

## **Proyecto 1**

### **Organización e Indexación Eficiente de Archivos con Datos Multidimensionales.**

#### **1- Enunciado**

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 espaciales.

Los alumnos deberán formar grupo de máximo cinco integrantes, y 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.

Para la validación funcional del sistema, se deberán utilizar archivos planos con datos reales, preferentemente obtenidos de plataformas como [Kaggle](https://www.kaggle.com/). 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 (de 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 datos espaciales con 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. ParserSQL: Implementar un mecanismo para transformar consultas de estilo 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. FrontEnd: mostrar los resultados de forma amigable a usuario. Ver la siguiente GUI de referencia.

Mi SGBD

Tables

Customer

Order

insert into table Order from file('C:\data.csv') using index hash;

select \* from Order;

Result

Explain

Transx

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

Table Order

10000 records

1.5 sec

### 3- Consideraciones de la implementación

- 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 con la llave seleccionada. Asegúrese de mantener los punteros actualizados.
- 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.
- 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.
- Para el **RTree** puede usar una implementación del índice ya desarrollada en Python.
- Use adecuadamente los conceptos de programación orientado a objetos y programación genérica para que el programa soporte cualquier dominio de datos.
- El backend se construye 100% en Python. Para el frontend se sugiere utilizar Python y alguna librería gráfica.
- Es importante que **todos los integrantes 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)	<ul style="list-style-type: none"><li>- Objetivo del proyecto.</li><li>- Describir la aplicación interesante en donde se pueda combinar las diferentes técnicas de indexación a implementar.</li><li>- Resultados que se esperan obtener al aplicar las técnicas de indexación.</li></ul>
- Técnicas Utilizadas. (9 pts)	<ul style="list-style-type: none"><li>- Describa brevemente las técnicas de indexación de archivos que ha elegido.</li><li>- Explique el algoritmo de inserción, eliminación y búsqueda (use gráficos para un mayor entendimiento)</li><li>- Se debe realizar un análisis comparativo teórico de las técnicas implementadas en base a los <b>accesos a memoria secundaria</b> tanto para las operaciones de inserción, búsqueda y eliminación.</li><li>- En el código debe estar optimizado en el manejo de memoria secundaria.</li><li>- Explicar claramente como se realizó el parser del SQL.</li></ul>

<ul style="list-style-type: none"><li>- Resultados Experimentales (4 pts)</li></ul>	<ul style="list-style-type: none"><li>- Cuadro y/o gráfico comparativo de desempeño de las técnicas de indexación de archivos sobre el dominio de datos. <b>Tanto para inserción como para búsqueda.</b></li><li>- Considerar dos métricas: total de accesos a disco duro (read &amp; write) y tiempo de ejecución en milisegundos.</li><li>- Discusión y análisis de los resultados experimentales.</li></ul>
<ul style="list-style-type: none"><li>- Pruebas de uso y presentación (5 pts)</li></ul>	<ul style="list-style-type: none"><li>- Presentar las pruebas de uso de la aplicación en interfaz gráfica amigable e intuitiva.</li><li>- Debe ser evidente el aporte de los índices en la aplicación seleccionada.</li><li>- Recuerde mostrar la funcionalidad del aplicativo en el video.</li></ul>

#### 5- Entregable

- El código fuente del proyecto será alojado en GitHub.
- Para facilitar el despliegue del software deben usar Docker Compose.
- En el Canvas subir solo el **enlace público** del proyecto.
- La fecha límite de la entrega figura en el Anexo del Silabo.