PH-Tree: Una revisión bibliográfica

Prof. Rolando Jesús Cárdenas Talavera
Ciencia de la Computación

18 de diciembre de 2024

Universidad Nacional de San Agunstín
Facultad de Ingeniería de Producción y Servicios

Escuela Profesional de
Ciencia de la Computación

Contenido

Introducción al PH-Tree

En la era de la **Big Data** y **aplicaciones científicas**, gestionar datos multidimensionales de forma eficiente es clave. Estas áreas dependen de estructuras como **Quadtrees**, **KD-Tree** y **R-Trees** que enfrentan desafíos de escalabilidad con datos de alta dimensionalidad. Entonces aparece la estructura **PH-Tree** que fue desarrollado por **Tilmann Zäschke** que combina características de los **Quadtrees** y **Critbit Trees**, este ofrece ventajas como navegación eficiente en hipercubos, uso óptimo de espacio de comparación de prefijos y capacidad de manejar 64 dimensiones en paralelo.

The PH-Tree – A Space-Efficient Storage Structure and Multi-Dimensional Index

Revised Version – 26 June 2014

Paper de Tilmann Zäschke

Definición

El **PH-Tree** es una estructura multidimensional que combina características de **Quadtrees** y **Critbit Trees**. Estos se destacan por manejar datos en múltiples dimensiones con alta eficiencia en espacio y rendimiento. A diferencia de otras estructuras, este no requiere reequilibrio y evita la degeneración del árbol manteniendo la estructura estable. Utiliza los hipercubos para dividir el espacio en todas las dimensiones de forma simultánea en vez de hacerlo en una sola. Esto reduce el número de nodos necesarios y limita la profundidad máxima del árbol al número de bits del valor almacenado.

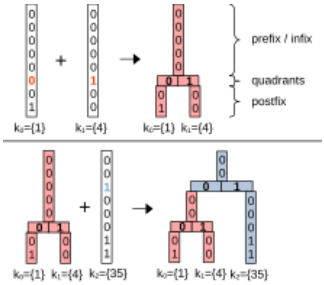
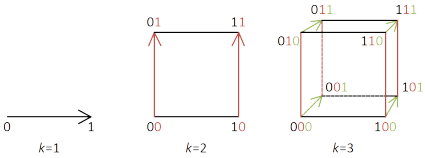


Ilustración del PH-Tree

Principios del PH-Tree

- **División en Hipercubos:** Cada nodo puede contener hasta 2^k hijos, donde k va a ser las dimensiones del espacio. Esto permite navegar por el árbol de manera eficiente, independientemente del orden de las dimensiones.
- **Intercalado de bits:** Los valores almacenados en el árbol van a ser representados en cadenas de bits intercaladas. Esto facilita la navegación, realizar consultas por rangos y vecinos más cercanos.
- **Representación de los nodos:** Se puede hacer 3 representaciones distintas para almacenar nodos:
  - AHC (Array HyperCube)
  - LHC (Linearized HyperCube)
  - BHC (Binary HyperCube)



Hipercubos del PH-Tree

Ventajas del PH-Tree

La estructura PH-Tree nos ofrece las siguientes ventajas:

- **Eficiencia en el Espacio:** Gracias a la compartición de prefijos y las representaciones optmizadas de los nodos, el **PH-Tree** utiliza menos memoria que otras estructuras al manejar datos de alta dimensionalidad.
- **Escalabilidad:** La profundidad del árbol va a estar limitada al número de bits de los valores almacenados, lo que evita la degeneración y manejar datos de hasta 64 dimensiones.
- **Navegación Eficiente:** La división en hipercubos e intercalado de bits permite realizar búsquedas y consultas por rango de forma eficiente.
- **Actualización Rápida:** Insertar y eliminar afecta máximo hasta 2 nodos, facilitando actualizaciones concurrentes y almacenamiento en sistemas concurrentes.

Estructura del PH-Tree

- **Funcionamiento Interno:** Divide el espacio de datos en todas las dimensiones simultáneamente, esto a diferencia de estructuras como el **KD-Tree**. Cada nodo en el árbol representa una división del espacio en los hipercubos de dimensión k. Los datos se almacenan por intercalado de bits facilitando su navegación y consultas. La profundidad está limitada por el número de bits utilizados para representar los valores almacenados evitando la degenaración del árbol y el reequilibrio.
- **Representación de Nodos:** Se puede hacer 3 representaciones a los nodos, esto depende de la cantidad de datos y su distribución:
  - **AHC:** Se almacena los nodos en un hipercubo denso. Es eficiente cuando el nodo tiene muchos hijos y la mayoría de las posiciones del hipercubo están ocupadas.
  - **LHC:** Si 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 permitiendo búsquedas binarias.
  - **BHC:** Para nodos grandes en espacios de alta dimesionalidad, se utiliza una representación en árbol binario y es eficiente para inserciones y eliminaciones en nodos con gran cantidad de elementos.

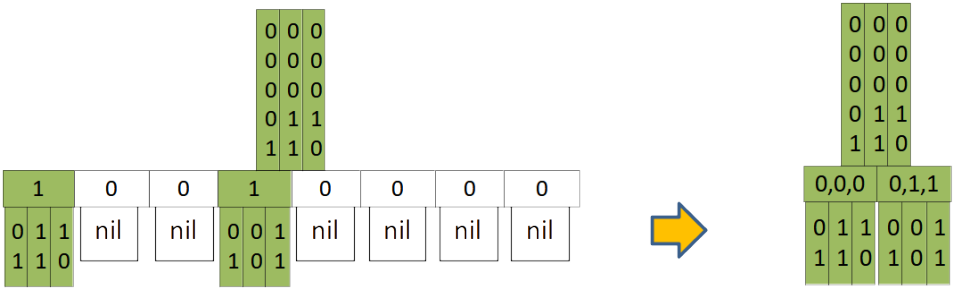


Figura: Representación de nodos AHC y LHC

Operaciones en el PH-Tree

Este ofrece un conjunto de operaciones y algoritmos muy eficientes para manejar datos multidimensionales. Gracias a que su estructura esta basada en **Hipercubos** y **uso intercalado de bits**, las inserciones, búsquedas, eliminaciones y consultas van a ser realizadas eficientemente. Ahora se explicará todas las funciones que tiene el **PH-Tree**.

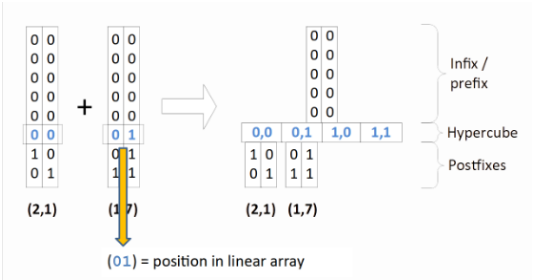
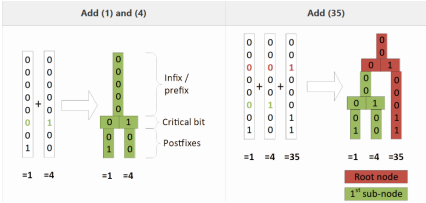


Figura: Una muestra de inserción PH-Tree

Inserción en el PH-Tree

Esta operación consta de añadir un nuevo punto multidimensional al árbol. Este proceso incluye los siguientes pasos:

- **Intercalado de Bits:** Los valores de cada dimensión se van a convertir en cadenas de bits y se van a ir intercalando en una sola cadena de bits.
- **Localización del Nodo de Inserción:** Se va a ir recorriendo el árbol utilizando los bits intercalados para encontrar el nodo donde se va a insertar el nuevo punto.
- **Actualización del Nodo:** Si la posición esta vacía, se almacena el nuevo punto. En caso de que haya un punto existente se crea un subnodo para manejar la división del espacio.

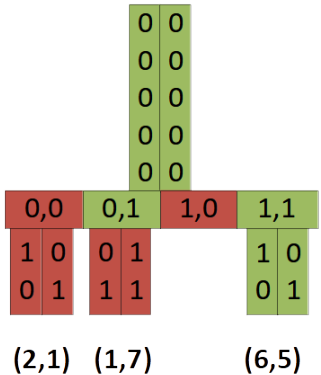


Inserción en un 1D PH-Tree

Búsqueda en el PH-Tree

Esto va a verificar si es que un punto se encuentra en nuestro árbol. Se va a realizar el siguiente proceso:

- **Intercalado de Bits:** El punto a buscar lo vamos a convertir en una cadena de bits intercalada.
- **Recorrido en el Árbol:** Se va a navegar por el árbol utilizando los bits intercalados para localizar el nodo correspondiente.
- **Verificación:** Comparar el punto encontrado con el punto buscado para confirmar su existencia.



Existe el punto (3, 4)?

Eliminación en el PH-Tree

La eliminación implica localizar el punto a eliminar y eliminarlo del árbol. El proceso es el siguiente:

- **Búsqueda del Punto:** Se utiliza la cadena bits intercalada para navegar por el árbol y localizar el nodo que contiene el punto.
- **Eliminación del Punto:** Simplemente se elimina el punto del nodo y si el nodo queda vacío también se elimina para optimizar el uso de memoria.

Búsqueda por Rango y Vecinos más Cercanos en el PH-Tree

Permiten encontrar todos los puntos dentro de un **Hiper-Rectángulo** definido por límites inferiores y superiores en cada dimensión. Se va a realizar el siguiente proceso:

- **Localización del Punto de Inicio:** Se realiza la búsqueda de un punto para localizar el nodo que va a representar la esquina inferior.
- **Recorrido por el Rango:** Se va reccoriendo el árbol visitando todos los nodos que intersectan.
- **Filtrado de Resultados:** Verifica los puntos dentro de cada nodo para confirmar que estén dentro del rango especificado.

Este busca los k vecinos más cercanos de un punto dado. Se realiza el siguiente proceso:

- **Inicialización:** Primero se crea una cola de prioridad para almacenar los candidatos más cercanos.
- **Búsqueda incremental:** Se explora los nodos de forma ordenada según la distancia al punto de consulta.
- **Actualzación de Resultados:** Se va actualizando la lista de los k puntos más cercanos a medida que encuentran nuevos candidatos.

Complejidad

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

Trabajos y Aplicaciones Relacionadas

Con la estructura del **PH-Tree**, se han realizado diversos trabajos donde se ha explorado y comparado el **PH-Tree** con otras estructuras de indexación multidimensional por la eficiencia en la búsqueda y manejo de datos dispersos. En total vamos a realizar una síntesis de trabajos destacados que hacen uso del **PH-Tree**.
En total vamos a realizar una síntesis de 4 artículos que son los siguientes:

- Pre-processing and Indexing techniques for Constellation Queries in Big Data (2017).
- BB-tree: A main-memory index structure for multidimensional range queries (2019).
- GeoBlocks: A Query-Cache Accelerated Data Structure for Spatial Aggregation over Polygons (2019).
- Managing Sparse Spatio-Temporal Data in SAVIME: an Evaluation of the Ph-tree Index (2021).

Pre-processing and Indexing techniques for Constellation Queries in Big Data (2017)

El **PH-Tree** también fue utilizado con éxito en la optimización de consultas espaciales complejas (conocidas como `constellation queries`). Lo que se busca en estas consultas es en encontrar patrones geométricos en grandes volúmenes de datos espaciales, como recopilados astronómicos. Para esta investigación se tomo el caso particular que es el fenómeno de *Einstein Cross* donde un único quásar aparece como 4 objetos distintos debido a la lente gravitacional.
Ahora, el principal desafío es la explosión combinatoria que ocurre al buscar coincidencias geométricas en grandes conjuntos de datos. Entonces, se han implementado técnicas de pre-procesamiento y se utiliza el **PH-Tree** como estructura de indexación para mejorar la eficiencia de búsqueda.

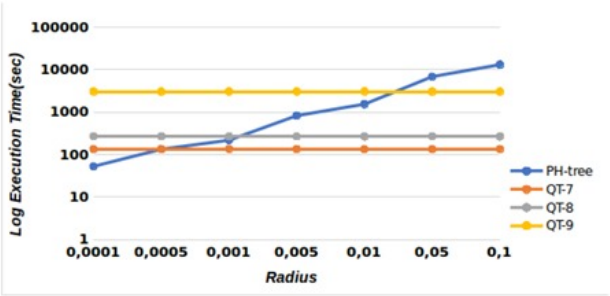
Estrategias de Optimización

- **Pre-procesamiento del Conjunto de Datos:** Usando el **PH-Tree** se indexan los datos espaciales para facilitar el acceso eficiente de los vecinos de cada objeto. Se reduce el tamaño del conjunto que necesita explorar al ejecutar una consulta.
- **Pre-procesamiento de Consulta:** Se optimiza las consultas eliminando elementos redundantes que no van a contribuir al patrón geométrico de interés.
- **Anclaje de Elementos:** Se selecciona un elemento de anclaje para la consulta y va buscando candidatos cercanos con ayuda del indexado en el **PH-Tree**.

Resultados experimentales

Se realizaron experimentos con datos del **Sloan Digital Sky Survey** y se demuestra que el PH-Tree ofrece una mejora significativa en el tiempo de búsqueda de vecinos en comparación de otras estructuras como el **Quadtree**. Entonces, el **PH-Tree**:

- Redujo el tiempo de búsqueda en grandes conjuntos de datos espaciales.
- Garantiza respuestas completas a las consultas, incluso en datos de alta dimensionalidad.
- Mostró un mayor uso de memoria en comparación con el **Quadtree**, pero con una compensación aceptable debido a su mayor velocidad de búsqueda.



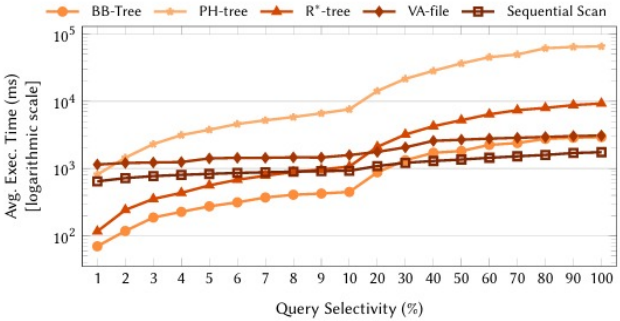
Resultados de la investigación

Estas optimizaciones convierten al PH-Tree en una herramienta efectiva para ejecutar consultas de constelaciones en grandes volúmenes de datos espaciales, lo que permite aplicarlo en astronomía, geoinformática y otras áreas que requieren análisis de patrones científicos.

BB-tree: A main-memory index structure for multidimensional range queries (2019)

Este trabajo nos habla acerca del **BB-Tree**, este esta diseñado para manejar datos multidimensionales con operaciones de lectura y escritura eficientes. Ahora, lo que se busca es comparar el rendimiento del **BB-Tree** con varias estructuras, entre ellas se encuentra el **PH-Tree**.

El **PH-Tree** se destaca por su rendimiento en consultas de coincidencia exactas y capacidad de manejar datos de alta dimensionalidad. Se ha observado que el **PH-Tree** ofrece el rendimiento competitivo para este tipo de consultas, pero si hablamos de consultas de rango con una selectividad superior del 20 %, la estructura **BB-Tree** llega a superar a lo que es el **PH-Tree** en términos de velocidad debido a la arquitectura optimizada para escaneos en memoria principal. Además, se destaca que el **PH-Tree** es una de las estructuras más eficientes en términos de espacio, lo que lo va a convertir en una opción atractiva para aplicaciones con grandes volúmenes de datos en entornos limitados.



Resultados de la investigación

Gracias a este estudio, se logra demostrar que el **PH-Tree** va a seguir siendo una opción sólida para aplicaciones que van a requerir consultas rápidas en datos multidimensionales y dispersos.

GeoBlocks: A Query-Cache Accelerated Data Structure for Spatial Aggregation over Polygons (2019)

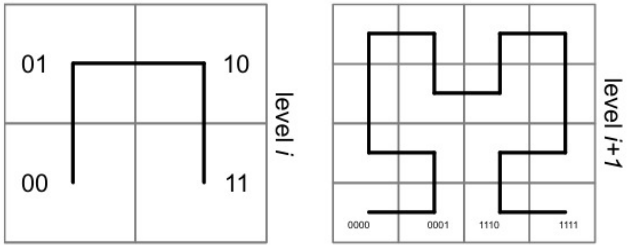
El **PH-Tree** también se ha utilizado en sistemas de agregación espacial que manejan grandes volúmenes de datos multidimensionales. Un ejemplo es su aplicación en procesamiento y almacenamiento eficiente de datos geoespaciales dentro de los bloques jerárquicos (**GeoBlocks**). En este enfoque, el **PH-Tree** facilita la indexación y consulta de datos organizados en una estructura de celdas espaciales dentro de bloques jerárquicos. Cada celda va a corresponder a una subdivisión específica en el espacio multidimensional permitiendo acceso rápido y eficiente a los datos almacenados. Este es usado en aplicaciones de sistemas de información geográfica (**GIS**) y análisis de datos metereológicos.

Estrategias de Optimización

- **Descomposición en GeoBlocks:** El espacio se va subdividiendo en bloques jerárquicos usando el **PH-Tree** indexando los límites y contenidos de cada bloque.
- **Almacenamiento Jerárquico:** El **PH-Tree** almacena datos en diferentes niveles de granularidad dentro de los bloques, facilita consultas a nivel de bloques o sub-bloques.
- **Agregación Eficiente:** Se permite operaciones de agregación espacial directamente sobre los bloques indexados.

Resultados Experimentales

- **Reducción de Tiempo de Consulta:** Se 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 la estructura compacta y compartición de prefijos, se optimiza el uso de memoria al indexar grandes volúmenes de datos geoespaciales.
- **Escalabilidad:** La estructura es capaz de manejar datos de alta dimensionalidad sin degradar su rendimiento, lo que lo hace adecuado a aplicaciones con grandes cantidades de datos geoespaciales.



Resultados de la investigación

Estas estrategias, van a convertir al **PH-Tree** en una herramienta ideal en sistemas que requieren realizar agregaciones espaciales rápidas y consultas eficientes.

Managing Sparse Spatio-Temporal Data in SAVIME: an Evaluation of the Ph-tree Index (2021)

En este artículo se investiga la adopción del **PH-Tree** como estructura de indexación en memoria para datos dispersos. Se utiliza un sistema de base de datos multidimensional **SAVIME** como plataforma de prueba, comparando el rendimiento del **PH-Tree** en la ingestión de datos y consultas puntuales y de rango. De este estudio podemos destacar:

- **Desafíos en Gestión de Datos Dispersos:** Los datos científicos a menudo son multidimensionales y cuando son dispersos presenta un desafío significativo. Una gestión eficiente requiere estructuras de indexación que minimicen el uso de memoria y mejoren los tiempos de respuestas en las consultas.
- **Uso del PH-Tree en SAVIME:** Evaluando 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, esto resulta un tiempo de consulta muy elevado para datos dispersos. Entonces, el **PH-Tree** ofrece una indexación multidimensional más eficiente al combinar prefijos binarios y utlizar hipercubos para la navegación entre nodos.

- **Resultados Experimentales:** Si hablamos de tiempo, **SAVIME** puede motrar un rendimiento superior con un tiempo de 3899 ms frente a los 68756 ms del **PH-Tree**, esto se debe a la simplicidad del proceso de ingestión de **SAVIME**. Sin embargo, al hacer consulta por rangos, el **PH-Tree** llega a superar al **SAVIME** mostrando tiempos de respuesta más rápidos. En consultas puntuales, el **PH-Tree** muestra una eficiencia notable con tiempos inferiores a 1 ms, mientras que **SAVIME** ha tomado varios segundos para completar consultas similares.

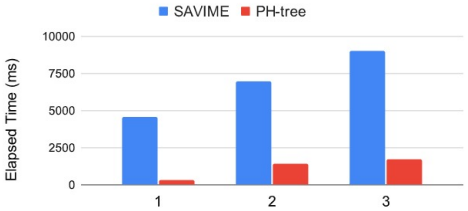


Figura: Tiempos en PH-Tree y SAVIME en consulta por rangos

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.

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

Gracias