REVISTA DE LA ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN
UNIVERSIDAD NACIONAL DE SAN AGUSTIN DE AREQUIPA
VOL. 1, NO. 1, DICIEMBRE 2024

Trabajo Final EDA: PH-Tree

Joao Franco Emanuel Chávez Salas[1], Diego Zanabria Sacsi[2], Josué Samuel Philco Puma[3] y Fernando David Deza Sotomayor[4]

[1,2,3,4] *Arequipa, Universidad Nacional de San Agustín, Arequipa, Perú*

Resumen— El presente artículo analiza el PH-Tree como una estructura de datos eficiente para el manejo de datos multidimensionales y su aplicación en sistemas de agregación espacial. Se describen sus características fundamentales, algoritmos y operaciones básicas, así como las mejoras recientes que optimizan su rendimiento en escenarios con alta dimensionalidad y datos dispersos. Además, se destacan aplicaciones prácticas del PH-Tree en áreas como bases de datos científicas, consultas de constelaciones astronómicas y sistemas de información geográfica (GIS). Se presentan ejemplos de trabajos relevantes donde el PH-Tree ha demostrado ventajas significativas en eficiencia de búsqueda, uso de memoria y escalabilidad.

Palabras clave— PH-Tree, datos multidimensionales, agregación espacial, indexación espacial, consultas geoespaciales, optimización de consultas.

Abstract— This paper analyzes the PH-Tree as an efficient data structure for managing multidimensional data and its application in spatial aggregation systems. The fundamental characteristics, basic algorithms, and operations of the PH-Tree are described, along with recent improvements that optimize its performance in scenarios involving high dimensionality and sparse data. Practical applications of the PH-Tree in areas such as scientific databases, astronomical constellation queries, and geographic information systems (GIS) are highlighted. Relevant examples of studies where the PH-Tree demonstrated significant advantages in search efficiency, memory usage, and scalability are presented.

Keywords— PH-Tree, multidimensional data, spatial aggregation, spatial indexing, geospatial queries, query optimization.

INTRODUCCIÓN

En la era del big data y las aplicaciones científicas, el manejo eficiente de datos multidimensionales es esencial. Campos como la meteorología, la geoinformática, el análisis espacial y la visualización de datos dependen de estructuras de datos eficientes para realizar consultas y operaciones complejas sobre grandes volúmenes de información. Tradicionalmente, se han utilizado estructuras como *Quadtrees*, *KD-Trees* y *R-Trees* para indexar datos multidimensionales. Sin embargo, estas estructuras presentan limitaciones de rendimiento y escalabilidad cuando se manejan datos de alta dimensionalidad o conjuntos de datos muy grandes.

El **PH-Tree** es una estructura de índice multidimensional que pertenece a la familia de los *Quadtrees* pero incorpora técnicas avanzadas para mejorar su eficiencia en términos de espacio y rendimiento. Fue introducido por **Tilmann Zäschke** como una solución escalable para el almacenamiento y recuperación eficiente de datos en múltiples dimensiones. A diferencia de otros árboles multidimensionales, el PH-Tree combina características de los *Quadtrees* y los *Critbit Trees* (también conocidos como *Binary Prefix Tries*), lo que permite una navegación eficiente mediante el uso de hipercubos y operaciones a nivel de bits. Esta estructura es capaz de procesar hasta **64 dimensiones** en paralelo, aprovechando las capacidades de los registros de **64 bits** de los procesadores modernos. El PH-Tree destaca por su eficiencia en el espacio gracias a su uso de la compartición de prefijos y su capacidad para evitar la degeneración del árbol, incluso en presencia de actualizaciones frecuentes. Además, no requiere reequilibrado, lo que facilita las operaciones concurrentes y su implementación en sistemas de almacenamiento persistente. Estas características lo convierten en una opción eficiente para manejar grandes volúmenes de datos multidimensionales con una estructura estable y predecible. En escenarios de datos dispersos o *sparse data*, el PH-Tree también ofrece ventajas significativas. Por ejemplo, en sistemas de bases de datos de arrays como **SAVIME**, el PH-Tree ha demostrado ser más eficiente en consultas de rango y consultas puntuales en comparación con enfoques tradicionales sin índices. Aunque la inserción de datos en el PH-Tree puede ser más costosa debido a su complejidad inherente, esta desventaja se ve compensada por una recuperación de datos mucho más rápida en consultas complejas. Este artículo tiene como objetivo explorar las características y ventajas del PH-Tree, así como sus aplicaciones prácticas en el manejo de datos multidimensionales y dispersos. Se abordará su estructura interna, sus algoritmos de inserción, eliminación y consulta, además de las mejoras recientes que optimizan su

rendimiento con datos de alta dimensionalidad. También se discutirán aplicaciones en bases de datos científicas y se evaluará su rendimiento comparándolo con otras estructuras de indexación multidimensional.

DESCRIPCIÓN DEL PH-TREE

El **PH-Tree** es una estructura de datos multidimensional que combina características de *Quadtrees* y *Critbit Trees* (también conocidos como *Binary Prefix Tries*). Se destaca por su capacidad para manejar datos en múltiples dimensiones con alta eficiencia en espacio y rendimiento. Esta estructura permite realizar búsquedas, inserciones y eliminaciones de manera eficiente en conjuntos de datos que pueden llegar a tener hasta 64 dimensiones, aprovechando los registros de 64 bits de los procesadores modernos.

A diferencia de otras estructuras de árboles multidimensionales, el PH-Tree no requiere reequilibrado y evita la degeneración del árbol, manteniendo su estructura estable incluso ante actualizaciones frecuentes. Utiliza hipercubos para dividir el espacio en todas las dimensiones simultáneamente, en lugar de hacerlo una dimensión a la vez. Esto reduce el número de nodos necesarios y limita la profundidad máxima del árbol al número de bits del valor almacenado, lo que mejora la eficiencia de las operaciones de búsqueda y actualización.

El PH-Tree se basa en los siguientes principios fundamentales:

- **División en Hipercubos**: Cada nodo del árbol puede contener hasta 2^k hijos, donde k es la dimensionalidad del espacio. Esta división en hipercubos permite navegar por el árbol de manera eficiente, independientemente del orden de las dimensiones.

- **Intercalado de Bits (Z-Ordering)**: Los valores almacenados en el árbol se representan como cadenas de bits intercaladas. Esta técnica facilita la navegación y permite realizar consultas por rangos y vecinos más cercanos de manera eficiente.

- **Representaciones de Nodos**: El PH-Tree puede utilizar tres representaciones distintas para almacenar los nodos:

    1. *AHC (Array HyperCube)*: Representación en forma de hipercubo denso, adecuada cuando el nodo está casi lleno.
    2. *LHC (Linearized HyperCube)*: Representación lineal compacta, adecuada para nodos con pocos elementos.
    3. *BHC (Binary HyperCube)*: Representación basada en un árbol binario, útil para nodos grandes en espacios de alta dimensionalidad.

Una característica importante del PH-Tree es su capacidad para cambiar dinámicamente entre estas representaciones según la cantidad de datos almacenados en cada nodo, optimizando así el uso de memoria y el rendimiento de las operaciones.

Ejemplo Ilustrativo

Para comprender mejor la estructura del PH-Tree, consideremos el ejemplo de una división de un espacio bidimensional. La Figura 1 muestra una representación de un PH-Tree en dos dimensiones con tres puntos almacenados: (0001, 1000), (0011, 1000) y (0011, 1010). La raíz del árbol contiene referencias a sub-nodos basadas en los primeros bits de cada coordenada, y los sub-nodos contienen las divisiones adicionales necesarias para almacenar los puntos de manera eficiente.

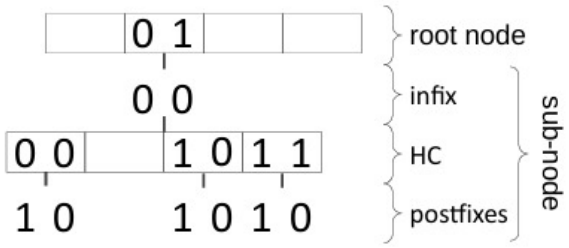

Fig. 1: Ejemplo de un PH-Tree bidimensional con tres puntos almacenados.

Este ejemplo ilustra cómo el PH-Tree organiza los datos en hipercubos y cómo los nodos almacenan las referencias a los sub-nodos y a los valores específicos (postfixes).

ESTRUCTURA DEL PH-TREE

El **PH-Tree** es una estructura de datos multidimensional diseñada para indexar y gestionar eficientemente grandes volúmenes de datos en múltiples dimensiones. La estructura del PH-Tree combina características de *Quadtrees* y *Critbit Trees* y se caracteriza por su capacidad para dividir el espacio en hipercubos, lo que permite un acceso eficiente y una reducción en el número de nodos requeridos. A continuación, se describen los principales componentes y representaciones de esta estructura.

Funcionamiento Interno del PH-Tree

El PH-Tree divide el espacio de datos en todas las dimensiones simultáneamente, a diferencia de otras estructuras como los *KD-Trees* que dividen una dimensión a la vez. Cada nodo en el árbol representa una división del espacio en hipercubos de dimensiones k. Los datos se almacenan mediante una técnica de intercalado de bits (*Z-ordering*), lo que facilita la navegación y permite realizar consultas eficientes.

La profundidad máxima del árbol está limitada por el número de bits utilizados para representar los valores almacenados. Por ejemplo, si los datos se representan con números enteros de 32 bits, la profundidad máxima del árbol será 32. Esta limitación evita la degeneración del árbol y elimina la necesidad de reequilibrado.

Representación de Nodos

El PH-Tree utiliza tres representaciones diferentes para almacenar los nodos, dependiendo de la cantidad de datos que contienen y de su distribución. Estas representaciones son:

1. **AHC (Array HyperCube)**: En esta representación, los nodos se almacenan en una estructura de hipercubo den-

so. Es eficiente cuando el nodo tiene muchos hijos y la mayoría de las posiciones del hipercubo están ocupadas. La búsqueda en un nodo AHC se realiza en tiempo constante $O(1)$.

2. **LHC (Linearized HyperCube)**: Cuando el nodo contiene pocos elementos, se utiliza una representación lineal compacta para reducir el uso de memoria. Los hijos se almacenan en una tabla ordenada, lo que permite realizar búsquedas mediante búsqueda binaria en $O(\log n)$.
3. **BHC (Binary HyperCube)**: Para nodos grandes en espacios de alta dimensionalidad, el PH-Tree utiliza una representación basada en un árbol binario. Esta representación es eficiente para inserciones y eliminaciones en nodos con una gran cantidad de elementos, evitando costosos desplazamientos en memoria.

El PH-Tree cambia dinámicamente entre estas representaciones dependiendo de la cantidad de datos en el nodo, optimizando el uso de memoria y el rendimiento de las operaciones.

Ejemplo de Estructura del PH-Tree

La Figura 2 muestra un ejemplo de un PH-Tree bidimensional con varios puntos insertados. La raíz del árbol divide el espacio en hipercubos y cada sub-nodo representa una subdivisión adicional. Los valores almacenados se representan como cadenas de bits intercaladas, lo que facilita la navegación dentro del árbol.

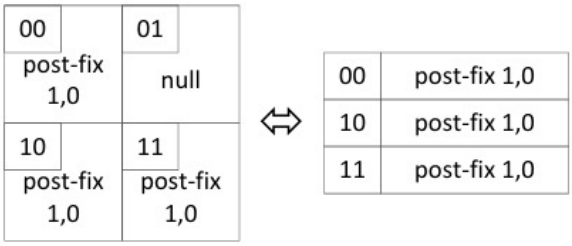

Fig. 2: Ejemplo de la estructura del PH-Tree bidimensional con subdivisiones en hipercubos.

Ventajas de la Estructura del PH-Tree

La estructura del PH-Tree ofrece varias ventajas en comparación con otras estructuras de indexación multidimensional:

- **Eficiencia en el Espacio**: Gracias a la compartición de prefijos y a las representaciones optimizadas de nodos, el PH-Tree utiliza menos memoria que otras estructuras cuando se manejan datos de alta dimensionalidad.
- **Escalabilidad**: La profundidad del árbol está limitada por el número de bits de los valores almacenados, lo que evita la degeneración y permite manejar eficientemente datos en hasta 64 dimensiones.
- **Navegación Eficiente**: La división en hipercubos y el intercalado de bits permiten realizar búsquedas y consultas de rango de manera eficiente, incluso en grandes volúmenes de datos.
- **Actualización Rápida**: Las inserciones y eliminaciones afectan como máximo a dos nodos, lo que facilita las actualizaciones concurrentes y el almacenamiento en sistemas persistentes.

Estas características hacen que el PH-Tree sea una opción adecuada para aplicaciones que requieren el manejo eficiente de datos multidimensionales, como bases de datos científicas, sistemas de información geográfica y análisis de datos en tiempo real.

Algoritmos y Operaciones del PH-Tree

El **PH-Tree** ofrece un conjunto de algoritmos y operaciones eficientes para manejar datos multidimensionales. Gracias a su estructura basada en hipercubos y el uso de intercalado de bits (*Z-ordering*), el PH-Tree permite realizar inserciones, eliminaciones y consultas de manera eficiente incluso en espacios de alta dimensionalidad. A continuación, se describen las principales operaciones y sus características.

Inserción de Datos

La operación de inserción en el PH-Tree consiste en añadir un nuevo punto multidimensional al árbol. El proceso incluye los siguientes pasos:

1. **Intercalado de Bits**: Los valores de cada dimensión se convierten en cadenas de bits y se intercalan para formar una sola cadena de bits.
2. **Localización del Nodo de Inserción**: Se recorre el árbol utilizando los bits intercalados para encontrar el nodo donde se insertará el nuevo punto.
3. **Actualización del Nodo**: Si la posición está vacía, se almacena el nuevo punto. Si hay un conflicto (un punto ya existente), se crea un sub-nodo para manejar la división del espacio.

La complejidad de la inserción es $O(w \cdot k)$, donde w es el número de bits por dimensión y k es la dimensionalidad del espacio. El uso de hipercubos permite realizar esta operación sin necesidad de reequilibrar el árbol, lo que facilita las actualizaciones concurrentes.

Eliminación de Datos

La eliminación de un punto en el PH-Tree implica localizar el nodo que contiene el punto y eliminarlo. Los pasos son los siguientes:

1. **Búsqueda del Punto**: Se utiliza la cadena de bits intercalada para navegar por el árbol y localizar el nodo que contiene el punto.
2. **Eliminación del Punto**: Se elimina el punto del nodo. Si el nodo queda vacío después de la eliminación, se elimina el nodo para optimizar el uso de memoria.

La complejidad de la eliminación es similar a la de la inserción, $O(w \cdot k)$. El proceso afecta como máximo a dos nodos, lo que garantiza una operación eficiente y sin reequilibrado.

Consultas de Punto (Point Queries)

Las consultas de punto permiten verificar si un punto específico existe en el árbol. El proceso consiste en:

1. **Intercalado de Bits**: Convertir el punto en una cadena de bits intercalada.
2. **Recorrido del Árbol**: Navegar por el árbol utilizando los bits intercalados para localizar el nodo correspondiente.
3. **Verificación**: Comparar el punto encontrado con el punto buscado para confirmar su existencia.

La complejidad de las consultas de punto es $O(w \cdot k)$ para árboles con nodos en representación AHC y $O(w \cdot k^2)$ para nodos en representación LHC debido a la búsqueda binaria en la tabla de referencias.

Consultas de Rango (Range Queries)

Las consultas de rango permiten encontrar todos los puntos dentro de un hiper-rectángulo definido por límites inferiores y superiores en cada dimensión. El algoritmo para realizar una consulta de rango es el siguiente:

1. **Localización del Punto de Inicio**: Se realiza una consulta de punto para localizar el nodo que representa la esquina inferior del hiper-rectángulo.
2. **Recorrido del Rango**: Se recorre el árbol visitando todos los nodos que intersectan con el hiper-rectángulo.
3. **Filtrado de Resultados**: Se verifican los puntos dentro de cada nodo para confirmar que están dentro del rango especificado.

El PH-Tree optimiza las consultas de rango mediante el uso de máscaras de bits para evitar explorar nodos que no pueden contener resultados válidos. La complejidad de las consultas de rango depende del número de puntos en el rango y del nivel de subdivisión del árbol.

Consultas de Vecino Más Cercano (kNN)

Las consultas de vecino más cercano (*k-nearest neighbors*) buscan los k puntos más cercanos a un punto dado. El algoritmo del PH-Tree para realizar estas consultas es el siguiente:

1. **Inicialización**: Se define una cola de prioridad para almacenar los candidatos más cercanos.
2. **Búsqueda Incremental**: Se exploran los nodos del árbol de manera ordenada según la distancia al punto de consulta.
3. **Actualización de Resultados**: Se actualiza la lista de los k puntos más cercanos a medida que se encuentran nuevos candidatos.

Gracias al orden Z implícito del PH-Tree, la búsqueda de vecinos más cercanos se realiza de manera eficiente, ya que el orden de exploración sigue una proximidad espacial aproximada.

Actualización de Datos

El PH-Tree permite actualizar los datos sin necesidad de reestructurar el árbol. La actualización de un punto implica eliminar el punto antiguo y luego insertar el nuevo punto. Dado que el PH-Tree no requiere reequilibrado, la operación de actualización es rápida y afecta solo a los nodos involucrados.

Resumen de Complejidad de Operaciones

- **Inserción**: $O(w \cdot k)$
- **Eliminación**: $O(w \cdot k)$
- **Consulta de Punto**: $O(w \cdot k)$ (AHC) o $O(w \cdot k^2)$ (LHC)
- **Consulta de Rango**: $O(w \cdot k \cdot n_{\text{matches}})$
- **Consulta kNN**: Dependiente del número de vecinos k y de la distribución de los datos

Estas operaciones hacen del PH-Tree una estructura eficiente y escalable para manejar datos multidimensionales en una amplia gama de aplicaciones.

MEJORAS RECIENTES DEL PH-TREE

El **PH-Tree** ha sido objeto de diversas mejoras desde su concepción original, con el objetivo de optimizar su rendimiento y adaptarse a los desafíos relacionados con la alta dimensionalidad y el manejo de datos dispersos. Estas mejoras están orientadas a resolver problemas de eficiencia en la inserción, consultas en datos agrupados y el uso eficiente de memoria. A continuación, se describen las principales optimizaciones implementadas.

Solución para Nodos Grandes

En la versión original del PH-Tree, los nodos con alta dimensionalidad podían contener hasta 2^k hijos, lo que resultaba en nodos muy grandes y operaciones de inserción y eliminación costosas debido al desplazamiento de grandes bloques de memoria. Para solucionar este problema, se introdujo la representación **BHC (Binary HyperCube)**. Esta representación divide los nodos grandes en una estructura de árbol binario, permitiendo realizar inserciones y eliminaciones de manera eficiente sin necesidad de desplazar grandes cantidades de datos.

La representación BHC se activa automáticamente cuando un nodo supera un umbral de 500 sub-nodos o 50 postfixes. Aunque esta representación puede ser ligeramente más lenta en consultas en comparación con AHC y LHC, su impacto negativo es mínimo debido a que solo se aplica a una pequeña fracción de los nodos en el árbol.

Preprocesamiento de Datos Agrupados

El PH-Tree mostró problemas de rendimiento con ciertos conjuntos de datos agrupados, debido a la creación de nodos con una baja relación de entradas por nodo y una cantidad excesiva de nodos poco poblados. Para mitigar este problema, se implementó una técnica de **preprocesamiento de datos** que transforma los datos antes de insertarlos en el árbol. Este preprocesamiento distribuye los puntos de manera más

uniforme, mejorando así la eficiencia en las consultas y reduciendo el número de nodos generados.

Optimización de Consultas en Datos Dispersos

El manejo eficiente de datos dispersos (*sparse data*) es un desafío común en aplicaciones científicas y de análisis espacial. Se optimizó el PH-Tree para acelerar las consultas de rango y de punto en datos dispersos. Esta optimización convierte al PH-Tree en una opción viable para sistemas que requieren consultas rápidas en grandes volúmenes de datos con cobertura incompleta del espacio, como registros meteorológicos y observaciones astronómicas.

Reducción de la Sobrecarga del Recolector de Basura

En implementaciones del PH-Tree en Java, las consultas generaban una gran cantidad de objetos temporales al extraer los resultados del árbol. Esto provocaba una sobrecarga en el recolector de basura (*garbage collector*), afectando el rendimiento del sistema. Para solucionar este problema, se implementó una técnica que permite reutilizar los objetos de resultado, reduciendo así la cantidad de memoria dinámica asignada durante las consultas y mejorando el rendimiento general.

Resumen de las Mejoras

Las mejoras recientes del PH-Tree pueden resumirse en los siguientes puntos:

- **Representación BHC**: Manejo eficiente de nodos grandes con alta dimensionalidad.
- **Preprocesamiento de Datos**: Mejora del rendimiento en conjuntos de datos agrupados.
- **Optimización de Datos Dispersos**: Mejora del rendimiento en consultas sobre datos con cobertura incompleta.
- **Reducción de Sobrecarga**: Reutilización de objetos para minimizar la carga del recolector de basura.

Estas optimizaciones aseguran que el PH-Tree continúe siendo una estructura eficiente y escalable para el manejo de datos multidimensionales en una amplia gama de aplicaciones.

APLICACIONES DEL PH-TREE

Trabajos y Aplicaciones Relacionadas

Diversos trabajos han explorado y comparado el **PH-Tree** con otras estructuras de indexación multidimensional debido a sus ventajas en eficiencia de búsqueda y manejo de datos dispersos. A continuación, se presenta una síntesis de trabajos destacados que hacen uso del PH-Tree.

BB-Tree: Una Estructura de Índice en Memoria Principal para Cargas de Trabajo Multidimensionales

El **BB-Tree** es una estructura de índice en memoria principal diseñada para manejar datos multidimensionales con operaciones de lectura y escritura eficientes. En este trabajo, se

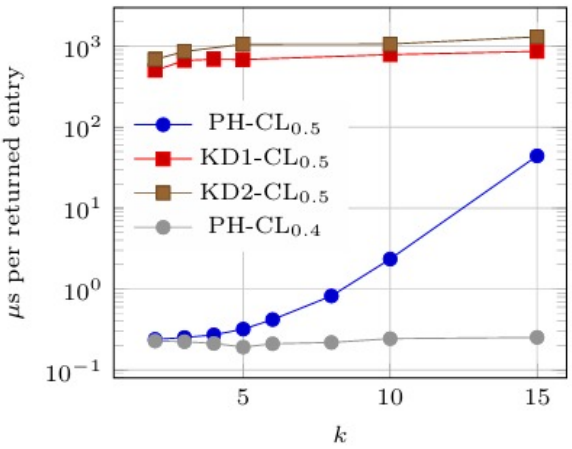

Fig. 3: Comparación del rendimiento en consultas con conjuntos de datos agrupados (2).

compara el rendimiento del BB-Tree con varias estructuras de indexación, incluido el PH-Tree.

El PH-Tree se destaca en este estudio por su excelente rendimiento en consultas de coincidencia exacta y su capacidad para manejar datos de alta dimensionalidad. Se observa que el PH-Tree ofrece un rendimiento competitivo en consultas de coincidencia exacta, siendo una de las estructuras más rápidas en este tipo de consultas. Sin embargo, para consultas de rango con una selectividad superior al 20 %, el BB-Tree supera al PH-Tree en términos de velocidad debido a su arquitectura optimizada para escaneos en memoria principal.

La evaluación también destaca que el PH-Tree es una de las estructuras más eficientes en términos de espacio, lo que lo convierte en una opción atractiva para aplicaciones con grandes volúmenes de datos en entornos de memoria limitada.

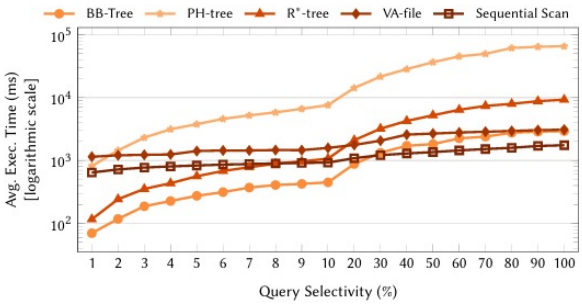

Fig. 4: Rendimiento de consultas de rango dependiendo de la selectividad.

Este estudio demuestra que el PH-Tree sigue siendo una opción sólida para aplicaciones que requieren consultas rápidas en datos multidimensionales y dispersos, especialmente cuando se prioriza la eficiencia en el uso de memoria y la simplicidad en el manejo de actualizaciones.

Pre-processing and Indexing techniques for Constellation Queries in Big Data

El **PH-Tree** se ha utilizado con éxito en la optimización de consultas espaciales complejas, conocidas como **constellation queries**. Estas consultas buscan patrones geométricos

en grandes volúmenes de datos espaciales, como los recopilados en estudios astronómicos. Un caso particular de interés es el fenómeno conocido como *Einstein Cross*, donde un único quásar aparece como cuatro objetos distintos debido a la lente gravitacional.

El principal desafío de las consultas de constelaciones es la explosión combinatoria que ocurre al buscar coincidencias geométricas en grandes conjuntos de datos. Para abordar este problema, se implementaron técnicas de pre-procesamiento y se utilizó el PH-Tree como estructura de indexación para mejorar la eficiencia de búsqueda.

Estrategias de Optimización Utilizadas:

1. **Pre-procesamiento del Conjunto de Datos**: Se indexan los datos espaciales utilizando el PH-Tree para facilitar el acceso eficiente a los vecinos de cada objeto. Esta técnica permite reducir el tamaño del conjunto de datos que necesita ser explorado al ejecutar una consulta.

2. **Pre-procesamiento de la Consulta**: Se optimizan las consultas eliminando elementos redundantes que no contribuyen a definir el patrón geométrico de interés. Esto reduce el número de combinaciones posibles y acelera el proceso de búsqueda.

3. **Anclaje de Elementos**: Se selecciona un elemento de anclaje en la consulta y se buscan candidatos cercanos en el conjunto de datos indexado con el PH-Tree. La búsqueda se limita a un radio definido por la distancia máxima esperada en el patrón geométrico.

Resultados Experimentales:

Los experimentos realizados con datos del **Sloan Digital Sky Survey (SDSS)** demostraron que el PH-Tree ofrece una mejora significativa en el tiempo de búsqueda de vecinos en comparación con estructuras como el **Quad-tree**. En particular, el PH-Tree:

- Redujo el tiempo de búsqueda en grandes conjuntos de datos espaciales.
- Garantizó respuestas completas a las consultas, incluso cuando los datos eran de alta dimensionalidad.
- Mostró un mayor uso de memoria en comparación con el Quad-tree, pero con una compensación aceptable debido a su mayor velocidad de búsqueda.

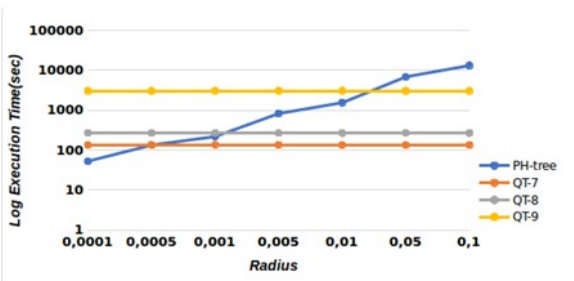

Fig. 5: Comparación de tiempos de ejecución entre el PH-Tree y el Quad-tree para consultas de vecinos en el conjunto de datos SDSS.

Estas optimizaciones convierten al PH-Tree en una herramienta efectiva para ejecutar consultas de constelaciones en grandes volúmenes de datos espaciales, permitiendo aplicaciones en astronomía, geoinformática y otras áreas que requieren el análisis eficiente de patrones geométricos.

GeoBlocks: A Query-Cache Accelerated Data Structure for Spatial Aggregation over Polygons

El **PH-Tree** se ha utilizado eficazmente en sistemas de agregación espacial que manejan grandes volúmenes de datos multidimensionales. Un ejemplo es su aplicación en el procesamiento y almacenamiento eficiente de datos geoespaciales dentro de bloques jerárquicos, denominados **GeoBlocks**.

En este enfoque, el PH-Tree facilita la indexación y consulta de datos organizados en una estructura de celdas espaciales. Cada celda corresponde a una subdivisión específica del espacio multidimensional, permitiendo un acceso rápido y eficiente a los datos almacenados. Esta técnica es especialmente útil en aplicaciones donde se requiere realizar agregaciones y consultas sobre regiones específicas de un espacio geoespacial, como sistemas de información geográfica (GIS) y análisis de datos meteorológicos.

Estrategias de Optimización Utilizadas:

1. **Descomposición en GeoBlocks**: Se subdivide el espacio en bloques jerárquicos (GeoBlocks) utilizando el PH-Tree para indexar los límites y contenidos de cada bloque. Esta descomposición permite realizar consultas espaciales limitando el área de búsqueda a bloques relevantes.

2. **Almacenamiento Jerárquico**: El PH-Tree permite almacenar los datos en diferentes niveles de granularidad dentro de los bloques, facilitando consultas a nivel de bloque o de sub-bloques según la precisión requerida.

3. **Agregación Eficiente**: La estructura del PH-Tree permite realizar operaciones de agregación espacial directamente sobre los bloques indexados, evitando el procesamiento innecesario de datos fuera del área de interés.

Resultados Experimentales:

Los experimentos realizados muestran que el uso del PH-Tree en sistemas de agregación espacial proporciona varias ventajas:

- **Reducción del Tiempo de Consulta**: El PH-Tree limita las búsquedas a los bloques relevantes, reduciendo significativamente el tiempo necesario para responder consultas espaciales complejas.
- **Eficiencia en el Uso de Memoria**: Gracias a su estructura compacta y la compartición de prefijos, el PH-Tree optimiza el uso de memoria al indexar grandes volúmenes de datos geoespaciales.
- **Escalabilidad**: El PH-Tree es capaz de manejar datos de alta dimensionalidad sin degradar su rendimiento, lo que lo hace adecuado para aplicaciones con grandes cantidades de datos geoespaciales.

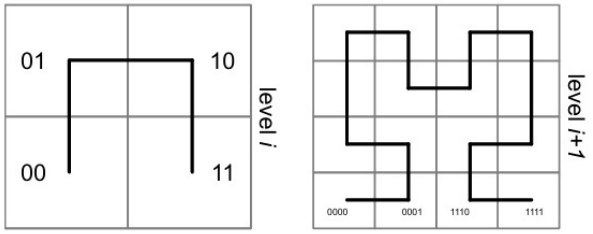

Fig. 6: Descomposición jerárquica de datos geoespaciales utilizando GeoBlocks indexados con el PH-Tree.

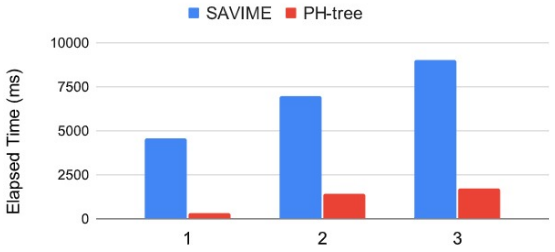

Fig. 7: Comparación de tiempos de ejecución entre el PH-Tree y SAVIME para consultas de rango.

Estas estrategias convierten al PH-Tree en una herramienta ideal para sistemas que requieren realizar agregaciones espaciales rápidas y consultas eficientes en grandes volúmenes de datos multidimensionales.

Managing Sparse Spatio-Temporal Data in SAVIME: an Evaluation of the Ph-tree Index

En el trabajo, los autores investigan la adopción del PH-Tree como una estructura de indexación en memoria para datos dispersos. Utilizan el sistema de base de datos multidimensional SAVIME como plataforma de prueba, comparando el rendimiento del PH-Tree en la ingestión de datos y en consultas puntuales y de rango.

El estudio destaca las siguientes contribuciones y hallazgos:

Desafíos en la Gestión de Datos Dispersos: - Los datos científicos a menudo son multidimensionales y presentan desafíos significativos cuando son dispersos. La gestión eficiente de estos datos requiere estructuras de indexación que minimicen el uso de memoria y mejoren los tiempos de respuesta de las consultas.

Uso del PH-Tree en SAVIME: - Se evaluó el rendimiento del PH-Tree en comparación con la estrategia TOTAL de SAVIME. La estrategia TOTAL clasifica cada valor de celda con sus correspondientes valores de índice, lo que resulta en un tiempo de consulta elevado para datos dispersos. - El PH-Tree, por otro lado, ofrece una estructura de indexación multidimensional eficiente, adecuada para datos dispersos, al combinar prefijos binarios y utilizar hipercubos para la navegación entre nodos.

Resultados Experimentales: - En términos de tiempo de inserción, SAVIME mostró un rendimiento superior con un tiempo de 3899 ms frente a los 68756 ms del PH-Tree, debido a la simplicidad del proceso de ingestión de SAVIME. - Sin embargo, en consultas de rango, el PH-Tree superó significativamente a SAVIME, mostrando tiempos de respuesta órdenes de magnitud más rápidos, especialmente en consultas que devolvieron grandes conjuntos de celdas. - En consultas puntuales, el PH-Tree también demostró una eficiencia notable, con tiempos de respuesta inferiores a 1 ms, mientras que SAVIME tomó varios segundos para completar consultas similares.

CONCLUSIONES

El **PH-Tree** se consolida como una estructura de datos robusta y eficiente para manejar grandes volúmenes de datos multidimensionales en una amplia gama de aplicaciones. A lo largo del artículo se han detallado sus características, operaciones, mejoras recientes y casos de uso, lo que permite extraer las siguientes conclusiones principales.

El **PH-Tree** destaca por su capacidad para manejar datos de hasta 64 dimensiones de manera eficiente. Su uso de hipercubos y el intercalado de bits permiten realizar búsquedas, inserciones y eliminaciones con una complejidad controlada, evitando los problemas de degeneración que afectan a otras estructuras de datos multidimensionales como los *Quadtrees* y *KD-Trees*.

Las representaciones de nodos **AHC (Array HyperCube)**, **LHC (Linearized HyperCube)** y **BHC (Binary HyperCube)** permiten adaptar dinámicamente la estructura del árbol según el número de elementos y la distribución de los datos. Esto optimiza el uso de memoria y el rendimiento en diferentes escenarios.

Se han implementado optimizaciones importantes, como la representación **BHC** para manejar nodos grandes y técnicas de preprocesamiento para datos agrupados y dispersos. Estas mejoras permiten que el PH-Tree mantenga su eficiencia incluso en contextos con alta dimensionalidad y grandes volúmenes de datos.

El **PH-Tree** ha demostrado su utilidad en diversas aplicaciones, como bases de datos científicas, consultas de constelaciones astronómicas, sistemas de información geográfica (GIS) y agregación espacial. Su capacidad para realizar consultas rápidas y eficientes lo convierte en una herramienta versátil en entornos donde la velocidad y el uso eficiente de memoria son críticos.

Los experimentos y estudios comparativos muestran que el PH-Tree supera a estructuras tradicionales como el *Quadtree* y el *kd-tree* en consultas de coincidencia exacta y agregación espacial. Aunque en algunas situaciones específicas (como consultas de rango con alta selectividad) otras estructuras pueden ser más rápidas, el PH-Tree ofrece un equilibrio ideal entre eficiencia de búsqueda, escalabilidad y uso de memoria.

En conclusión, el PH-Tree representa una solución sólida para los desafíos relacionados con la indexación y consulta de datos multidimensionales. Las mejoras recientes y su capacidad de adaptación a diferentes escenarios aseguran su relevancia en aplicaciones actuales y futuras que requieran un manejo eficiente de grandes volúmenes de datos espaciales y multidimensionales.

REFERENCIAS

[1] *The PH-tree: a space-efficient storage structure and multi-dimensional index* https://dl.acm.org/doi/10.1145/2588555.2588564

[2] Tilmann Zäschke. *The PH-Tree Revisited*. 2016. https://dl.acm.org/doi/abs/10.1145/2588555.2588564

[3] *Repositorio de PH-Tree* https://github.com/tzaeschke/phtree.git

[4] Pre-processing and Indexing techniques for Constellation Queries in Big Data. 2017. https://link.springer.com/chapter/10.1007/978-3-319-64283-3_12

[5] BB-tree: A main-memory index structure for multidimensional range queries. 2019. https://ieeexplore.ieee.org/abstract/document/8731440/

[6] GeoBlocks: A Query-Cache Accelerated Data Structure for Spatial Aggregation over Polygons. 2019. https://arxiv.org/abs/1908.07753

[7] Stiw Herrera, Larissa Miguez da Silva, Paulo Ricardo Reis, Anderson Silva, Fabio Porto. *Managing Sparse Spatio-Temporal Data in SAVIME: an Evaluation of the Ph-tree Index*. 2021. https://sol.sbc.org.br/index.php/sbbd/article/view/17895/17729