como se observa en la Tabla 3.23, el esquema con cuatro fragmentos es mejor en la mayoría de los casos. Solo en dos consultas, el esquema con dos fragmentos obtuvo menor

costo de ejecución. Por lo tanto, el modelo obtenido por J48 superó a los de Naive Bayes y MultiLayerPerceptron.

3.8 Mejorar el método desarrollado

En esta etapa del desarrollo del proyecto de tesis se identificaron los aspectos débiles de la misma y se procedió a hacer mejoras que beneficien al proyecto, para esto fue necesario realizar una serie de cambios en el algoritmo de generación de conjuntos de datos propuesto puesto que en la etapa anterior se encontraron áreas de mejora, así mismo se desarrolló una aplicación Web que proporciona una interfaz gráfica al usuario y que permite conectar tanto al DW como al algoritmo de fragmentación. En esta etapa de mejoramiento el algoritmo modificado se probó usando el SSB, para conocer su comportamiento con datos sintéticos y siguiendo el mismo proceso que tuvo el desarrollo del algoritmo en la primera etapa del proyecto de tesis. Los resultados obtenidos hasta ahora demostraron que el sistema funciona eficientemente y que conlleva beneficios esta serie de cambios.

En primer lugar, se hicieron varias modificaciones al algoritmo para generar conjuntos de datos. En un principio se consideró que el último conjunto de datos generado era $data\_set_W$, donde W es el número máximo de fragmentos que el administrador de la base de datos (DBA) quiere generar. Sin embargo, no se estaba contemplando el complemento de los fragmentos, por lo tanto, solo será necesario analizar W-2 fragmentos. En este algoritmo también se llevó a cabo un segundo cambio, en la versión anterior se consideró obtener la Matriz de Beneficio de Fusión (MPM, *Merging Profit Matrix*) antes de iniciar con la construcción del árbol de partición (PT, *Partition Tree*). En esta versión, se eliminó ese paso por considerarse innecesario. Además, anteriormente, se utilizaba un contador c que iniciaba en 0 y servía para identificar al conjunto de datos generado, únicamente se generaba un conjunto de datos cuando el número de paso i igualaba o excedía W, lo cual provocaba que los esquemas de fragmentación obtenidos excedieran el valor de W. En esta versión, c inicia en 2, ya que el primer esquema generado por el método siempre tendrá dos fragmentos, este se obtiene cuando el paso i es igual a r-(W-c). La Tabla 3.24 muestra distintos valores de W considerando la construcción de un PT con 10 y 20 pasos, así como los pasos que se deben considerar. El Algoritmo 1 modificado se muestra a continuación.

```
1   Data: PUM of the fact table F and dim (a set of queries Q={q1, q2, …, qn},
2   the frequency fi of every query qi, a set of predicates Pr={p1, p2, …, pr}, the
3   selectivity selj of every predicate pj), W
4   Result: data sets D={data_set2, data_set3, …, data_setw-1}
5   c=2;
6   for each stepi ∈PT |1≤i≤r-1 do
7      getMPM(PUM, MPM);
8      choose two fragments with the greatest merging profit;
9      fuse the fragments;
10     if i>=r-(W-c)
11        generate data_setc;
12        c=c+1;
13     end;
14  end;
```

Algoritmo 1. Generar data sets.

Tabla 3.24. Conjuntos de datos obtenidos con distintos valores de *W*, *r* e *i*.

W	R	I	Conjuntos de datos
3	10	9	*data_set2*
4	10	8,9	*data_set2, data_set3*
3	20	19	*data_set2*
4	20	18,19	*data_set2, data_set3*

También se modificó el modelo de costos, el nuevo modelo considera que el costo de una consulta OLAP q_i está dado por la cantidad de datos irrelevantes accedidos (TIC) y la cantidad de datos relevantes ubicados en otros fragmentos (TRC) como se muestra en la Ecuación 2.

$$Cost(q_i) = TIC(q_i) + TRC(q_i) \quad (2)$$

La cantidad de datos irrelevantes a los que se accede desde una consulta se calcula multiplicando el número de tuplas irrelevantes a las que accede la consulta en cada fragmento fr_k por la frecuencia de la consulta *(fi)*. La Ecuación 3 se usa para obtener TIC donde *card(frk)* es la cardinalidad del fragmento fr_k, sel_j es la selectividad del predicado p_j y rt_j es el número de tuplas del predicado p_j ubicadas en otros fragmentos requeridos para responder la consulta.

$$TIC(q_i) = \sum_{k=1}^{m} (card(fr_k) - \sum_{p_j | PUM(q_i,p_j)=1 \wedge p_j \in fr_k} (sel_j - rt_j)) * f_i \quad (3)$$

La cantidad de datos relevantes ubicados en otros fragmentos se obtiene mediante la Ecuación 4, donde el número de tuplas relevantes ubicadas en otros fragmentos se multiplica por la frecuencia de la consulta al cuadrado y por la cantidad de fragmentos (nf_i) necesarios para responder la consulta.

$$\mathrm{RTC}(\mathrm{q_i}) = \sum_{\mathrm{p_j}|\mathrm{PUM}(q_i, p_j)=1 \wedge rt_j>0}^{\mathrm{n}} rt_j * \mathrm{f_i^2} * nf_i \qquad (4)$$

Se desarrolló una aplicación Web usando el lenguaje de programación Java, la API de Weka, y el Sistema Gestor de Bases de Datos PostgreSQL para aplicar FTree en un DW, primero el DWA debe proporcionar la IP del servidor donde está ubicado el almacén de datos, el puerto, así como su usuario y contraseña. Luego, se selecciona el DW a fragmentar, la tabla de hechos, además del número máximo de fragmentos (*W*), esto se visualiza en las Figuras 3.70-3.72. Cuando se proporcionan estos datos y continúa la fragmentación, se crea el esquema FTree de la Figura 3.68 en el DW. La Tabla 3.25 describe cada elemento de este esquema.

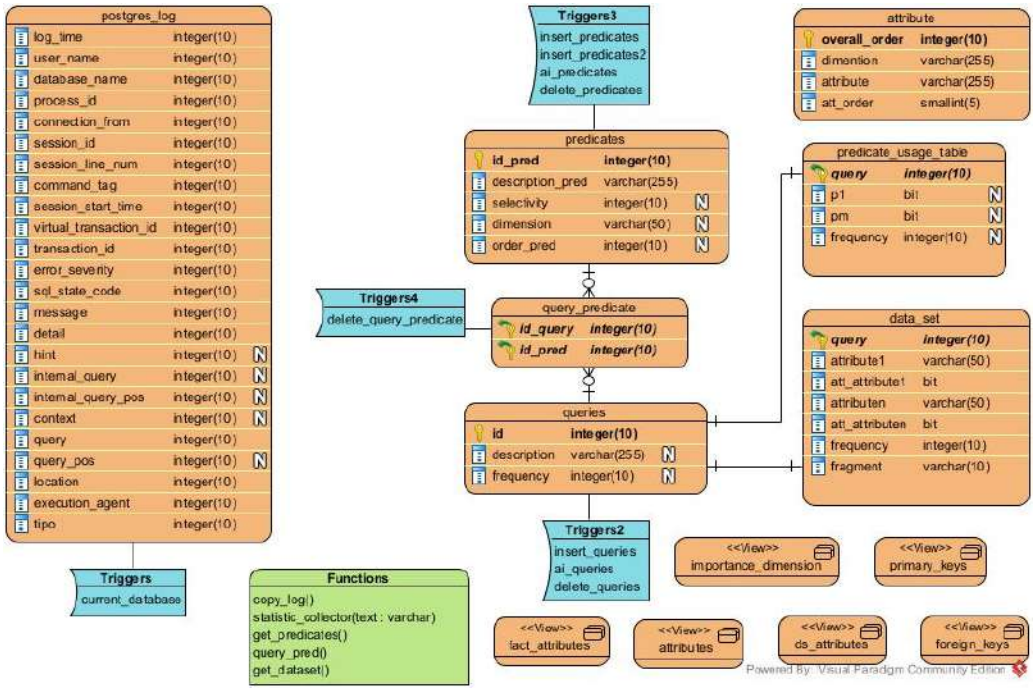


Figura 3.68. Modelo físico del esquema FTree

Tabla 3.25. Elementos del esquema FTree creados en el DW

Nombre	Tipo	Descripción
postgres_log	Tabla	Almacena todas las consultas que se encuentran en el archivo log.
current_database	Disparador	Antes de insertar una consulta en la tabla *postgres_log* verifica que solo se guarden las consultas relevantes.
copy_log()	Función	Copia el contenido del log de transacciones del día actual en la tabla *postgres_log*.
queries	Tabla	Almacena el identificador, la descripción y la frecuencia de las consultas.
delete_queries	Disparador	Después de eliminar una consulta en la tabla *queries*, reduce el valor del identificador en 1.
insert_queries	Disparador	Antes de insertar una consulta en la tabla *queries*, si la consulta ya está almacenada en esta tabla, entonces no se inserta y su frecuencia se incrementa en 1.
ai_queries	Disparador	Posterior a insertar una consulta en la tabla *queries,* actualiza su identificador si es mayor que el número máximo de filas.
statistic_collector(text)	Función	Analiza las consultas almacenadas en *postgres_log* para insertar en la tabla *queries* solo las que se ejecutaron en la tabla que se quiere fragmentar. Su parámetro es la tabla de hechos.
attributes	Vista	Almacena los nombres de las dimensiones, así como de los atributos de cada dimensión y el orden en que aparecen en la tabla de dimensión.
attribute	Tabla	Guarda los datos de la vista *attributes* y agrega un atributo de orden global.
fact_attributes	Vista	Almacena el nombre de la tabla de hechos además de los nombres y el orden de cada uno de sus atributos.
predicates	Tabla	Guarda el identificador, la descripción y la selectividad de los predicados. Asi como sus dimensiones y orden en esa dimensión.
insert_predicates	Disparador	Antes de insertar en la tabla *predicates*, analiza que solo se tomen los predicados de las consultas que involucren a la tabla de hechos con la tabla de dimensión seleccionada.
insert_predicates2	Disparador	Antes de insertar en la tabla *predicates*, valida que solo se inserten predicados diferentes.
ai_predicates	Disparador	Después de insertar un predicado en la

Nombre	Tipo	Descripción
		tabla *predicates*, actualiza su identificador si es mayor que el número máximo de filas.
delete_predicates	Disparador	Después de eliminar un predicado en la tabla *predicates*, reduce el valor del identificador en 1.
get_predicates()	Función	Obtiene los predicados de las consultas almacenadas en *queries* y los guarda en *predicates*.
query_predicate	Tabla	Guarda la relación entre las consultas y los predicados.
delete_query_predicate	Disparador	Después de eliminar en la tabla *query_predicate*, reduce el valor de los identificadores de la consulta y del predicado en 1.
predicate_usage_table	Tabla	Analiza qué predicados aparecen en qué consultas y guarda esta relación en *query_predicate*.
data_set	Tabla	Crea los conjuntos de datos que se utilizan para entrenar el árbol de decisión.
query_pred()	Función	Analiza cuales predicados aparecen en que consultas y guarda esta relación en *query_predicates*.
get_dataset()	Función	Crea el conjunto de datos que se usa para la construcción del árbol de decisión.
importance_dimension	Vista	Almacena la importancia de las dimensiones.
ds_attributes	Vista	Los atributos de los conjuntos de datos y su orden se encuentran en esta vista.
primary_keys	Vista	Almacena el nombre y los atributos de las llaves primarias para todas las tablas del DW.
foreign_keys	Vista	El nombre de las claves foráneas para cada dimensión y el atributo en la tabla de hechos involucrado en esta restricción se encuentra en esta vista.

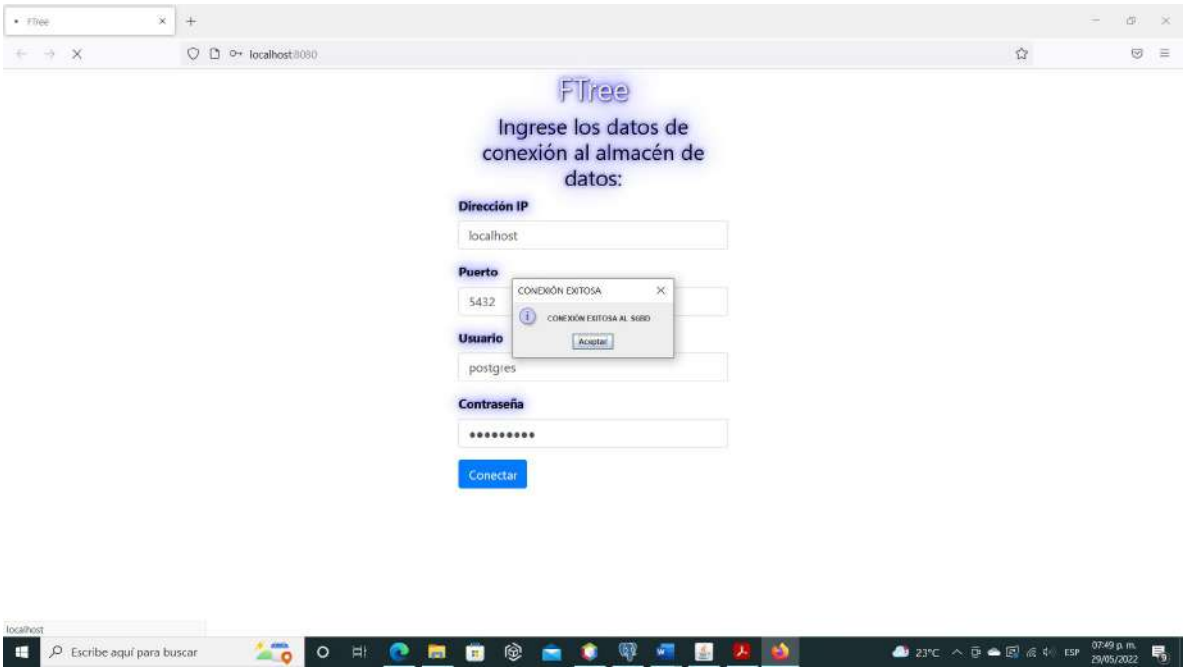


Figura 3.69. Página de conexión al almacén de datos.

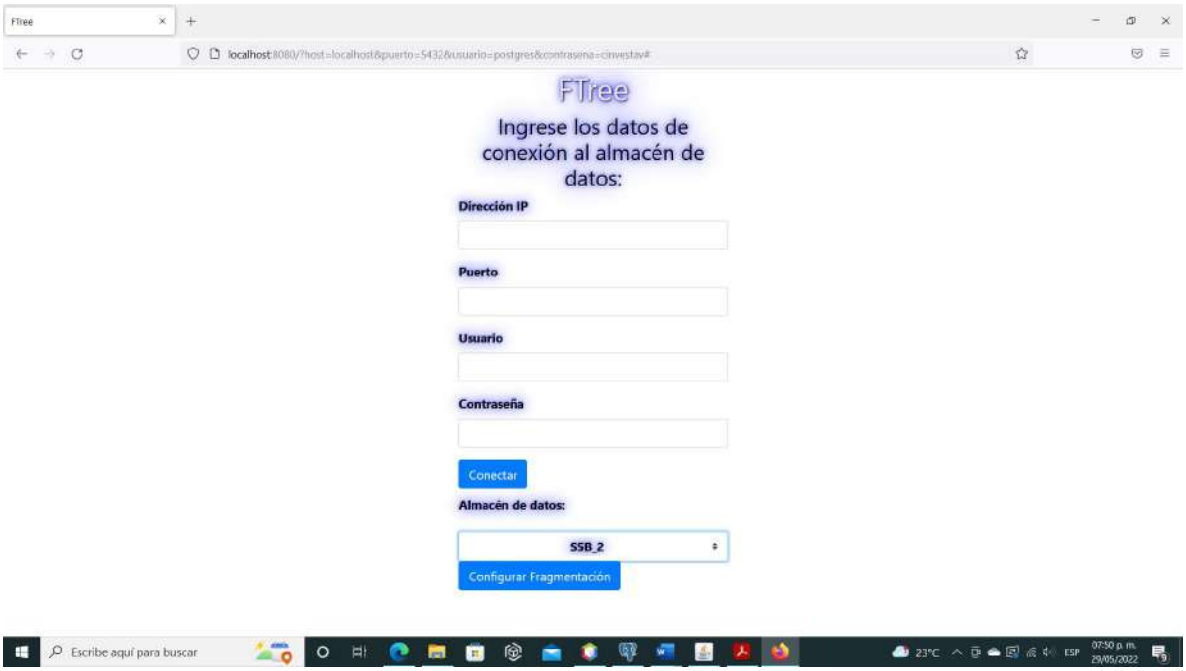


Figura 3.70. Página de conexión después de ingresar los datos correctos.

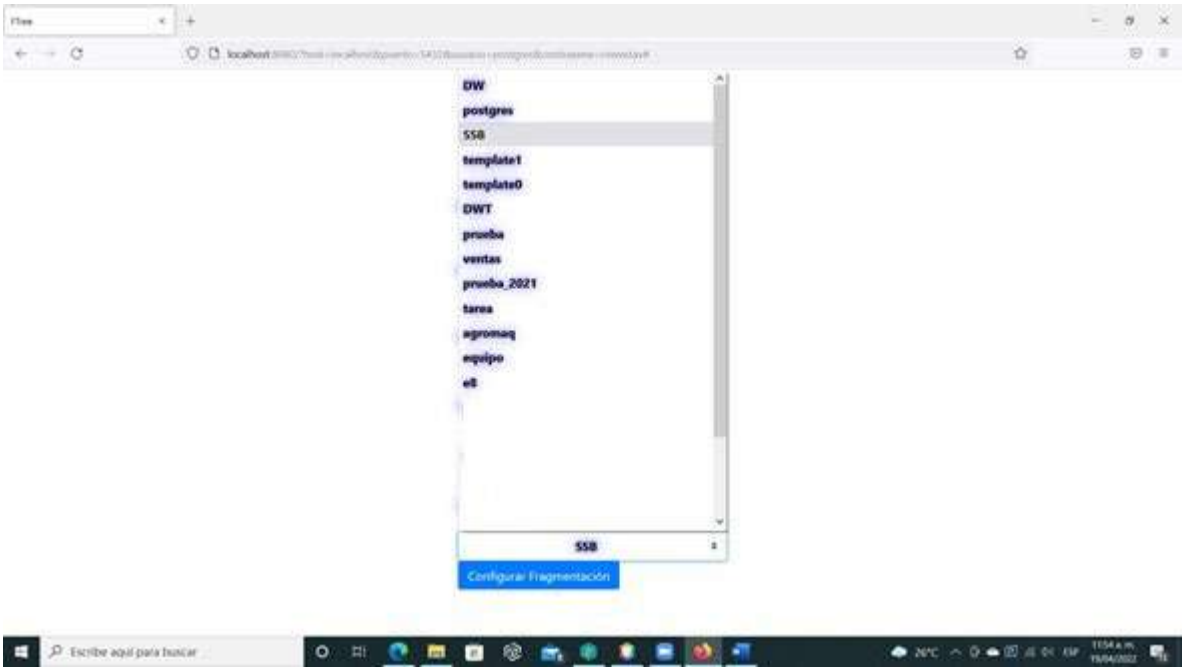


Figura 3.71. Página de conexión con lista desplegable de bases de datos.

En la Figura 3.73 se observa cómo se creó en el almacén de datos SSB. Posteriormente, se ejecuta la función *copy_log()* la cual recupera todas las consultas que se realizaron en el almacén de datos del archivo de registro (*log*) y las inserta en la tabla *postgres_log* (ver Figura 3.73), esto dispara *current_database()*, que verifica que solo se guarden en *postgres_log* las consultas realizadas en el almacén de datos.

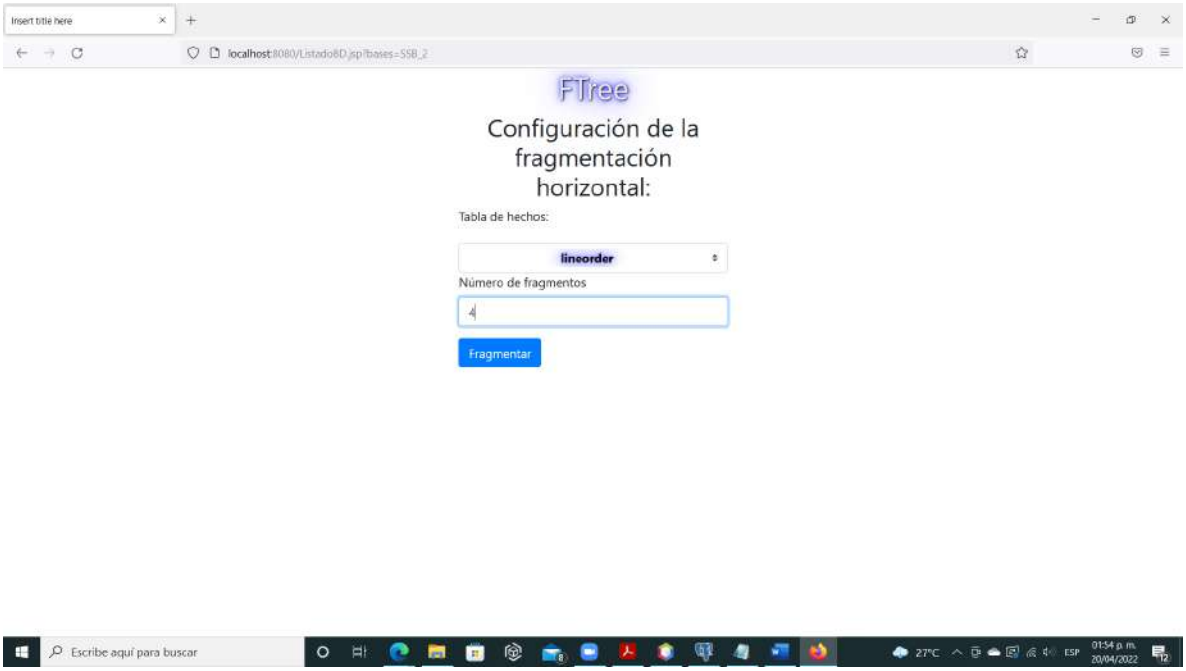


Figura 3.72. Página de configuración de la fragmentación.

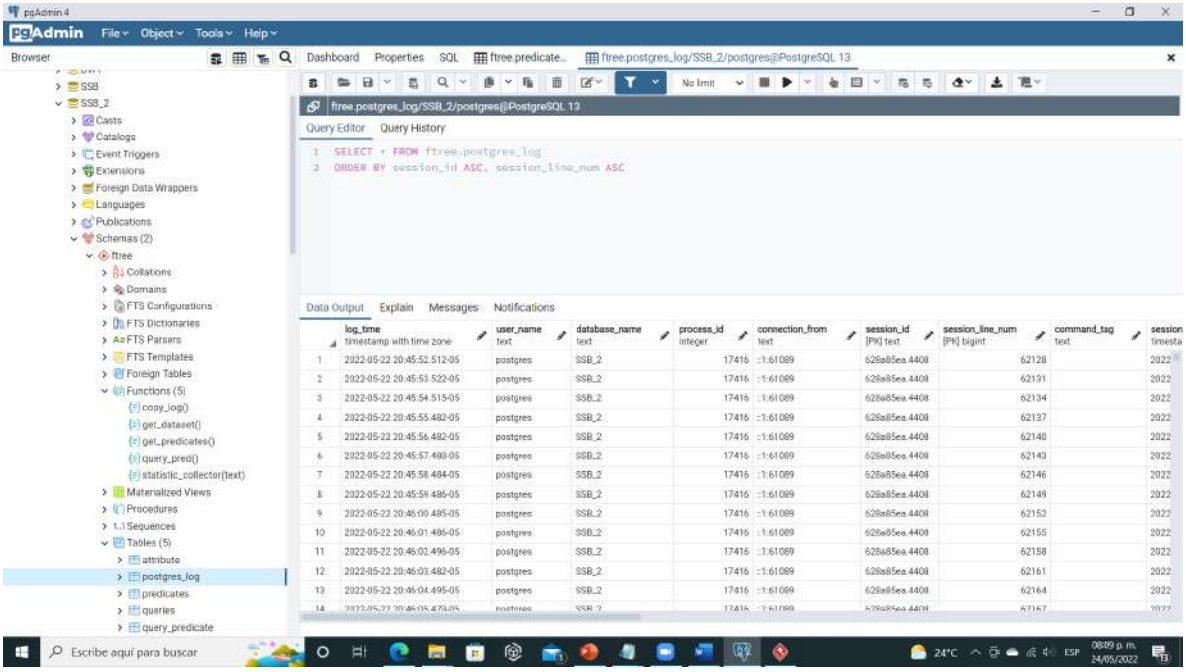


Figura 3.73. SSB con esquema ftree.

Después, como se observa en la Figura 3.74, se ejecuta la función *statistic_collector(ft),* donde *ft* es la tabla de hechos seleccionada en la página de la Figura 3.72, para insertar en la tabla *queries* las consultas relevantes, es decir, aquellas que implican reuniones de la tabla de hechos con las tablas de dimensión. Las consultas relevantes se observan en la página de resultados de la fragmentación de la Figura 3.75.

A continuación, se ejecuta la función *get_predicates()* (ver Figura 3.74) para insertar en la tabla *predicates* de la Figura 3.76 los predicados de las consultas, la selectividad de los predicados y las tablas de dimensión a las que pertenece cada predicado. Esta información junto con la frecuencia de las consultas es necesaria para calcular la importancia de cada dimensión en la vista de la Figura 3.77. El contenido de la vista se aprecia en la página de la Figura 3.78.

```
String sqli2 ="select ftree.statistic_collector('"+ft+"');";
Statement st2;
try
{
        st2 = cn.createStatement();
        st2.executeQuery(sqli2);
        out.println("Se guardaron las consultas relevantes del DW "+base);
}

catch(Exception e)
{
        out.println(e);

}
String sqli3 ="select * from ftree.get_predicates();";
Statement st3;
try
{
        st3 = cn.createStatement();
        st3.executeQuery(sqli3);
        out.println("Se obtuvieron los predicados de las consultas "+base);
}
catch(Exception e)
{
        out.println(e);
}
```

Figura 3.74. Código de la aplicación.

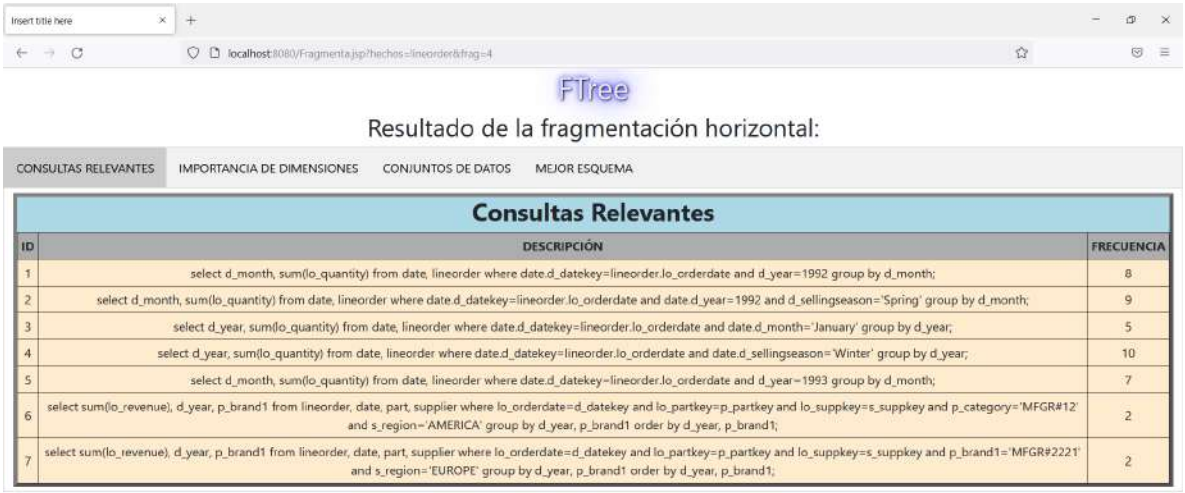


Figura 3.75. Página de resultados de la fragmentación con las consultas relevantes.

	id_pred [PK] integer	description_pred text	selectivity integer	dimention name	order_pred integer
1	1	d_year=1992	366	date	1
2	2	d_sellingseason='Spri...	210	date	2
3	3	d_month='January'	217	date	3
4	4	d_sellingseason='Wint...	632	date	4
5	5	d_year=1993	365	date	5
6	6	p_category='MFGR#12'	7883	part	1
7	7	s_region='AMERICA'	378	supplier	1
8	8	p_brand1='MFGR#2221'	203	part	2
9	9	s_region='EUROPE'	380	supplier	2

Figura 3.76. Tabla *predicates*.

```
CREATE OR REPLACE VIEW public.importance_dimention
 AS
 SELECT predicates.dimention,
    sum(predicates.selectivity * queries.frequency) AS importance
   FROM ftree.predicates,
    ftree.queries,
    ftree.query_predicate
  WHERE predicates.id_pred = query_predicate.id_pred AND queries.id = query_predicate.id_query
  GROUP BY predicates.dimention
  ORDER BY (sum(predicates.selectivity * queries.frequency)) DESC;
```

Figura 3.77. Código de la vista *importance_dimention*

Figura 3.78. Página de resultados de la fragmentación con la importancia de las dimensiones

En la página de la Figura 3.79 se muestra la matriz de uso de atributos que se obtuvo a partir de los datos almacenados en las tablas *queries* y *predicates*, así como el árbol de partición generado con el Algoritmo 1. El cual da como resultado dos conjuntos de datos que se evaluaron con el algoritmo J48 para seleccionar el mejor esquema.

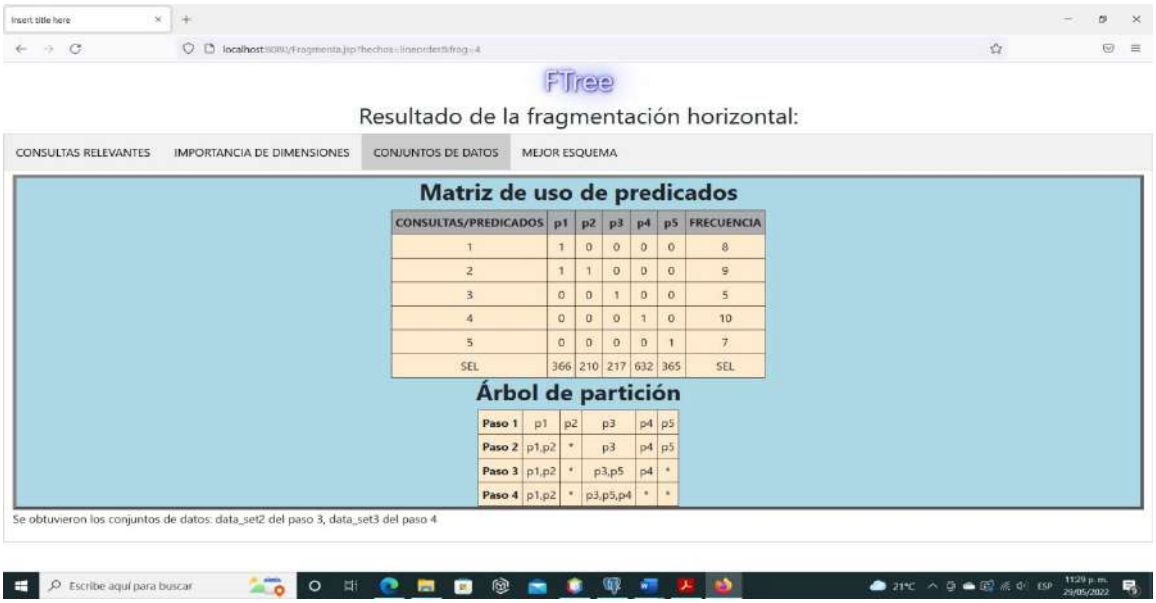


Figura 3.79. Página de resultados de la fragmentación con el árbol de partición.

CAPÍTULO 4. RESULTADOS

En esta sección se presentan los resultados de la aplicación de FTree en el *benchmark* SSB y en el DW turístico, enseguida se explican los cuatro pasos del método y más tarde se detalla la aplicación web para aplicar FTree en un DW.

4.1 Método FTree

En la Tabla 4.1 se muestran las consultas OLAP realizadas en SSB.

Tabla 4.1. Consultas ejecutadas en SSB

Consulta	Frecuencia
q_1: SELECT d_month, sum(lo_quantity) from date, lineorder WHERE date.d_datekey=lineorder.lo_orderdate AND date.d_year=1992 GROUP BY d_mes;	3
q_2: SELECT d_mes, sum(lo_quantity) FROM date, lineorder WHERE date.d_datekey=lineorder.lo_orderdate AND date.d_year=1992 AND date.d_sellingseason='Summer' GROUP BY d_month;	5
q_3: SELECT d_month, avg(lo_ordtotalprice) from date, lineorder WHERE date.d_datekey=lineorder.lo_orderdate AND date.d_year=1993 GROUP BY d_month;	4
q_4: SELECT d_month, avg(lo_ordtotalprice) FROM date, lineorder WHERE date.d_datekey=lineorder.lo_orderdate AND date.d_year=1993 AND date.d_sellingseason='Christmas' GROUP BY d_month;	3
q_5: SELECT d_year, sum(lo_quantity) FROM date, lineorder WHERE date.d_datekey=lineorder.lo_orderdate AND date.d_month='January' GROUP BY d_year;	3
q_6: SELECT d_month, sum(lo_revenue) FROM lineorder, date WHERE date.d_datekey=lineorder.lo_orderdate and date.d_year=1995 GROUP BY d_month;	2
q_7: SELECT d_year, sum(lo_quantity) FROM date, lineorder WHERE date.d_datekey=lineorder.lo_orderdate AND date.d_sellingseason='Winter' GROUP BY d_year;	5
q_8: SELECT sum(lo_revenue), d_year, p_brand1 FROM lineorder, date, part, supplier WHERE lo_orderdate=d_datekey AND lo_partkey=p_partkey AND lo_suppkey=s_suppkey AND p_category='MFGR#12' AND s_region='AMERICA' GROUP BY d_year, p_brand1 ORDER BY d_year, p_brand1;	2
q_9: SELECT sum(lo_revenue), d_year, p_brand1 FROM lineorder, date, part, supplier WHERE lo_orderdate=d_datekey AND lo_partkey=p_partkey AND lo_suppkey=s_suppkey AND p_brand1='MFGR#2221' AND s_region='EUROPE' GROUP BY d_year, p_brand1 ORDER BY d_year, p_brand1;	2

Para llevar a cabo la fragmentación se realizan los cuatro pasos de FTree:

- **Obtener consultas relevantes del DW.** Se consideran las consultas relevantes que se muestran en la Tabla 28. La Tabla 4.2 muestra los predicados utilizados por las consultas y su selectividad.

Tabla 4.2. Predicados usados por las consultas

Pr	Descripción	S
p_1	*d_year=1992*	sel_1=907,987
p_2	*d_sellingseason='Summer'*	sel_2=2,076,040
p_3	*d_year=1993*	sel_3=908,288
p_4	*d_sellingseason='Christmas'*	sel_4=912,008
p_5	*d_mont='January'*	sel_5=541,483
p_6	*d_year=1995*	sel_6=909,991
p_7	*d_sellingseason=' Winter '*	sel_7=1,577,160
p_8	*p_category=' MFGR#12'*	sel_8=236,816
p_9 p_{10}	*s_region='AMERICA"* *p_brand1='MFGR#2221',*	sel_9=1,134,371 sel_{10}=5,848
p_{11}	*s_region='EUROPE'*	sel_{11}=1,140,311

- **Seleccionar la mejor dimensión.** La Figura 4.1 muestra la importancia de las dimensiones en la aplicación web para el escenario. Por ejemplo, la importancia de la dimensión *part* es 485,328 porque se multiplica f_8*sel_8+f_9*sel_{10}=2*236,816+2*5,848 de acuerdo con la Ecuación (1) del capítulo anterior.

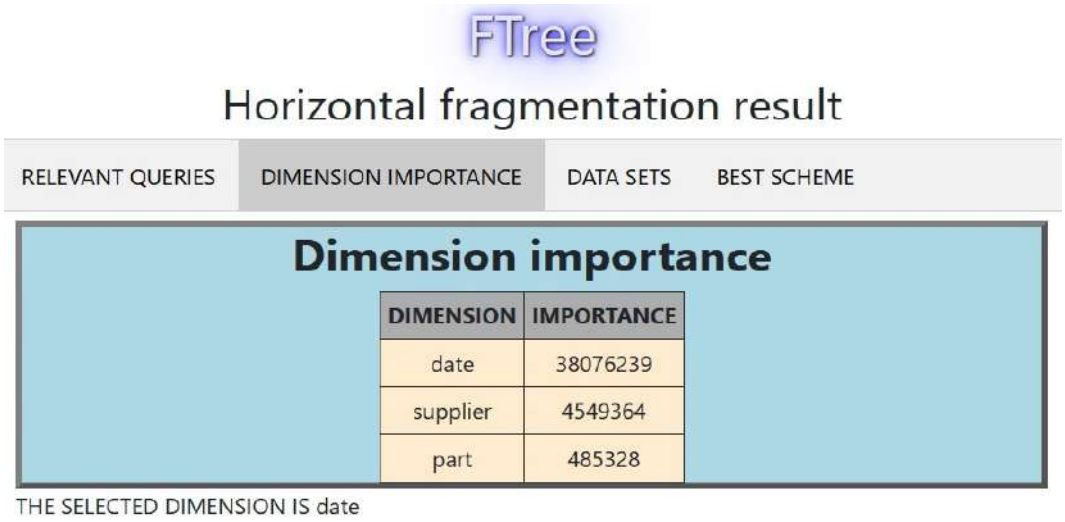
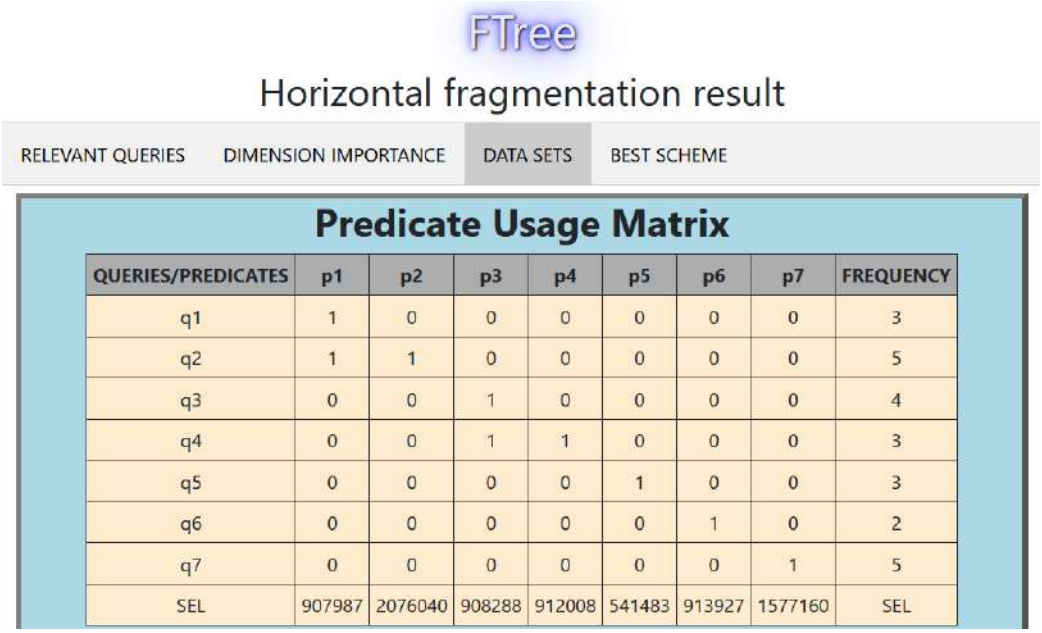


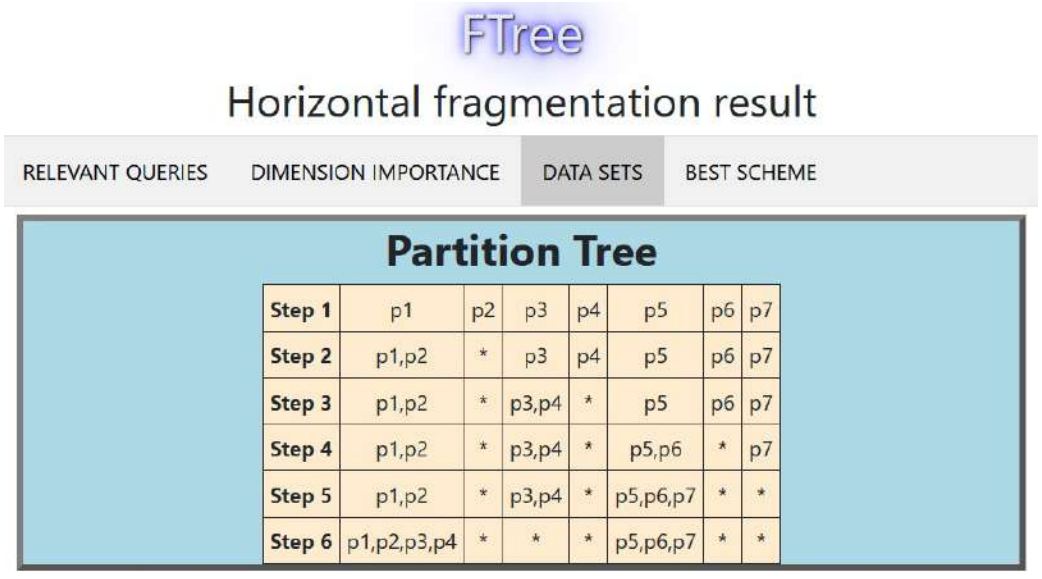
Figura 4.1. Importancia de la dimensión para el escenario

- **Crear conjuntos de datos.** Este paso consiste en construir conjuntos de datos *W*-1 utilizados para entrenar el algoritmo J48 para seleccionar el mejor esquema de fragmentación horizontal (EFH). La Figura 4.2 muestra la PUM para el ejemplo en la aplicación web y la Figura 4.3 presenta el PT. La Figura 4.4 muestra *data_set2.arff*, que es uno de los conjuntos de datos obtenidos como resultado del Algoritmo 1 de la Sección anterior. Este conjunto de datos incluye la carga de trabajo proporcionada como entrada a J48 para construir el árbol de decisión.



QUERIES/PREDICATES	p1	p2	p3	p4	p5	p6	p7	FREQUENCY
q1	1	0	0	0	0	0	0	3
q2	1	1	0	0	0	0	0	5
q3	0	0	1	0	0	0	0	4
q4	0	0	1	1	0	0	0	3
q5	0	0	0	0	1	0	0	3
q6	0	0	0	0	0	1	0	2
q7	0	0	0	0	0	0	1	5
SEL	907987	2076040	908288	912008	541483	913927	1577160	SEL

Figura 4.2. Matriz de uso de predicados para el escenario



Step 1	p1	p2	p3	p4	p5	p6	p7
Step 2	p1,p2	*	p3	p4	p5	p6	p7
Step 3	p1,p2	*	p3,p4	*	p5	p6	p7
Step 4	p1,p2	*	p3,p4	*	p5,p6	*	p7
Step 5	p1,p2	*	p3,p4	*	p5,p6,p7	*	*
Step 6	p1,p2,p3,p4	*	*	*	p5,p6,p7	*	*

Figura 4.3. Árbol de partición para el escenario

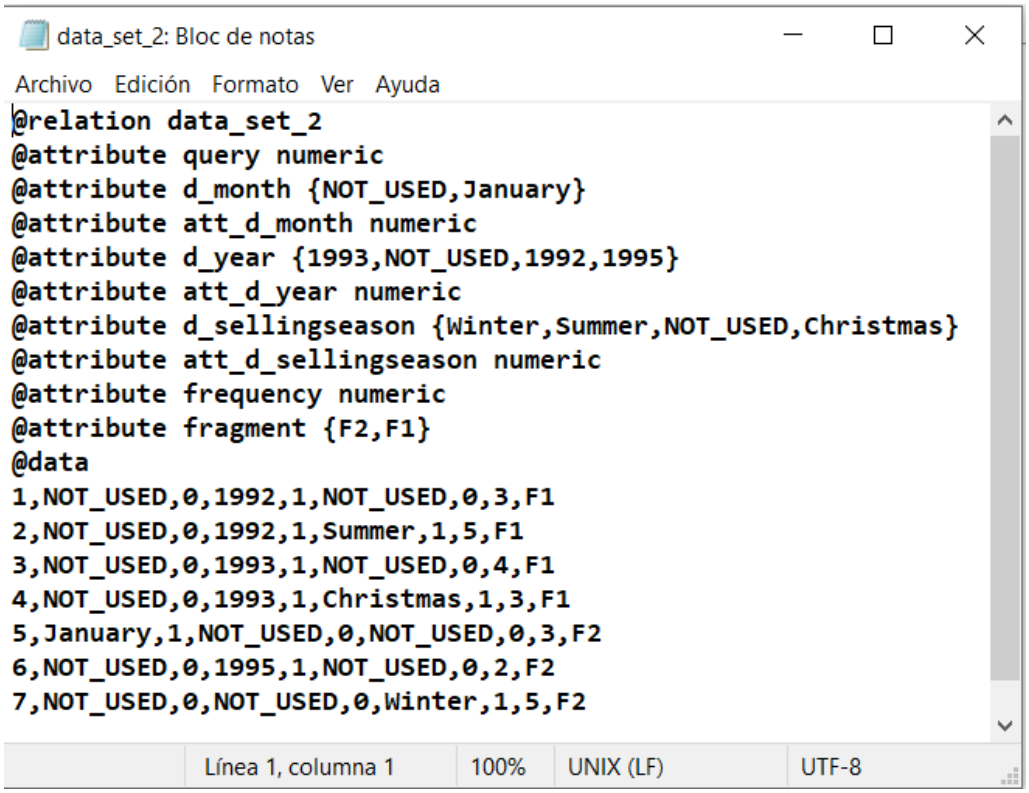


Figura 4.4. Data set para el paso 6 del árbol de partición

La Figura 4.4 muestra que las instancias del conjunto de datos representan las consultas y los atributos corresponden a los predicados, se agrega una variable numérica para cada atributo involucrado en los predicados. Además, la frecuencia de las consultas se considera en los conjuntos de datos. El atributo de etiqueta de clase es *fragment,* el valor de este atributo se asigna de acuerdo con el árbol de partición y cuando una consulta utiliza predicados ubicados en diferentes fragmentos, se considera la selectividad. Por ejemplo, según la Figura 82, q_4 usa los predicados p_3 y p_4. La selectividad de estos predicados es sel_3=908,288, sel_4=912,008. En el segundo paso del árbol de partición de la Figura 4.3, están en diferentes fragmentos, p_3 está en el segundo fragmento (F2) y p_4 está en el tercero (F3). Por lo tanto, la etiqueta F3 se asigna a q_4, porque $sel4 > sel_3$.

- **Aplicar J48 a conjuntos de datos.** La API (Interfaz de Programación de Aplicaciones) de Weka se utilizó para aplicar el algoritmo J48 a los conjuntos de datos W-1. Se seleccionó el conjunto de datos con los mejores resultados logrados para cuatro métricas de evaluación (Precisión, Recuperación, Área ROC y Medida F). El mejor esquema de fragmentación horizontal del ejemplo fue $data\_set_2$, como se muestra en la Figura 4.6. Se usó una validación cruzada de 5 pliegues. La Figura 4.6 muestra el árbol de decisiones para $data\_set_2$. Por lo tanto, el esquema de fragmentación es $fr_1 = \{p_1, p_2, p_3, p_4\}$, $fr_2 = \{p_5, p_6, p_7\}$, $fr_3 = \{\neg (p_1, p_2, p_3, p_4), \neg (p_5, p_6, p_7)\}$.

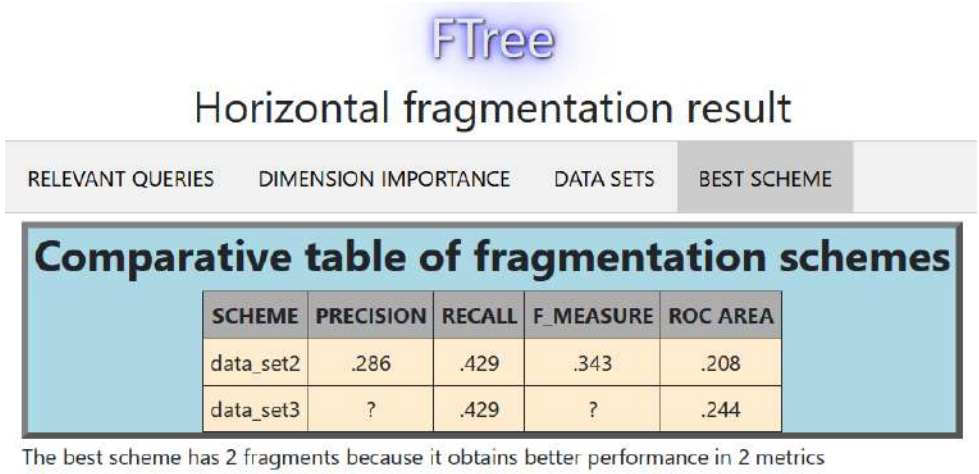


Figura 4.5. Tabla comparativa de esquemas de fragmentación

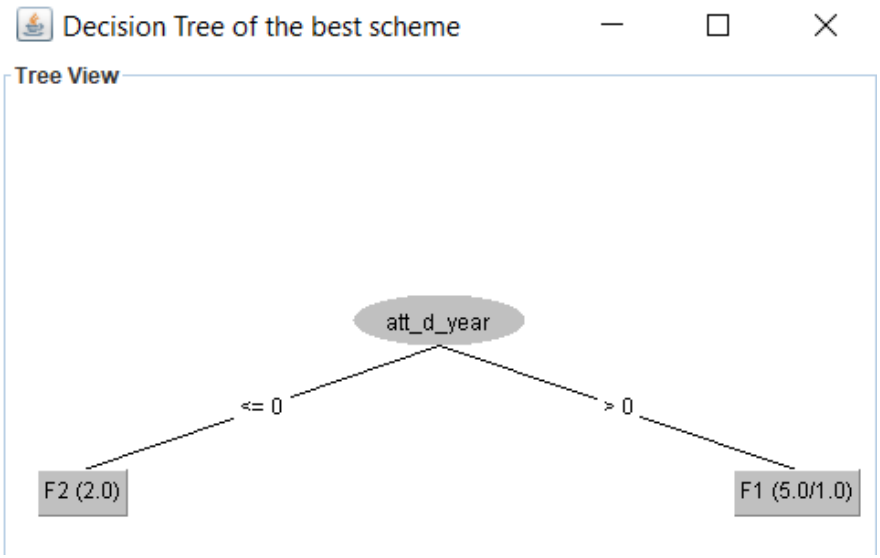


Figura 4.6. Árbol de decisión del mejor esquema encontrado (con mejores resultados en las métricas de evaluación)

La Figura 4.3 muestra el árbol de partición en la aplicación Web, mientras que la Figura 4.2 representa una MUP, la Figura 4.5 presenta la tabla comparativa de los dos esquemas de fragmentación, y la Figura 4.6 exhibe un árbol de decisión en la aplicación Web.

En la siguiente sección, se comparan los esquemas de fragmentación horizontal obtenidos por FTree con J48 versus FTree con otras técnicas de clasificación como Naïve Bayes y MultiLayer Perceptron. Esos algoritmos fueron usados en el análisis comparativo porque son técnicas de clasificación ampliamente conocidas por su efectividad (Anitha & Vanitha, 2022; Razdan et al., 2021).

4.2 Aplicación de FTree en el Data Warehouse Turístico

Esta sección incluye los resultados obtenidos con FTree usando un DW real y la comparación de FTree con J48 contra otras técnicas de clasificación usando un modelo de costo.

4.2.1 Descripción del segundo escenario: El Data Warehouse Turístico

Para evaluar FTree con datos reales, se utilizó el Data Warehouse turístico construido anteriormente (Figura 4.7 y Tabla 4.3), mismo que se desarrolló siguiendo la metodología propuesta por (Kimball & Ross, 2016).

Figura 4.7. Modelo Multidimensional del DWT

Tabla 4.3. Tablas del DWT

Nombre	Tipo	Atributos	Tuplas
actividad_hotelera	Hechos	9	35424
visitantes_internacionales	Hechos	5	576
d_lugar	Dimensión	4	123
d_tiempo	Dimensión	4	144
d_turismo	Dimensión	3	4
d_turista	Dimensión	2	2

4.2.2. Aplicación de FTree en el Data Warehouse Turístico

Se aplicó y probó FTree en el Data Warehouse turístico. La tabla de hechos fragmentada fue Actividad_ hotelera. Se realizó el primer experimento utilizando la matriz de uso predicado (MUP) de la Tabla 4.4.

Tabla 4.4. Matriz de uso de predicados para el primer experimento

Q/P	p_1	p_2	p_3	p_4	p_5	p_6	p_7	F_i
q_1	1	1	1	0	0	0	0	20
q_2	1	0	0	1	0	0	0	15
q_3	1	0	0	0	1	0	0	15
q_4	0	1	0	0	0	1	0	35
q_5	0	1	1	0	0	0	0	35
q_6	1	0	0	0	0	0	1	30
q_7	0	0	0	1	0	0	0	10
q_8	0	0	0	0	0	0	1	10
q_9	0	0	0	0	1	0	0	10
q_{10}	0	0	1	0	0	0	0	10
s_j	2952	8856	2952	2952	2952	2952	2952	

En este caso, el número máximo de fragmentos permitido por el administrador del almacén de datos (W) es 4. Los predicados considerados se presentan en la Tabla 4.5. Usando la PUM de la Tabla 4.4, el árbol de partición de la Figura 4.8 fue obtenido por FTree. La Tabla 4.5. Predicados para el primer experimento muestra la comparación entre $data\_set_2$ con fragmentos $fr_1=\{p_1, p_4, p_5, p_7\}$, $fr_2=\{p_2, p_3, p_6\}$ y $data\_set_3$ con fragmentos $fr_1=\{p_1, p_4, p_7\}$,

fr_2={p_2, p_3, p_6}, fr_3={p_5}. Se utilizó una validación cruzada de 8 pliegues. El mejor esquema es $data\_set_2$. El árbol de decisión obtenido por FTree se visualiza en la Figura 4.9.

Tabla 4.5. Predicados para el primer experimento

Pr	Pr
p_1	*mes*='feb'
p_2	*trimestre*='T3'
p_3	*año=2018*
p_4	*año=2014*
p_5	*año*=2017
p_6	año=2015
p_7	*año=2016*

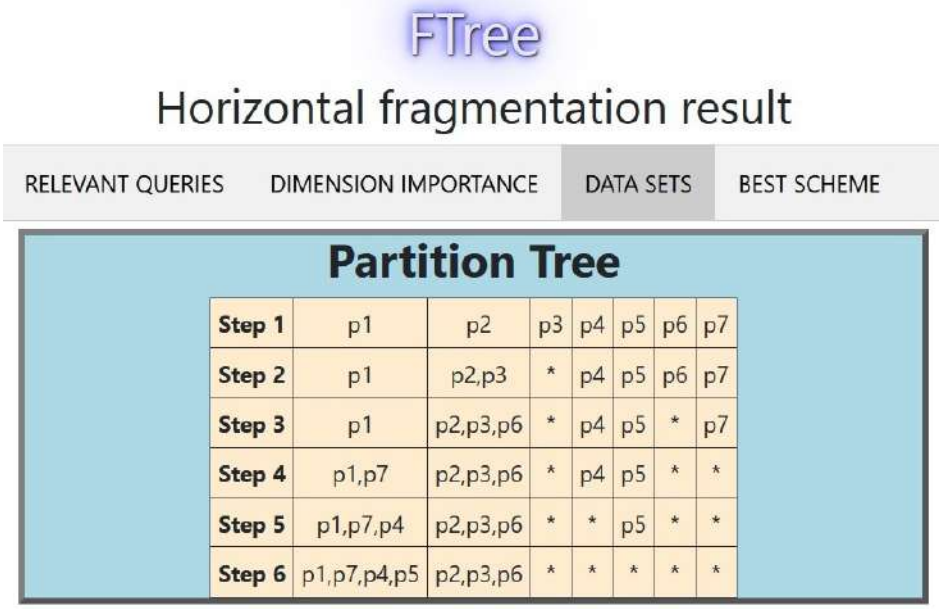


Figura 4.8. Árbol de partición para el primer experimento

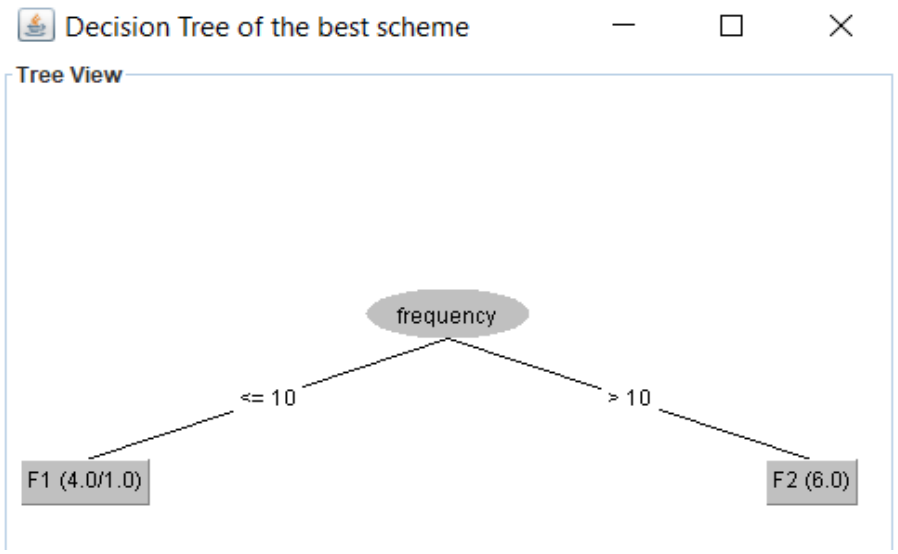


Figura 4.9. Árbol de decisión obtenido por FTree para 2 fragmentos

Se utilizaron otros algoritmos de clasificación para seleccionar el esquema de fragmentación. La Tabla 4.6 presenta los resultados de las métricas de evaluación Precisión (P), Recall (R), F-Medida-F), and Área ROC (ROC) con Naïve Bayes y Multi-layer Perceptron. Con una capa oculta con seis unidades, función de activación sigmoidea, tasa de aprendizaje = 0,3, *momentum* = 0,2 y número de épocas = 500.

Como se ve en la Tabla 4.6, todos los algoritmos coinciden en que el esquema de dos fragmentos es el más adecuado. Además, el modelo obtenido por FTree (Figura 4.9.) es más fácil de interpretar, lo que permite fragmentar el almacén de datos en función del atributo *frequency*.

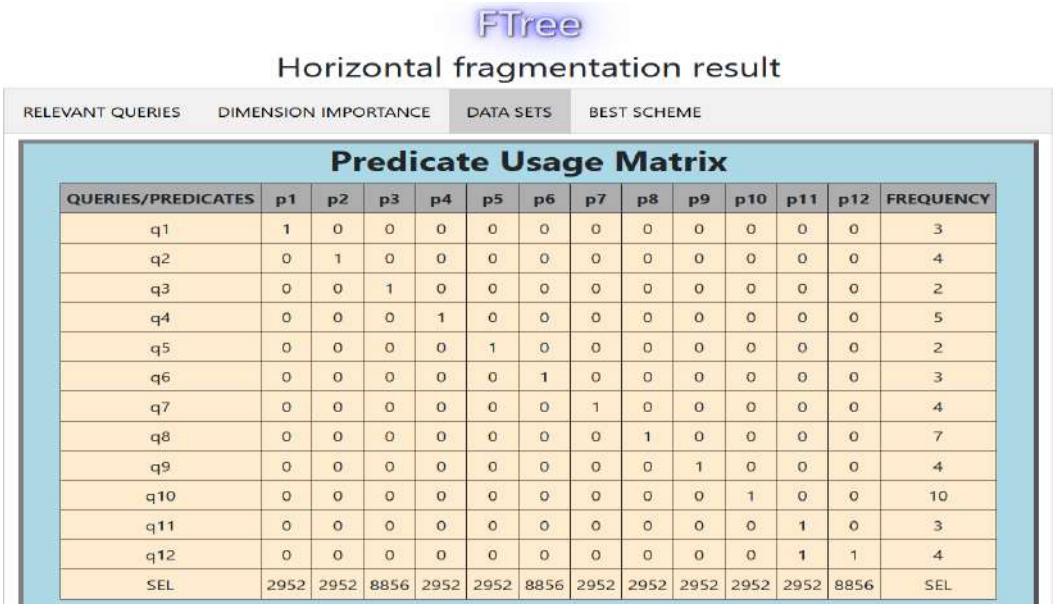
Se realizó un segundo experimento. La Tabla 4.7 presenta las consultas OLAP ejecutadas en el DW turístico. La PUM para esta carga de trabajo se muestra en la Figura 89.

Tabla 4.6. Comparación de los dos esquemas con J48, Naive Bayes y Multi-layer Perceptron

Esquema	J48				Naïve Bayes				Multi-layer Perceptron			
	P	R	F-M	ROC	P	R	F-M	ROC	P	R	F-M	ROC
data_set$_2$	0.925	0.900	0.903	0.881	0.925	0.900	0.903	0.952	0.880	0.800	0.808	0.857
data_set$_3$	-	0.600	-	0.439	-	0.800	-	0.774	0.750	0.600	0.650	0.694

Tabla 4.7. Consultas OLAP ejecutadas en el DW turístico en el segundo experimento

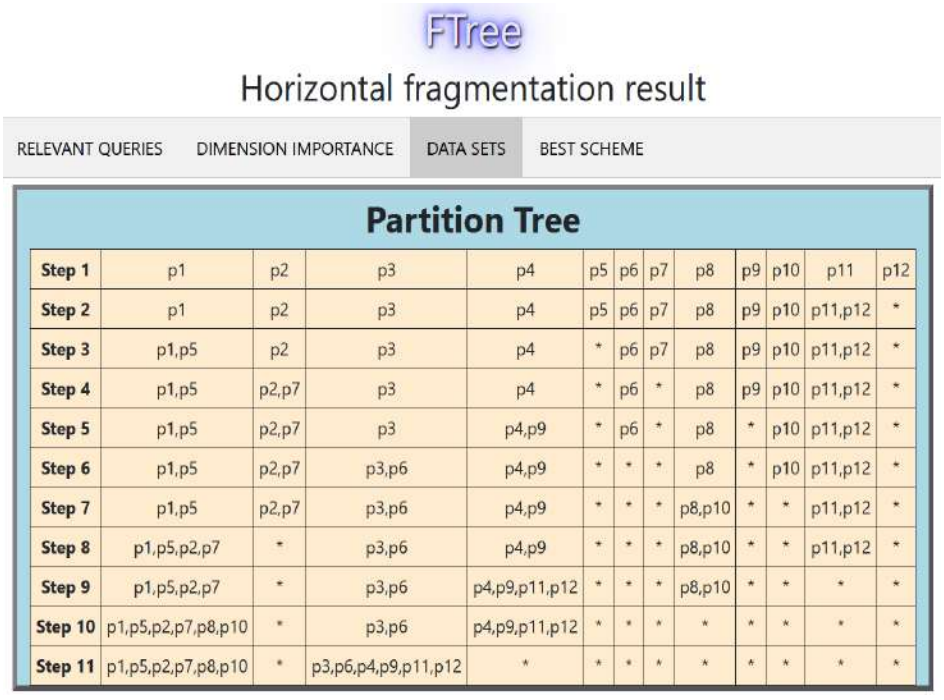
Consulta	Frecuencia
q_1: SELECT mes, sum(cuartos_ocupados) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.año=2007 GROUP BY mes;	3
q_2: SELECT mes, sum(cuartos_ocupados) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.año=2008 GROUP BY mes;	4
q_3: SELECT mes, sum(cuartos_ocupados) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.trimestre='T1' GROUP BY mes;	2
q_4: SELECT mes, sum(turistas_noche) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.año=2009 GROUP BY mes;	5
q_5: SELECT trimestre, sum(turistas_noche) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.año=2010 GROUP BY trimestre;	2
q_6: SELECT año, sum(turistas_noche) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.trimestre='T2' group by año;	3
q_7: SELECT mes, sum(llegada_turistas) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.año=2011 group by mes;	4
q_8: SELECT año, sum(llegada_turistas) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.mes='dic' group by año;	7
q_9: SELECT trimestre, sum(llegada_turistas) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.año=2012 group by trimestre;	4
q_{10}: SELECT año, avg(estadia) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.mes='jul' group by año;	10
q_{11}: SELECT trimestre, avg(estadia) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.año=2013 group by trimestre;	3
q_{12}: SELECT mes, avg(estadia) FROM tiempo, actividad_hotelera WHERE tiempo.id_tiempo=actividad_hotelera.id_tiempo AND tiempo.año=2013 AND tiempo.trimestre='T4' group by mes;	4



QUERIES/PREDICATES	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	FREQUENCY
q1	1	0	0	0	0	0	0	0	0	0	0	0	3
q2	0	1	0	0	0	0	0	0	0	0	0	0	4
q3	0	0	1	0	0	0	0	0	0	0	0	0	2
q4	0	0	0	1	0	0	0	0	0	0	0	0	5
q5	0	0	0	0	1	0	0	0	0	0	0	0	2
q6	0	0	0	0	0	1	0	0	0	0	0	0	3
q7	0	0	0	0	0	0	1	0	0	0	0	0	4
q8	0	0	0	0	0	0	0	1	0	0	0	0	7
q9	0	0	0	0	0	0	0	0	1	0	0	0	4
q10	0	0	0	0	0	0	0	0	0	1	0	0	10
q11	0	0	0	0	0	0	0	0	0	0	1	0	3
q12	0	0	0	0	0	0	0	0	0	0	1	1	4
SEL	2952	2952	8856	2952	2952	8856	2952	2952	2952	2952	2952	8856	SEL

Figura 4.10. PUM para el segundo experimento con el DW turístico

El árbol de partición para este experimento se muestra en la Figura 4.11. En este caso W=4, por lo tanto, se obtuvieron dos conjuntos de datos. La Figura 4.12 compara el rendimiento de estos conjuntos de datos utilizando una validación cruzada de 10 pliegues. El árbol de decisión para el mejor esquema de fragmentación se muestra en la Figura 4.13. Tiene dos fragmentos $fr_1 = \{p_1, p_2, p_5, p_7, p_8, p_{10}\}$, $fr_2 = \{p_3, p_4, p_6, p_9, p_{11}, p_{12}\}$.



Step 1	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12
Step 2	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11,p12	*
Step 3	p1,p5	p2	p3	p4	*	p6	p7	p8	p9	p10	p11,p12	*
Step 4	p1,p5	p2,p7	p3	p4	*	p6	*	p8	p9	p10	p11,p12	*
Step 5	p1,p5	p2,p7	p3	p4,p9	*	p6	*	p8	*	p10	p11,p12	*
Step 6	p1,p5	p2,p7	p3,p6	p4,p9	*	*	*	p8	*	p10	p11,p12	*
Step 7	p1,p5	p2,p7	p3,p6	p4,p9	*	*	*	p8,p10	*	*	p11,p12	*
Step 8	p1,p5,p2,p7	*	p3,p6	p4,p9	*	*	*	p8,p10	*	*	p11,p12	*
Step 9	p1,p5,p2,p7	*	p3,p6	p4,p9,p11,p12	*	*	*	p8,p10	*	*	*	*
Step 10	p1,p5,p2,p7,p8,p10	*	p3,p6	p4,p9,p11,p12	*	*	*	*	*	*	*	*
Step 11	p1,p5,p2,p7,p8,p10	*	p3,p6,p4,p9,p11,p12	*	*	*	*	*	*	*	*	*

Figura 4.11. Árbol de partición para el segundo experimento en el DW turístico

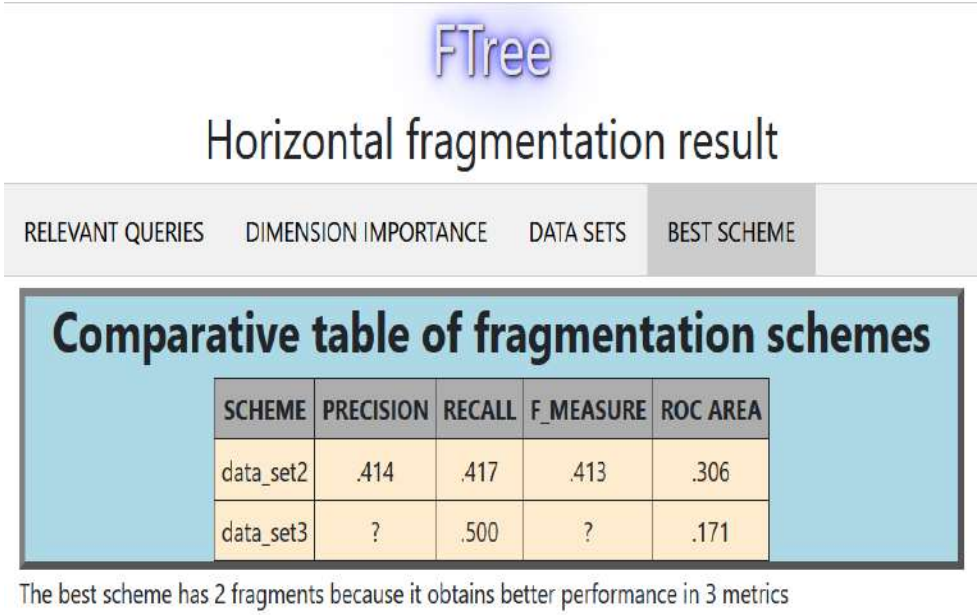


Figura 4.12. Comparación del esquema de fragmentación con 2 y 3 fragmentos

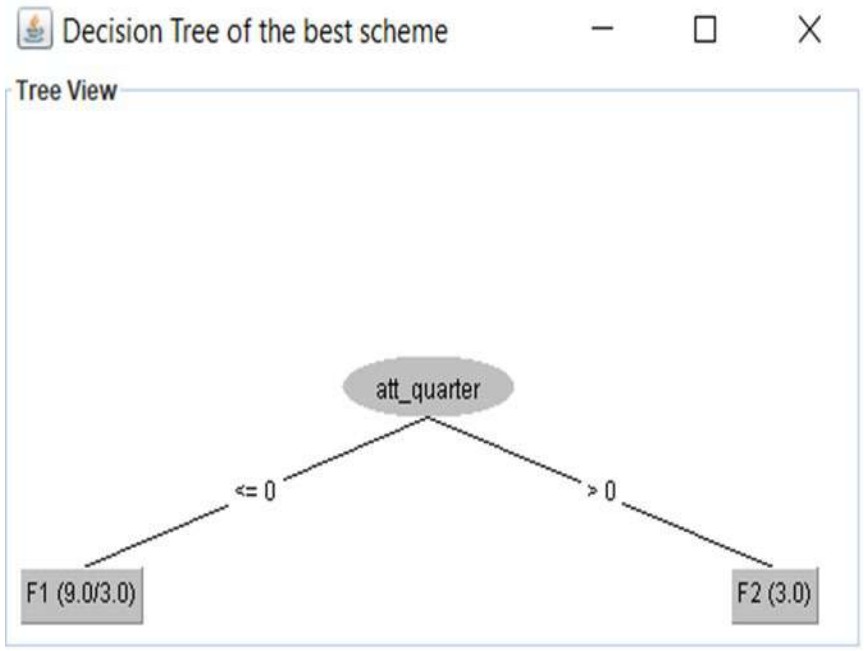


Figura 4.13. Árbol de decisión obtenido por FTree para 2 fragmentos

Se utilizaron otras técnicas de clasificación para seleccionar el esquema de fragmentación. La Tabla 4.8 presenta los resultados de las métricas de evaluación con Naïve Bayes y Multilayer Perceptron. Con una capa oculta con seis unidades, función de activación sigmoidea, tasa de aprendizaje = 0,3, impulso = 0,2 y número de épocas = 500.

Tabla 4.8 Comparación de los dos esquemas con Naïve Bayes and Multi-layer Perceptron.

Esquema	Naïve Bayes				Multi-layer Perceptron			
	Precisión	**Recall**	**Medida-F**	**Área ROC**	**Precisión**	**Recall**	**Medida-F**	**Área ROC**
data_set₂	0.800	0.667	0.625	0.708	**0.688**	**0.667**	**0.657**	0.694
data_set₃	**0.857**	**0.750**	**0.742**	**0.750**	0.563	0.583	0.570	**0.733**

Perceptron multicapa seleccionó el mismo esquema de fragmentación que FTree, mientras que Naïve Bayes eligió el esquema con tres fragmentos. Se compararon los esquemas usando un modelo de costos

La Tabla 4.9 muestra el costo de las consultas en ambos esquemas de fragmentación, el seleccionado por FTree con J48 y Multi-layer Perceptron y el esquema elegido por Naïve Bayes. En el caso del primer esquema, las cardinalidades de los fragmentos son $card(fr_1)$ = 15,744, $card(fr_2)$ = 17,220, $card(fr_3)$=2,460. El tercer fragmento es el complemento. El TIC de q_3 se calculó de la siguiente manera:

$$\begin{aligned} \mathrm{TIC}(q_3) &= (\mathrm{card}(fr_2) - (s_3 - rt_3)) * f_3 \\ &= (17{,}220 - (8{,}856 - 2{,}952)) * 2 = 22{,}632. \end{aligned}$$

El TRC de la misma consulta fue obtenido como se muestra a continuación:

$$\mathrm{TRC}(\mathrm{q}_3) = \mathrm{rt}_3 * \mathrm{f}_3^2 * \mathrm{nf}_3 = 2952 * 2^2 * 2 = 23{,}616.$$

Para el esquema elegido por Naïve Bayes, las cardinalidades del fragmento son $card(fr_1)$ = 15,744, $card(fr_2)$ = 11,808, $card(fr_3)$ = 5,412, $card(fr_4)$ = 2,460. El cuarto fragmento es el complemento. Por lo tanto, los costos de la tercera consulta en este esquema son:

$$\mathrm{TIC}(\mathrm{q}_3) = \left(\mathrm{card}(\mathrm{fr}_2) - (\mathrm{s}_3 - \mathrm{rt}_3)\right) * \mathrm{f}_3 = (11{,}808 - (8{,}856 - 2{,}952)) * 2 = 11{,}808.$$
$$\mathrm{TRC}(\mathrm{q}_3) = \mathrm{rt}_3 * \mathrm{f}_3^2 * \mathrm{nf}_3 = 2{,}952 * 2^2 * 2 = 23{,}616.$$

Tabla 4.9. Comparación del costo de consultas de los esquemas de fragmentación

Consulta	EFH del *data_set₂*			EFH del *data_set₃*		
	TIC	**TRC**	**Costo**	**TIC**	**TRC**	**Costo**
q_1	38,376	0	38,376	38,376	0	38,376

Consulta	EFH del *data_set*$_2$			EFH del *data_set*$_3$		
q_2	51,168	0	51,168	51,168	0	51,168
q_3	22,632	23,616	46,248	11,808	23,616	**35,424**
q_4	73,800	24,600	**98,400**	22,140	147,600	169,740
q_5	25,584	0	25,584	25,584	0	25,584
q_6	33,948	53,136	87,084	17,712	53,136	**70,848**
q_7	51,168	0	51,168	51,168	0	51,168
q_8	89,544	0	89,544	89,544	0	89,544
q_9	59,040	15,744	**74,784**	17,712	94,464	112,176
q_{10}	127,920	0	127,920	127,920	0	127,920
q_{11}	44,280	8,856	**53,136**	13,284	53,136	66,420
q_{12}	43,296	173184	**216,480**	1,968	330,624	332,592

La Tabla 4.9 muestra que el EFH seleccionado por FTree con J48 y Multi-layer Perceptron es mejor en la mayoría de los casos. En solo dos consultas, el HFS elegido por Naïve Bayes obtuvo un menor costo de ejecución de consultas. Además, el árbol de decisión generado por J48 es más fácil de interpretar que el modelo de perceptrón multicapa. Por tanto, el modelo obtenido por J48 superó a los de NaiveBayes y Multi-layer Perceptron. Además para evaluar FTree, este fue comparado con (Kechar & Nait-Bahloul, 2017), para esto se usó la PUM de la Tabla 4.10.

Tabla 4.10. PUM para el tercer experimento

Q/P	p_1	p_2	p_3	p_4	p_5	p_6	p_7	F_i
q_1	0	1	0	0	1	0	1	20
q_2	1	1	0	0	0	1	0	15
q_3	0	0	1	0	1	0	1	20
q_4	0	1	0	1	0	1	0	15
q_5	0	0	1	0	1	0	1	15
q_6	1	0	0	1	0	1	1	10
q_7	0	1	0	1	0	1	0	15
sel_j	30	50	80	65	20	35	40	

El mejor esquema de fragmentación según (Kechar & Nait-Bahloul, 2017) es el de tres fragmentos $fr_1 = \{p_1, p_2, p_3, p_4, p_5\}$, $fr_2 = \{p_6\}$, $fr_3 = \{p_7\}$. Con FTree, en este caso, la cantidad máxima de fragmentos permitidos por el administrador del almacén de datos (W)

es 4. Usando la PUM, se obtuvo el árbol de particiones generado por FTree. La Figura 4.14 muestra la comparación entre *data_set*$_2$ con los fragmentos $fr_1 = \{p_1, p_2, p_4, p_6\}, fr_2 = \{p_3, p_5, p_7\}$, y *data_set*$_3$ con los fragmentos $fr_1 = \{p_1, p_2, p_4\}, fr_2 = \{p_3, p_5, p_7\}, fr_3 = \{p_6\}$. Se utilizó una validación cruzada de 8 pliegues. El mejor esquema es *data_set*$_2$. El árbol de decisión obtenido con FTree se visualiza en la Figura 4.15.

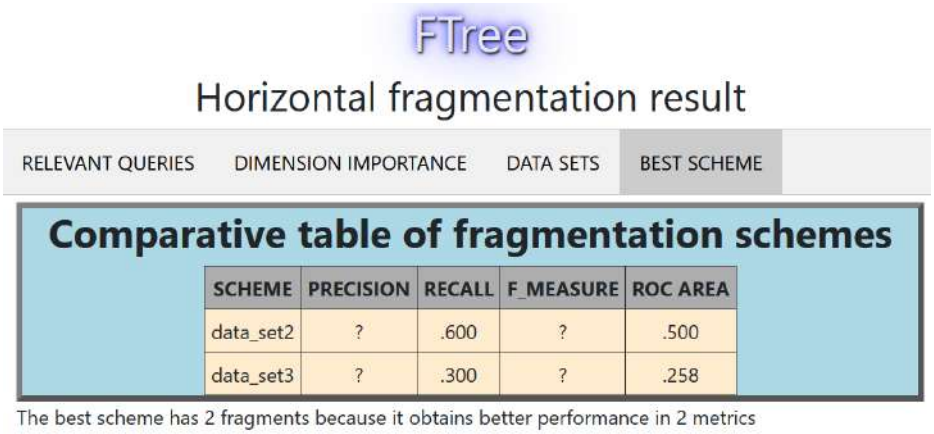


Figura 4.14. Comparación de los esquemas de fragmentación con 2 y 3 fragmentos para el tercer experimento

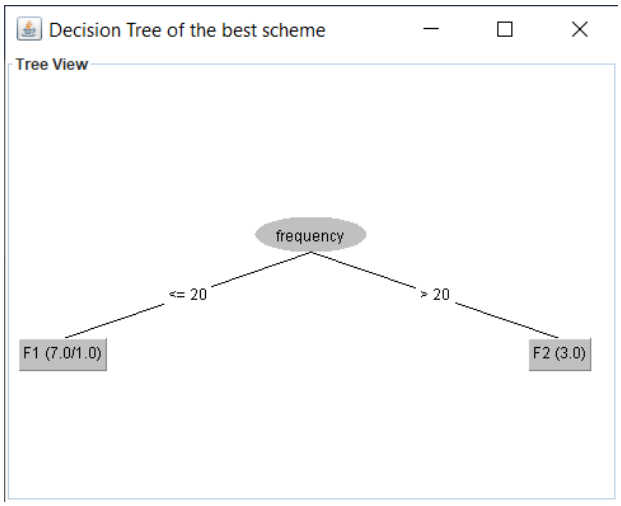


Figura 4.15.Árbol de decision para 2 fragmentos encontrado por FTree en el tercer experimento

La Tabla 4.11 muestra el costo de las consultas en los dos esquemas de fragmentación, el obtenido por (Kechar & Nait-Bahloul, 2017) y el de FTree. En el caso del esquema generado por FTree, las cardinalidades del primer y segundo fragmento son $card(fr_1) = 180$, $card(fr_2) = 140$. Para el esquema producido por (Kechar & Nait-Bahloul, 2017), las cardinalidades de los fragmentos son $card\ (fr1) = 245$, $card(fr_2) = 35$, $card(fr_3) = 40$.

Tabla 4.11. Comparación de costos de consulta para los dos esquemas de fragmentación

Consulta	FTree			Kechar & Nait-Bahloul		
	ITC	**RTC**	**Cost**	**ITC**	**RTC**	**Cost**
q_1	4200	0	4200	3500	0	**3500**
q_2	975	0	**975**	2475	0	2475
q_3	0	0	**0**	2900	0	2900
q_4	450	0	**450**	1950	0	1950
q_5	0	0	**0**	2175	0	2175
q_6	1500	0	1500	1500	0	1500
q_7	450	0	**450**	1950	0	1950

Como se observa en la Tabla 4.11, aunque ambos métodos de fragmentación obtuvieron esquemas sin TRC, el esquema generado por FTree tiene menor TIC en la mayoría de los casos. Solo la primera consulta fue más eficiente en el esquema encontrado por (Kechar & Nait-Bahloul, 2017). Estos experimentos demostraron la efectividad de FTree.

CONCLUSIONES

Los avances tecnológicos han hecho posible recopilar grandes cantidades de datos en diferentes campos; por lo tanto, existe una necesidad constante de desarrollar herramientas que ayuden al proceso de extraer información precisa de la enorme cantidad de datos que se convierten en conocimiento. Los DW han demostrado ser eficientes para el análisis de estas grandes cantidades de datos, por lo que son una excelente opción para implementar en el seguimiento de la actividad turística, en la que, para mantenerse en un nivel competitivo, es necesario conocer las preferencias del cliente. Como se mencionó a lo largo de este proyecto de tesis, la fragmentación en un DW permite optimizar el tiempo de respuesta y los costos de ejecución de las consultas OLAP, se han desarrollado varias técnicas para la fragmentación tanto vertical como horizontal, sin embargo, hasta donde se sabe, no se ha utilizado un árbol de decisión para fragmentar el DW, lo que representa un área inexplorada dentro de la investigación DW. En este trabajo se aprovechó la capacidad de los árboles de decisión en la clasificación para adaptarlos al proceso de fragmentación horizontal del DW. Se desarrolló un método de fragmentación horizontal, denominado FTree, que utiliza árboles de decisión para obtener esquemas de fragmentación que logran la optimización de consultas OLAP en un DW. Se aplicó FTree en un DW turístico, aprovechando la gran cantidad de datos disponibles en este ámbito y la construcción del mismo durante la duración del proyecto de tesis. Los resultados obtenidos benefician tanto a diseñadores y usuarios de almacenes de datos en cualquier ámbito y área profesional como particularmente a investigadores del área computacional.

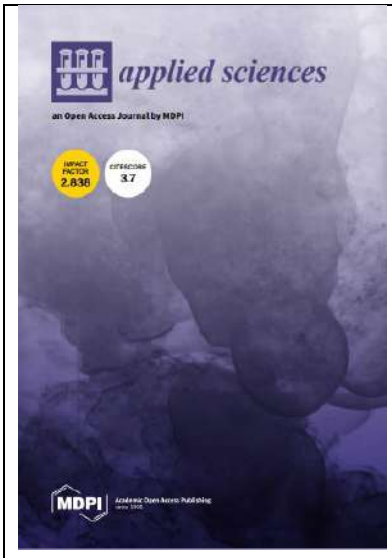
RECOMENDACIONES

En investigaciones futuras, se extenderá FTree para proporcionar fragmentación dinámica de almacenes de datos, monitoreando los patrones de acceso del DW para detectar cuándo modificar el esquema de fragmentación horizontal para evitar la reducción del rendimiento de las consultas.

Por otro lado, es de suma importancia considerar aplicar la fragmentación en otro tipo de tecnologías en crecimiento como los lagos de datos (*Data Lakes*). Aunado a esto, el utilizar otras técnicas de clasificación para comparar los esquemas de fragmentación seleccionados por FTree conllevará a enriquecer esta investigación.

PRODUCTOS ACADÉMICOS

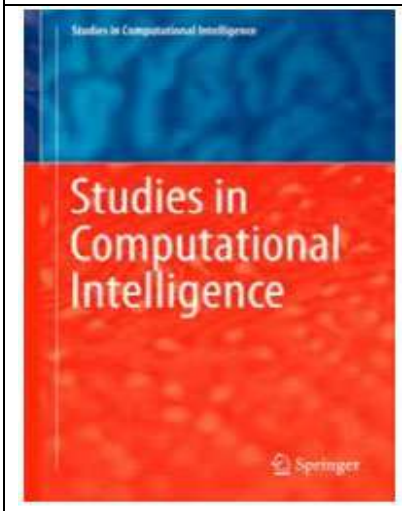
Revistas Indizadas- Journal Citation Reports

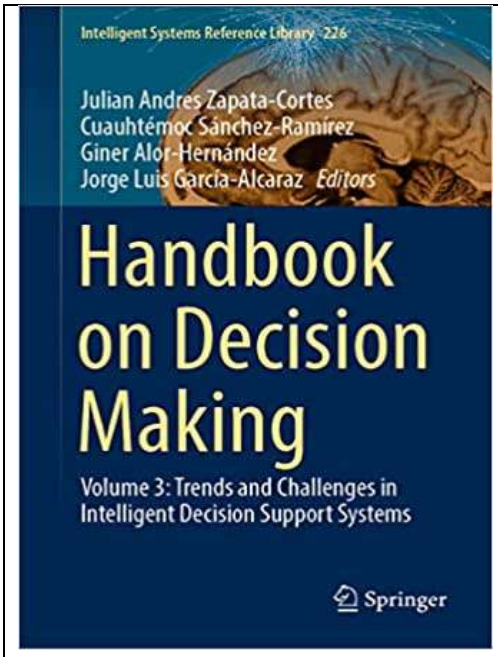
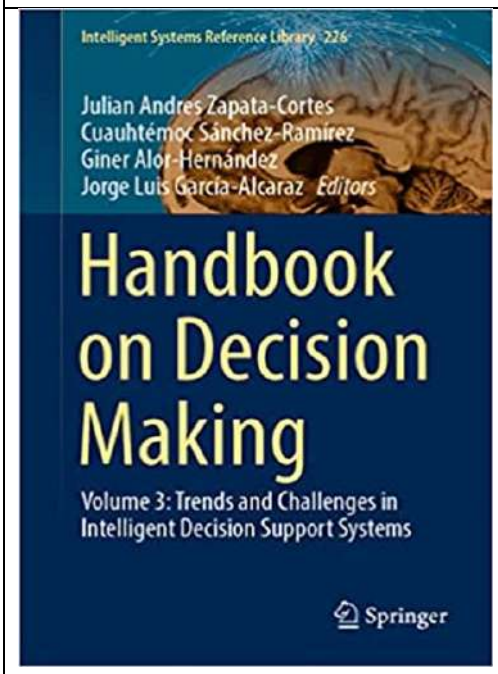
	**Rodríguez-Mazahua N, Rodríguez-Mazahua L, López-Chau A, Alor-Hernández G, Machorro-Cano I. Decision-Tree-Based Horizontal Fragmentation Method for Data Warehouses. *Applied Sciences*. 2022; 12(21):10942.** **https://doi.org/10.3390/app122110942** **Estado: Publicado**

Artículos en congreso

	**Como Primer Autor** **Nidia Rodríguez-Mazahua; Lisbeth Rodríguez-Mazahua; Asdrúbal López-Chau; Giner Alor-Hernández; Gustavo Silverio Peláez-Camarena. (2020). Comparative Analysis of Decision Tree Algorithms for Data Warehouse Fragmentation. Revista científica arbitrada de Educación empresarial. Revista Perspectiva Empresarial** **ISSN-e: 2389-8194** DOI: https://doi.org/10.16967/rpe **Estado: Publicado**

Capítulos de libros

	**Como primer Autor** **Nidia Rodríguez Mazahua; Lisbeth Rodríguez Mazahua; Asdrúbal López Chau and Giner Alor Hernández. (2020). Horizontal Fragmentation of Data Warehouses Using Decision Trees. María Lucia Barrón Estrada, ´ José Antonio Camarena Ibarrola, ´ Karina Mariela Figueroa Mora, Valeria Soto Mendoza (eds). Aplicaciones de la Computación. ISBN: 978-607-506-395-9 pág. 276-280.** **Estado: Publicado**
	**Como Primer Autor** **Nidia Rodríguez Mazahua; Lisbeth Rodríguez Mazahua; Asdrúbal López Chau; Giner Alor Hernández; Gustavo Silverio Peláez- Camarena. (2020). Comparative Analysis of Decision Tree Algorithms for Data Warehouse Fragmentation. (2020)Julian Andres Zapata-Cortes, Giner Alor- Hernández, Cuauhtémoc Sánchez-Ramírez, Jorge Luis García-Alcaraz (eds). New Perspectives on Enterprise Decision – Making Applying Artificial Intelligence Springer Verlag in the Studies in Computational Intelligence series (https://www.springer.com/series/7092) ISSN: 1860-949X.** **Estado: Publicado**

	**Ortiz-Ballona, A.O., Rodríguez-Mazahua, L., López-Chau, A., Castro-Medina, F., Abud-Figueroa, M.A., Rodríguez-Mazahua, N. (2023). A Vertical Fragmentation Method for Multimedia Databases Considering Content-Based Queries. In: Zapata-Cortes, J.A., Sánchez-Ramírez, C., Alor-Hernández, G., García-Alcaraz, J.L. (eds) Handbook on Decision Making. Intelligent Systems Reference Library, vol 226. Springer, Cham. https://doi.org/10.1007/978-3-031-08246-7_1** **Estado: Publicado.**
	**Castillo-García, A., Rodríguez-Mazahua, L., Castro-Medina, F., Arrioja-Rodríguez, M.L., Sánchez-Cervantes, J.L., Rodríguez-Mazahua, N. (2023). Design of a Dynamic Horizontal Fragmentation Method for Multimedia Databases. In: Zapata-Cortes, J.A., Sánchez-Ramírez, C., Alor-Hernández, G., García-Alcaraz, J.L. (eds) Handbook on Decision Making. Intelligent Systems Reference Library, vol 226. Springer, Cham. https://doi.org/10.1007/978-3-031-08246-7_4** **Estado: Publicado.**

REFERENCIAS BIBLIOGRÁFICAS

Amina, G., & Boukhalfa, K. (2013). Very Large Workloads Based Approach to Efficiently Partition Data Warehouses. En A. Amine, A. M. Otmane, & L. Bellatreche (Eds.), *Modeling Approaches and Algorithms for Advanced Computer Applications* (Vol. 488, pp. 285–294). Springer International Publishing. https://doi.org/10.1007/978-3-319-00560-7_32

Amirat, H., & Boukhalfa, K. (2014). A data mining-based approach for data warehouse optimisation. *2émes journées internationales de chimie organométallique et catalyse jicoc'2014*, 6.

Anitha, S., & Vanitha, M. (2022). Classification of VASA Dataset Using J48, Random Forest, and Naive Bayes. En S. C. Satapathy (Ed.), *Intelligent Data Engineering and Analytics. Smart Innovation, Systems, and Tecnnologies* (Vol. 266). Springer. https://doi.org/10.1007/978-981-16-6624-7_28

Awad, M., & Jebreen, I. (2015). Interoperable Distributed Data Warehouse Components. *International Journal of Computer Science Issues*, *12*(2), 10.

Barkhordari, M., & Niamanesh, M. (2018). Chabok: A Map-Reduce based method to solve data warehouse problems. *Journal of Big Data*, *5*(40), 1–25. https://doi.org/10.1186/s40537-018-0144-5

Barr, M. (2012). Self—Monitoring for the horizontal fragmentation evolution based on ants in the Relational datawarehouses. *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, *1*, 538–541.

Barr, M. (2013). Bi-Objective Optimization Based on Compromise Method for Horizontal Fragmentation in Relational Data Warehouses. *International Journal of Machine Learning and Computing*, *3*(3), 250–254. https://doi.org/10.7763/IJMLC.2013.V3.313

Barr, M., & Bellatreche, L. (2010). A new approach based on ants for solving the problem of horizontal fragmentation in relational data warehouses. *2010 International Conference on Machine and Web Intelligence*, 411–415. https://doi.org/10.1109/ICMWI.2010.5648104

Barr, M., Boukhalfa, K., & Bouibede, K. (2018). Bi-Objective Optimization Method for Horizontal Fragmentation Problem in Relational Data Warehouses as a Linear Programming Problem. *Applied Artificial Intelligence*, *32*(9–10), 907–923. https://doi.org/10.1080/08839514.2018.1519096

Bellatreche, L., & Benkrid, S. (2009). A Joint Design Approach of Partitioning and Allocation in Parallel Data Warehouses. En T. B. Pedersen, M. K. Mohania, & A. M. Tjoa (Eds.), *Data Warehousing and Knowledge Discovery* (Vol. 5691, pp. 99–110). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-03730-6_9

Bellatreche, L., Bouchakri, R., Cuzzocrea, A., & Maabout, S. (2013). Horizontal Partitioning of Very-large Data Warehouses Under Dynamically-changing Query Workloads via Incremental Algorithms. *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 208–210. https://doi.org/10.1145/2480362.2480406

Bellatreche, L., & Boukhalfa, K. (2005). An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse. En A. M. Tjoa & J. Trujillo (Eds.), *Data Warehousing and Knowledge Discovery* (Vol. 3589, pp. 115–125). Springer Berlin Heidelberg. https://doi.org/10.1007/11546849_12

Bellatreche, L., Boukhalfa, K., & Abdalla, H. I. (2006). SAGA: A Combination of Genetic and Simulated Annealing Algorithms for Physical Data Warehouse Design. En D. A. Bell & J. Hong (Eds.), *Flexible and Efficient Information Handling* (pp. 212–219). Springer Berlin Heidelberg.

Bellatreche, L., Boukhalfa, K., & Richard, P. (2008). Data Partitioning in Data Warehouses: Hardness Study, Heuristics and ORACLE Validation. En I.-Y. Song, J. Eder, & T. M. Nguyen (Eds.), *Data Warehousing and Knowledge Discovery* (pp. 87–96). Springer Berlin Heidelberg.

Bellatreche, L., Karlapalem, K., Mohania, M., & Schneider, M. (2000). What can partitioning do for your data warehouses and data marts? *Proceedings 2000 International Database Engineering and Applications Symposium (Cat. No.PR00789)*, 437–445. https://doi.org/10.1109/IDEAS.2000.880634

Bellatreche, L., & Woameno, K. Y. (2009). Dimension Table Driven Approach to Referential Partition Relational Data Warehouses. *Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP*, 9–16. https://doi.org/10.1145/1651291.1651294

Benkrid, S., Bellatreche, L., & Drias, H. (2008). A Combined Selection of Fragmentation and Allocation Schemes in Parallel Data Warehouses. *2008 19th International Conference on Database and Expert Systems Applications*, 370–374. https://doi.org/10.1109/DEXA.2008.63

Bilal, M., Sheikh, U., Raza, B., & Javaid, Q. (2016). Application of Data Warehouse in Real Life: State-of-the-art Survey from User Preferences' Perspective. *International Journal of Advanced Computer Science and Applications*, *7*(4). https://doi.org/10.14569/IJACSA.2016.070455

Boissier, M., & Kurzynski, D. (2018). Workload-Driven Horizontal Partitioning and Pruning for Large HTAP Systems. *2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW)*, 116–121. https://doi.org/10.1109/ICDEW.2018.00026

Bouchakri, R., Bellatreche, L., & Faget, Z. (2014). Algebra-Based Approach for Incremental Data Warehouse Partitioning. En H. Decker, L. Lhotská, S. Link, M. Spies, & R. R. Wagner (Eds.),

Database and Expert Systems Applications (Vol. 8645, pp. 441–448). Springer International Publishing. https://doi.org/10.1007/978-3-319-10085-2_40

Bouchakri, R., Bellatreche, L., Faget, Z., & Breß, S. (2014). A coding template for handling static and incremental horizontal partitioning in data warehouses. *Journal of Decision Systems*, *23*(4), 481–498. https://doi.org/10.1080/12460125.2014.934123

Brki, L. (2012). Data Quality Improvement through Horizontal Fragmentation in Data Warehouses. *International Journal of Mathematical Models and Methods in Applied Sciences*, *6*(5), 634–642.

Costa, M. R., Lopes, S. T. L., Cesáreo, T. V., Rodrigues, C. R., & de Aguiar, C. C. D. (2016). Spatial data warehouses and spatial OLAP come towards the cloud: Design and performance. *Distributed and Parallel Databases*, *34*(3), 425–461. https://doi.org/10.1007/s10619-015-7176-z

Curino, C., Jones, E., Zhang, Y., & Madden, S. (2010). Schism: A workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment*, *3*(1–2), 48–57. https://doi.org/10.14778/1920841.1920853

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. https://doi.org/10.1504/IJBIDM.2009.029076

Cuzzocrea, A., & Moussa, R. (2013). Multidimensional Database Design via Schema Transformation: Turning TPC-H into the TPC-H*d Multidimensional Benchmark. *The 19th International Conference on Management of Data (COMAD)*, 12.

Daniel, C., Salamanca, E., & Nordlinger, B. (2020). Hospital Databases: AP-HP Clinical Data Warehouse. En *Healthcare and Artificial Intelligence* (pp. 57–67). Springer. https://doi.org/10.1007/978-3-030-32161-1_8

Danubianu, M., Socaciu, C., & Barila, C. (2009). *Some Aspects of Data Warehousing in Tourism Industry*. *9*, 7.

de Aguiar, C., C. D., Rodrigues, C. R., Tuler, T., Diogo, Machado, T., Agma ,. Juci, & da Fonseca, de S., Fernando. (2007). Horizontal fragmentation as a technique to improve the performance of drill-down and roll-up queries. *Proceedings of the 2007 ACM symposium on Applied computing - SAC '07*, 494. https://doi.org/10.1145/1244002.1244117

Dean, J. (2014). *Big Data , Data Mining, and Machine Learning Value Creation for Business Leaders and Practitioners*. John Wiley & Sons, Inc.

Dimovski, A., Velinov, G., & Sahpaski, D. (2010). Horizontal Partitioning by Predicate Abstraction and Its Application to Data Warehouse Design. En *Advances in Databases and Information Systems* (Vol. 6295, pp. 164–175). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-15576-5_14

Documentation—Weka Wiki. (2020). https://waikato.github.io/weka-wiki/documentation/

Elmansouri, R., Elbeqqali, O., & Ziyati, E. (2013). Normed principal components analysis: A new approach to data warehouse fragmentation. *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, 1–4. https://doi.org/10.1109/AICCSA.2013.6616465

Ettaoufik, A., & Ouzzif, M. (2014). Query's optimization in data warehouse on the cloud using fragmentation. *2014 International Conference on Next Generation Networks and Services (NGNS)*, 145–148. https://doi.org/10.1109/NGNS.2014.6990243

Ettaoufik, A., & Ouzzif, M. (2017a). Web Service for Incremental and Automatic Data Warehouses Fragmentation. *International Journal of Advanced Computer Science and Applications*, *8*(6), 1–10. https://doi.org/10.14569/IJACSA.2017.080661

Ettaoufik, A., & Ouzzif, M. (2017b). Web service for incremental data warehouses fragmentation assisted with temporary materialized views. *Journal of Theoretical and Applied Information Technology*, *95*(14), 3217–3229.

Ezeife, C. I. (2001). Selecting and materializing horizontally partitioned warehouse views. *Data & Knowledge Engineering*, *36*(2), 185–210.

Furtado, P. (2004). Experimental Evidence on Partitioning in Parallel Data Warehouses. *Proceedings of the 7th ACM international workshop on Data Warehousing and OLAP*, 23–30.

Gopalkrishnan, Qing Li, & Kamalakar Karlapalem. (2000). Efficient Query Processing with Associated Horizontal Class Partitioning in an Object Relational Data Warehousing Environment. *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000*, 1–9.

Gopalkrishnan, V., Li, Q., & Karlapalem, K. (2001). Semantic Query Optimization based on Class Partitioning Techniques in an Object Relational Data Warehousing Environment. *IJIT*, *7*(2), 1–22.

Hamdi, I., Bouazizi, E., Alshomrani, S., & Feki, J. (2015). 2LPA-RTDW: A Two-Level data Partitioning Approach for Real-time Data Warehouse. *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, 632–638. https://doi.org/10.1109/ICIS.2015.7166669

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques* (Third edition). Morgan Kaufmann Publishers.

Hanane, A., & Kamel, B. (2014). A data mining-based approach for data warehouse optimisation. *ICA2IT International Conference on Artificial Intelligence and Information Technology.*, 1–6.

Hilprecht, B., Binnig, C., & Röhm, U. (2019a). Learning a Partitioning Advisor with Deep Reinforcement Learning. *arXiv:1904.01279*.

Hilprecht, B., Binnig, C., & Röhm, U. (2019b). Towards learning a partitioning advisor with deep reinforcement learning. *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, 1–4. https://doi.org/10.1145/3329859.3329876

Hilprecht, B., Binnig, C., & Röhm, U. (2020). Learning a Partitioning Advisor for Cloud Databases. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 143–157. https://doi.org/10.1145/3318464.3389704

Höpken, W., Fuchs, M., Keil, D., & Lexhagen, M. (2015). Business intelligence for cross-process knowledge extraction at tourism destinations. *Information Technology & Tourism*, *15*(2), 101–130. https://doi.org/10.1007/s40558-015-0023-2

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, 97–106. https://doi.org/10.1145/502512.502529

Janzen, T. J., & Ristino, L. (2018). *USDA and Agriculture Data: Improving Productivity while Protecting Privacy*. Agree.

Karima, T., Abdellatif, A., & Ounalli, H. (2010). Data mining based fragmentation technique for distributed data warehouses environment Using predicate construction technique. *The 6th International Conference on Networked Computing and Advanced Information Management*, 63–68.

Kechar, M., & Nait-Bahloul, S. (2017). Performance optimisation of the decision-support queries by the horizontal fragmentation of the data warehouse. *International Journal of Business Information Systems*, *26*(4), 506–537. https://doi.org/10.1504/IJBIS.2017.087750

Kechar, M., & Nait-Bahloul, S. (2019). Bringing Together Physical Design and Fast Querying of Large Data Warehouses: A New Data Partitioning Strategy. *BDIoT'19: Proceedings of the 4th International Conference on Big Data and Internet of Things*.

Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The definitive Guide to Dimensional Modeling* (Third Edition). John Wiley & Sons, Inc.

Kimball, R., & Ross, M. (2016). *The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence* (Second Edition). John Wiley & Sons, Inc.

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. (2008). *The Data Warehouse Lifecycle Toolkit.* (Second edition). Wiley Publishing,Inc.

Kotsiantis, S., Tsekouras, G., & Pintelas, P. (2005). *Local Bagging of Decision Stumps*. 377–391. https://doi.org/10.1007/11504894_57

Letrache, K., El Beggar, O., & Ramdani, M. (2019). OLAP cube partitioning based on association rules method. *Applied Intelligence*, *49*(2), 420–434. https://doi.org/10.1007/s10489-018-1275-2

Liu, J. Y. C., Wang, C. W., & Chan, C. Y. (2013). An Efficient Partitioning for Object-Relational Data Warehouses. *Applied Mechanics and Materials*, *284–287*, 3320–3324. https://doi.org/10.4028/www.scientific.net/AMM.284-287.3320

Mahboubi, H., & Darmont, J. (2008). Data mining-based fragmentation of XML data warehouses. *Proceeding of the ACM 11th international workshop on Data warehousing and OLAP - DOLAP '08*, 9. https://doi.org/10.1145/1458432.1458435

Mahboubi, H., & Darmont, J. (2009). Enhancing XML data warehouse query performance by fragmentation. *Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09*, 1555–1562. https://doi.org/10.1145/1529282.1529630

Mateus, R. C., Siqueira, T. L. L., Times, V. C., Ciferri, R. R., & de Aguiar Ciferri, C. D. (2016). Spatial data warehouses and spatial OLAP come towards the cloud: Design and performance. *Distributed and Parallel Databases*, *34*(3), 425–461. https://doi.org/10.1007/s10619-015-7176-z

Melton, J. E., Go, S., Zilliac, G. G., & Zhang, B. Z. (2022). *Greeenhouse Gas Emission Estimations for 2016-2020 using the Sherlock Air Traffic Data Warehouse,*. Report NASA/TM-202220007609.

Nam, Y.-M., Han, D., & Kim, M.-S. (2019). A parallel query processing system based on graph-based database partitioning. *Information Sciences*, *480*, 237–260. https://doi.org/10.1016/j.ins.2018.12.031

Nam, Y.-M., Kim, M.-S., & Han, D. (2018). A Graph-Based Database Partitioning Method for Parallel OLAP Query Processing. *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 1025–1036. https://doi.org/10.1109/ICDE.2018.00096

Noaman, A. Y., & Barker, K. (1999). A Horizontal Fragmentation Algorithm for the Fact Relation in a Distributed Data Warehouse. *CIKM 99*, 154–161.

OLAP Benchmark Study. (2020). OLAPcouncil.org. http://www.olapcouncil.org/research/spec1.htm

OMT. (2022). *¿Por qué el Turismo? | OMT*. https://www.unwto.org/es/turismo

O'Neil, P., O'Neil, B., & Chen, X. (2007). *The Star Schema Benchmark (SSB)*. 10.

O'neil, P., O'neil, B., & Chen, X. (2009). *The Star Schema Benchmark (SSB)*.

Ozsu, M. T., & Valduriez, P. (2020). *Principles of Distributed Database Systems* (Fourth edition). Springer Nature Switzerland AG.

Parchas, P., Naamad, Y., Bouwel, P. V., Faloutsos, C., & Petropoulos, M. (2020). Fast and effective distribution-key recommendation for amazon redshift. *Proc. VLDB Endow.* https://doi.org/10.14778/3407790.3407834

Provost, F., & Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking.* O'Reilly Media Inc.

Rafid, Y. (2013). Using Dimensional Hierarchy Analysis in Data Warehouse Fragmentation Process. *Journal of the College of Basic Education*, *19*(80), 709–723.

Ramdane, Y., Boussaid, O., Kabachi, N., & Bentayeb, F. (2018). Partitioning and Bucketing Techniques to Speed up Query Processing in Spark-SQL. *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 142–151. https://doi.org/10.1109/PADSW.2018.8644891

Ramdane, Y., Kabachi, N., Boussaid, O., & Bentayeb, F. (2019a). SDWP: A New Data Placement Strategy for Distributed Big Data Warehouses in Hadoop. En C. Ordonez, I.-Y. Song, G. Anderst-Kotsis, A. M. Tjoa, & I. Khalil (Eds.), *Big Data Analytics and Knowledge Discovery* (pp. 189–205). Springer International Publishing. https://doi.org/10.1007/978-3-030-27520-4_14

Ramdane, Y., Kabachi, N., Boussaid, O., & Bentayeb, F. (2019b). SkipSJoin: A New Physical Design for Distributed Big Data Warehouses in Hadoop. En A. H. F. Laender, B. Pernici, E.-P. Lim, & J. P. M. de Oliveira (Eds.), *Conceptual Modeling* (pp. 255–263). Springer International Publishing. https://doi.org/10.1007/978-3-030-33223-5_21

Razdan, S., Gupta, H., & Seth, A. (2021). Performance Analysis of Network Intrusion Systems using J48 and Naive Bayes Algorithm. *6th International Conference for Convergence in Technology (I2CT)*, 1–7. https://doi.org/10.1109/I2CT51068.2021.9417971.

Rodriguez, L. (2012). *Fragmentación Vertical Dinámica de Bases de Datos Multimedia Usando Reglas Activas*. CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL.

Rodríguez, L., Alor-Hernández, G., Abud-Figueroa, Ma. A., & Peláez-Camarena, S. G. (2014). Horizontal Partitioning of Multimedia Databases Using Hierarchical Agglomerative Clustering. *Mexican International Conference on Artificial Intelligence,MICAI 2014: Nature-Inspired Computation and Machine Learning*, *8857*, 296–309.

Rodríguez-Mazahua, L., Alor-Hernández, G., Abud-Figueroa, Ma. A., & Peláez-Camarena, S. G. (2014). Horizontal Partitioning of Multimedia Databases Using Hierarchical Agglomerative Clustering. En A. Gelbukh, F. C. Espinoza, & S. N. Galicia-Haro (Eds.), *Nature-Inspired Computation and Machine Learning* (pp. 296–309). Springer International Publishing. https://doi.org/10.1007/978-3-319-13650-9_27

Rodríguez-Mazahua, N., Rodríguez-Mazahua, L., López-Chau, A., Alor-Hernández, G., & Peláez-Camarena, S. G. (2021). Comparative Analysis of Decision Tree Algorithms for Data Warehouse Fragmentation. En J. A. Zapata-Cortes, G. Alor-Hernández, C. Sánchez-Ramírez, & J. L. García-Alcaraz (Eds.), *New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques* (Vol. 966, pp. 337–363). Springer International Publishing. https://doi.org/10.1007/978-3-030-71115-3_15

Saeh, I. S., Mustafa, M. W., Mohammed, Y. S., & Almaktar, M. (2016). Static Security classification and Evaluation classifier design in electric power grid with presence of PV power plants using C-4.5. *Renewable and Sustainable Energy Reviews*, *56*, 283–290. https://doi.org/10.1016/j.rser.2015.11.054

Sarka, D., Lah, M., & Jerkic, G. (2014). *Implementing a Data Warehouse with Microsoft SQL Server 2012*. Ó Reilly Media, Inc.

Shi, L., Duan, Q., Dong, P., Xi, L., & Ma, X. (2018). Signal prediction based on boosting and decision stump. *International Journal of Computational Science and Engineering*, *16*(2), 117–122. https://doi.org/10.1504/IJCSE.2018.090450

Sidi, E., El, M., & Amin, E. (2016). Star Schema Advantages on Data Warehouse: Using Bitmap Index and Partitioned Fact Tables. *International Journal of Computer Applications*, *134*(13), 11–13. https://doi.org/10.5120/ijca2016908108

Son, J. H., & Kim, M. H. (2004a). An adaptable vertical partitioning method in distributed systems. *J. Syst. Softw*, *73*, 551–561.

Son, J. H., & Kim, M. H. (2004b). An adaptable vertical partitioning method in distributed systems. *Journal of Systems and Software*, *73*(3), 551–561.

Sun, L., Franklin, M. J., Krishnan, S., & Xin, R. S. (2014). Fine-grained partitioning for aggressive data skipping. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, 1115–1126. https://doi.org/10.1145/2588555.2610515

Toumi, L., Moussaoui, A., & Ugur, A. (2015). EMeD-Part: An Efficient Methodology for Horizontal Partitioning in Data Warehouses. *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication - IPAC '15*, 1–7. https://doi.org/10.1145/2816839.2816876

TPC-DS Homepage. (2020). http://www.tpc.org/tpcds/

TPC-H Homepage. (2020). http://www.tpc.org/tpch/

Valentikova, E., Lieskovsky, A., & Zabovsky, M. (2012). Fragment pattern for data warehouse. *2012 IEEE International Conference on Computer Science and Automation Engineering*, 284–287. https://doi.org/10.1109/ICSESS.2012.6269461

Ward, J. H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, *58*(301), 236–244. https://doi.org/10.1080/01621459.1963.10500845

Weka sourceforge. (2022). *GreedyStepwise*. https://weka.sourceforge.io/doc.dev/weka/attributeSelection/GreedyStepwise.html

Witten, I. H., Frank, E., & Hall, M. (2011). *Data Mining Practical Machine Learning Tools and Techniques* (Third Edition). Elsevier.

Yeruva, S., Kumar, P. V., & Padmanabham, P. (2015). Design of distributed warehouse-a vertical fragmentation approach. *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, 616–621. https://doi.org/10.1109/NGCT.2015.7375195