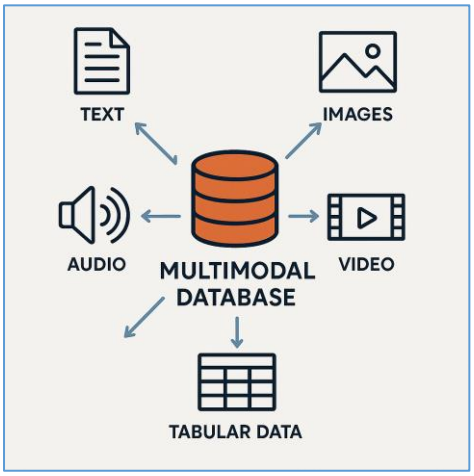
Proyecto Integrador
Sistema de Base de Datos Multimodal con Indexación Avanzada



1- Objetivo General

Diseñar e implementar un sistema de base de datos multimodal capaz de indexar y consultar datos estructurados y no estructurados, integrando técnicas de indexación avanzadas en Python (como R-Tree, IVF, etc.). El sistema incluirá un lenguaje de consulta tipo SQL (ParserSQL) y una API backend para integrarse con aplicaciones FrontEnd orientadas a tareas prácticas como sistemas de recomendación o reconocimiento facial. La integración multimodal permitirá gestionar y combinar diversos tipos de datos —texto, imágenes, audio, video y estructuras tabulares— ofreciendo una solución robusta y versátil para entornos de datos heterogéneos.

2- Ventajas de la Integración Multimodal

- **Mayor riqueza semántica:** La combinación de múltiples tipos de datos (texto, imágenes, audio, video, datos estructurados) proporciona un contexto más amplio y profundo, lo que permite una comprensión más completa de la información.
- **Incremento en la precisión analítica:** La integración de diversas modalidades permite realizar inferencias más robustas y precisas, al aprovechar diferentes perspectivas y representaciones de los datos.
- **Versatilidad en aplicaciones complejas**: Facilita el desarrollo de soluciones en dominios donde los datos provienen de fuentes heterogéneas, como sistemas de recomendación, análisis de contenido multimedia, salud digital, vigilancia inteligente, entre otros.

Jiaheng Lu and Irena Holubová. 2019. Multi-model Databases: A New Journey to Handle the Variety of Data. ACM Comput. Surv. 52, 3, Article 55 (May 2020)

3- Características Principales del Proyecto

1. Soporte para **Índices Tradicionales** de datos tabulares
    - Sequential File / AVL File
    - Static/Dynamic hash File
    - B+ Tree File
2. Soporte para **Índices Avanzados** de datos no estructurados
    - RTree: Para datos espaciales y multidimensionales.
    - GIN: Para campos de texto y búsquedas invertidas.
    - IVF: Para búsquedas vectoriales rápidas en embeddings.
    - LSH: Para similitud en datos de alta dimensionalidad.

3. **ParserSQL Personalizado:** El sistema debe incluir un lenguaje personalizado tipo SQL para interactuar con la base de datos.
4. **API Backend:** Proporcionar una API RESTful en Python (e.g., usando Flask o FastAPI) para permitir:
    - Gestión de tablas y datos (creación, consulta, eliminación).
    - Ejecución de consultas personalizadas desde el ParserSQL.
    - Interacción con aplicaciones FrontEnd.
5. **Aplicaciones FrontEnd:** Las funcionalidades del backend deben permitir la conexión con aplicaciones FrontEnd para casos de uso específicos:
    - **Sistema de gestión de inventarios:** Gestión de productos en almacenes, con búsquedas por código, nombre, categoría o ubicación.
        - Soporte para productos con dimensiones físicas (peso, tamaño).
        - Se puede indexar por ubicación en un almacén 3D (R-Tree).
    - **Sistema de gestión geoespacial:** Gestión de ubicaciones, rutas o zonas geográficas:
        - Estaciones meteorológicas
        - Puntos de interés turístico
        - Rastreo de vehículos o envíos
    - **Aplicaciones de IA con datos multidimensionales:** aplicaciones que permitan la búsqueda por similitud sobre vectores numéricos:
        - **Reconocedor de rostros**: Búsqueda de imágenes similares en base a descriptores faciales.
        - **Detector de copias de audios**: Detección de similitudes entre clips de audio usando descriptores.
        - **Sistema de recomendación de noticias**: Basado en similitudes entre textos o embeddings.
        - **Recomendación musical**: Usando similitudes en lyrics (texto) y características de audio.
        - **E-Commerce**: Recomendación de productos basados en descriptores de imágenes y características textuales.

4- Estructura del Proyecto

El proyecto se realizará en dos fases:

1) Organización e Indexación Eficiente de Archivos con Datos Multidimensionales
2) Mapeando el Caos: Indexación y Organización de Datos No Estructurados para Datos Multimedia (texto, imágenes, audio, video).

Proyecto 1
Organización e Indexación Eficiente de Archivos con Datos Multidimensionales.

1- Enunciado

En grupos de máximo cinco integrantes, deberán desarrollar un mini gestor de bases de datos que permita aplicar de manera eficiente técnicas de organización de archivos físicos, incluyendo sus operaciones fundamentales: inserción, eliminación y búsqueda. Además, el sistema deberá incorporar técnicas de indexación para datos multidimensionales (como vectores numéricos y datos espaciales), utilizando estructuras como el R-Tree.

El objetivo principal de este proyecto es comprender y aplicar dichas técnicas para optimizar la gestión, el almacenamiento y la recuperación de datos estructurados dentro de un modelo relacional basado en tablas, integrando también el soporte para datos complejos y multidimensionales.

Para la validación funcional del sistema, se deberán utilizar archivos planos con datos reales, preferentemente obtenidos de plataformas como Kaggle. El proyecto debe demostrar eficiencia en las operaciones, claridad en la estructura del código e incluir una breve documentación técnica que explique el diseño, las decisiones adoptadas y los resultados obtenidos.

2- Requerimientos generales

a. Implementar las siguientes técnicas de organización de archivos en memoria secundaria.
    1. Sequential File o AVL File
    2. ISAM-Sparse Index (solo dos niveles)
    3. Extendible Hashing
    4. B+ Tree
    5. RTree

b. Operaciones que se deben implementar para los índices tradicionales:
    1. La búsqueda específica puede retornar más de un elemento que coincide con la key
       `search(key)`
    2. La búsqueda por rango retorna todos los registros que se encuentran entre las dos llaves de búsqueda
       `rangeSearch(begin-key, end-key)`
    3. Agregar un registro al archivo respetando la técnica de organización
       `add(registro)`
    4. Proponer un algoritmo de eliminación para cada técnica
       `remove(key)`

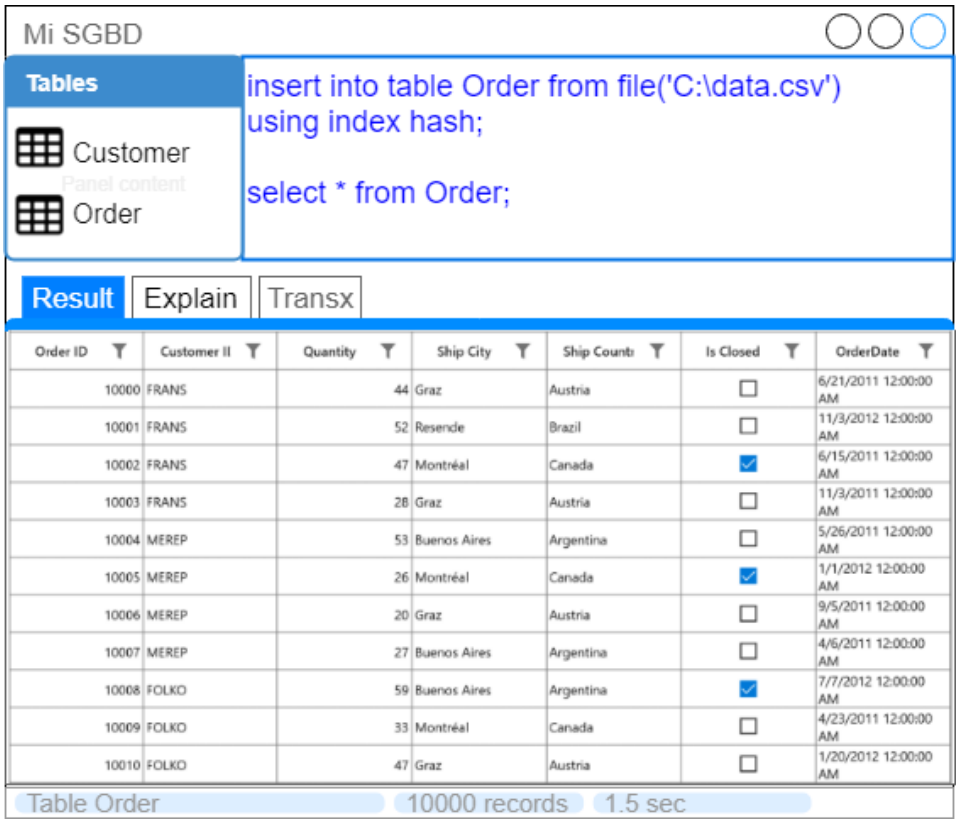
c. Operaciones que se deben implementar para el índice RTree
    1. Búsqueda por rango en donde el parámetro de consulta es un punto multidimensional y el radio
       `rangeSearch(point, radio)`
    2. Búsqueda de los K vecinos más cercanos al punto de consulta
       `rangeSearch(point, k)`

d. Implementar un parser de código SQL a sentencias ejecutables.
Ejemplo de creación de tablas:

```
CREATE TABLE Restaurantes (
  id INT KEY INDEX SEQ,
  nombre VARCHAR[20] INDEX BTree,
  fechaRegistro DATE,
  ubicacion ARRAY[FLOAT] INDEX RTree
);
```

Ejemplos de sentencias:
- create table Restaurantes from file "C:\restaurantes.csv" using index isam("id")
- select * from Restaurantes where id = x
- select * from Restaurantes where nombre between x and y
- insert into Restaurantes values (...)
- delete from Restaurantes where id = x
- select * from Restaurantes where ubicacion in (point, radio)

e. Mostrar los resultados de forma amigable a usuario. Ver la siguiente GUI de referencia.



Order ID	Customer II	Quantity	Ship City	Ship Countr	Is Closed	OrderDate
10000	FRANS	44	Graz	Austria	☐	6/21/2011 12:00:00 AM
10001	FRANS	52	Resende	Brazil	☐	11/3/2012 12:00:00 AM
10002	FRANS	47	Montréal	Canada	☑	6/15/2011 12:00:00 AM
10003	FRANS	28	Graz	Austria	☐	11/3/2011 12:00:00 AM
10004	MEREP	53	Buenos Aires	Argentina	☐	5/26/2011 12:00:00 AM
10005	MEREP	26	Montréal	Canada	☑	1/1/2012 12:00:00 AM
10006	MEREP	20	Graz	Austria	☐	9/5/2011 12:00:00 AM
10007	MEREP	27	Buenos Aires	Argentina	☐	4/6/2011 12:00:00 AM
10008	FOLKO	59	Buenos Aires	Argentina	☑	7/7/2012 12:00:00 AM
10009	FOLKO	33	Montréal	Canada	☐	4/23/2011 12:00:00 AM
10010	FOLKO	47	Graz	Austria	☐	1/20/2012 12:00:00 AM

3- Consideraciones de la implementación

a. ***En el Sequential Index La función add(Registro registro)*** debe utilizar un espacio auxiliar para guardar los nuevos registros. Cuando el espacio auxiliar llegue a K registros, aplique un algoritmo de reconstrucción del archivo de datos

manteniendo el orden físico de acuerdo a la llave seleccionada. Asegúrese de mantener los punteros actualizados.

b. **En el ISAM** partir construyendo el índice estático de solo 2 niveles de indexación a partir de un conjunto de datos. Para nuevas inserciones se generan los overflow pages (encadenamiento de páginas). Debe definir el factor de bloque tanto en las páginas de datos como en las páginas del índice.
c. La función **rangeSearch** debe usar el índice para buscar el begin-key y luego recorrer los registros de acuerdo la estructura del índice. Esta función no es soportada por las técnicas de hashing.
d. Para el **RTree** puede usar una implementación del índice ya desarrollada en Python.
e. Use adecuadamente los conceptos de programación orientado a objetos y programación genérica para que el programa soporte cualquier dominio de datos.
f. El backend se construye 100% en Python. Para el frontend se sugiere utilizar Python y alguna librería gráfica.
g. Es importante que todos participen en la implementación, se tomará en cuenta los commits en Github.

4- Informe del proyecto

- Archivo en Markdown / Wiki / Latex.
- Cuide la ortografía y consistencia en los párrafos.
- Al final del informe poner el **video explicando** el **funcionamiento** del programa, casos de uso, y aspectos importantes de la implementación. El video no debe exceder los 15 minutos y deben participar todos los integrantes del grupo.
- Aspectos de evaluación e informe:

Item	Descripción
- Introducción (2 pts)	- Objetivo del proyecto. - Describir la aplicación interesante en donde se pueda combinar las diferentes técnicas de indexación a implementar. - Resultados que se esperan obtener al aplicar las técnicas de indexación.
- Técnicas Utilizadas. (9 pts)	- Describa brevemente las técnicas de indexación de archivos que ha elegido. - Explique el algoritmo de inserción, eliminación y búsqueda (use gráficos para un mayor entendimiento) - Se debe realizar un análisis comparativo teórico de las técnicas implementadas en base a los **accesos a memoria secundaria** tanto para las operaciones de inserción, búsqueda y eliminación. - En el código debe estar optimizado en el manejo de memoria secundaria. - Explicar claramente como se realizó el parser del SQL.
- Resultados Experimentales (4 pts)	- Cuadro y/o gráfico comparativo de desempeño de las técnicas de indexación de archivos sobre el dominio de datos.

	Tanto para inserción como para búsqueda. - Considerar dos métricas: total de accesos a disco duro (read & write) y tiempo de ejecución en milisegundos. - Discusión y análisis de los resultados experimentales.
- Pruebas de uso y presentación (5 pts)	- Presentar las pruebas de uso de la aplicación en interfaz gráfica amigable e intuitiva. - Debe ser evidente el aporte de los índices en la aplicación seleccionada. - Recuerde mostrar la funcionalidad del aplicativo en el video.

5- Entregable

- Los alumnos formarán grupos de máximo cinco integrantes.
- El código fuente del proyecto será alojado en GitHub.
- En el Canvas subir solo el **enlace público** del proyecto.
- La fecha límite de entrega es el --------.