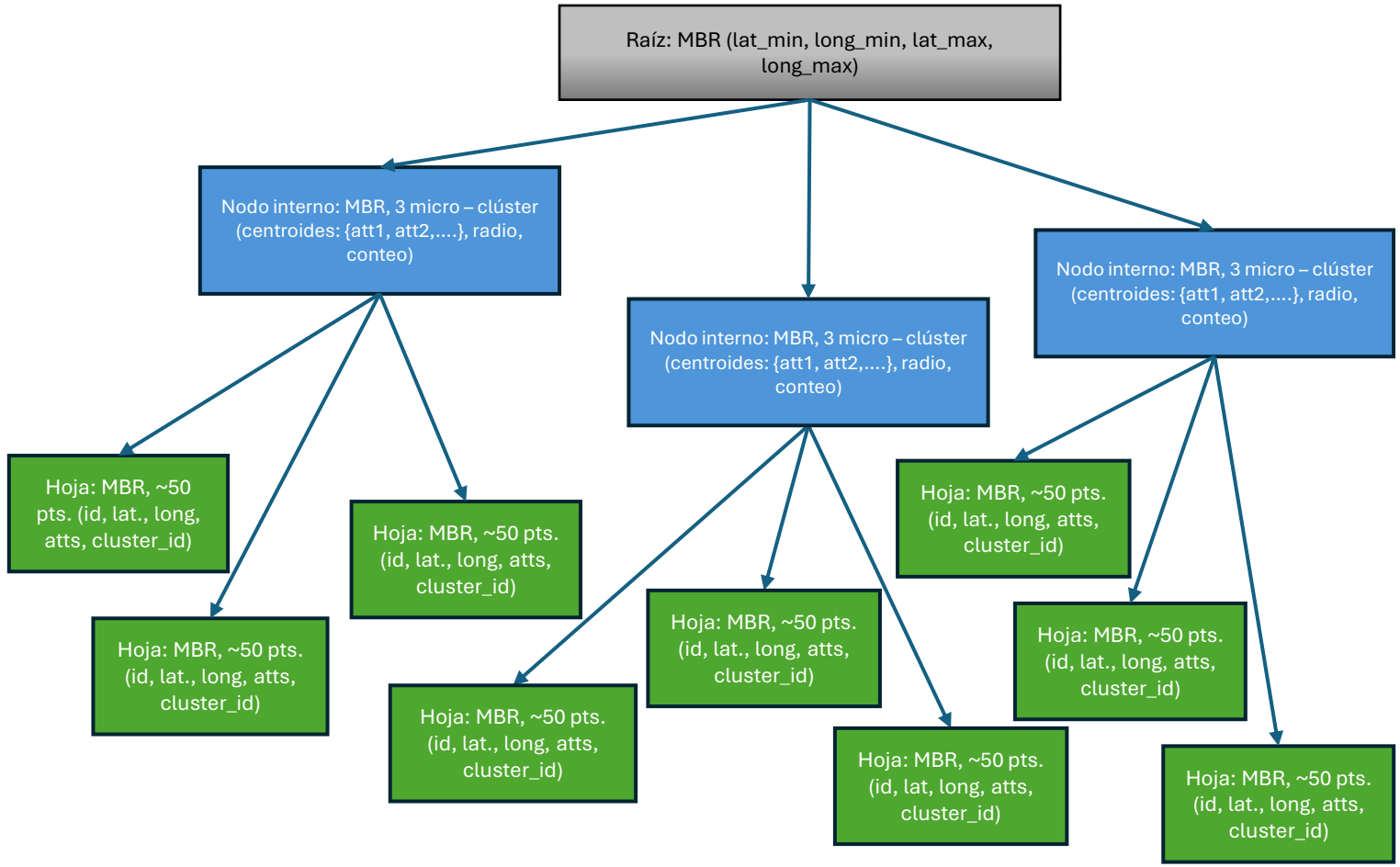
GeoCluster-Tree

- Estructura hibrida para consultas para K-NN y Clustering
- Estructura de Datos Avanzada
- Profesor: Erick Mauricio Gomez Nieto
- Alumno: Anthony Briceño Quiroz
- Fecha: 11 de mayo del 2025

Introducción y problema:

- Problema: Indexar ~100K-1M puntos (lat/long, 20 atributos(int)) para consultas en rango geográfico.
- Restricciones: C++/STL y uso de Python en preprocesamiento.
- Solución pensada y analizada: GeoCluter-Tree ,estructura con división hibrida y uso de micro-clustering y clustering inicial.
- Objetivo: Estructura eficiente para cualquier cantidad de datos y para consultas de rango geográfico.



- Descripción: Raíz con MBR, nodos con MBRs y ~3 micro-clusters, hojas con ~50 puntos (id, lat., long., attrs, cluster_id).
- Basado en R*-Tree para minimizar solapamiento de MBRs

GeoCluster-Tree

Diseño y Preprocesamiento

- ***Estructura:***
  1) GeoCluster – Tree es una estructura hibrida la cual toma como idea al R*- Tree , combinándolo con micro
     clusters (3) en cada nodo No Hoja, asi mismo poniendo un porcentaje para cada parte del dato (30~40% MBR y 70~60% atributos)
     3 micro cluster balancean memoria
- ***Preprocesamiento (Python):***
  1. CSV flexible: priorizar cualquier tipo de estructura del CSV,
     y lectura rápida de este mismo.
  2. Clustering: ~1000 clusters (MiniBatchKMeans)
  3. PCA(Principal): Reducción de dimensiones ~10 (~90-95% varianza) - NumPY
  4. (Distancia Euclidiana)
  5. UMAP(Opcional Secundario): Analizar los atributos (lineal – no lineal)
  6. Curva de Hilbert

  - CSV (flexible) → Clustering (~1000) → PCA (UMAP opcional) → Hilbert → Binario
  - Transformación de datos ásperos a ~10 optimizados

Algoritmos de la Estructura

Inserción O(n log n)

- La inserción para esta estructura será usada de la misma forma que un R*- Tree
- División Hibrida, lo cual garantiza una correcta distribución de los pts. en forma geográfica y en forma de atributo.
- Micro – Clúster (3 por nodo)
- Buffering (Preprocesamiento)
- Filtramos por MBR

Consulta 1 (Datos semejantes a un dato en particular en rango)

- Priorizamos subárboles con micro-clusters
- Calculamos distancias exactas en las hojas.
- Función Búsqueda Optimizada para este caso (trabajar con clústeres)

Consulta 2 (Puntos semejantes en rango)

- Filtración por MBR
- Combinación de micro-clusters cercanos (centroides, inicio_cluster_id)
- Función Búsqueda Optimizada para este caso (trabajar con clústeres)

Eficiencia, Justificación y Ventajas:

- ***Eficiencia:***
  a. Inserción: O(N log N)
  b. Consultas: O(N log N +M)
  c. Memoria (~10-96 MB 1M. pts.)
- ***Justificación:***
  a. R* - Tree, ideal para rangos geográficos(MBRs minimiza el solapamiento)
  b. Micro-clusters: Aceleración de consultas k-nn
  c. Clustering inicial: Mejora el ordenamiento de datos en el preprocesamiento,.
  d. PCA/Selección: Facil de implementar (Python NumPy) eficiente y robusto manteniendo una varianza de ~90%-95%. UMAP por su parte es ideal para datos no lineales, lo cual de alguna manera me puse a analizar si es que un att. Depende de otro (no lineal)
  e. Numpy: Ofrece herramientas matriciales, y no un PCA completo (scikit-learn)

¿Por qué elegiste un R-Tree como base para el GeoCluster-Tree en lugar de otras estructuras como k-d tree, quad-tree, o árbol B?

El R*-Tree es ideal para consultas espaciales multidimensionales, como k-NN y clustering con geográfica. Además de k-d tree, el R*-Tree puede manejar ~10 dimensiones sin degradar rendimiento y es más flexible para datos no uniformes. Además, el R*-Tree minimiza solapamiento y área de MBRs.

¿ Porque 3 micro-cluster? Y no mas o menos, ¿ o incluso No usarlos?
-50 pts. por nodo hoja
Sin ellos no podríamos priorizar subárboles por similitud, y esto aumenta el acceso a cada nodo de O(log N+M) a O (log N+Km) k = numero de nodos visitados

# Micro-Cluster	Memoria(2-20K nodos)	Tiempo k-NN (1M puntos)	Tiempo Clustering (1M puntos)	Precisión Clustering	Complejidad Implementación
0	0 mb	2-15 ms	5 – 20 ms	Baja	Muy baja
1	0.2 – 2 mb	1.2 -12 ms	3 – 15 m	Media-baja	Baja
2	0.4 -4 mb	1.1 – 11 ms	2 – 12 ms	Media	Media
3	0.5 – 5 mb	1 - 10 ms	1 – 10 ms	Media-Alta	Media
5	0.9 – 9 mb	0.95 – 9.5 ms	1.2 – 12 ms	Alta	Alta
Dinámico	0.5 – 5 mb(prom)	1 – 10 ms	1 -10 ms	Alta	Muy alta

Referencias

- [1] N. Beckmann, H.-P. Kriegel, R. Schneider, y B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," en Proc. ACM SIGMOD Int. Conf. Manage. Data, Atlantic City, NJ, USA, May 1990, pp. 322–331, doi: 10.1145/93597.98741.
- [2] J. Han, M. Kamber, y J. Pei, Data Mining: Concepts and Techniques, 3.ª ed. San Francisco, CA, USA: Morgan Kaufmann, 2011.
- [3]L. McInnes, J. Healy, y J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," arXiv preprint arXiv:1802.03426, 2018. [En línea]. Disponible:
- [4] IBM, "¿Qué es el clustering?," *IBM Think*, [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/clustering. [Accedido: 29-abril-2025].
- [5] IBM, "¿Qué es el análisis de componentes principales?," *IBM Think*, [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/principal-component-analysis. [Accedido: 30-abril-2025].
- [6] Matesmates, "La curva de Hilbert," *Matesmates Blog*, 29-mar-2012. [En línea]. Disponible en: https://matesmates.wordpress.com/2012/03/29/la-curva-de-hilbert/. [Accedido: 30-abril-2025].
- [7] Wikipedia, "Hilbert R-tree," *Wikipedia: The Free Encyclopedia*, 6 may. 2024. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Hilbert_R-tree. [Accedido: 11-may-2025].