UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Base de Datos II

Prof. Heider Sanchez

2025 - 2

Laboratorio 2: Sequential File vs AVL File

1. Introducción

El propósito de este laboratorio es implementar y comparar el desempeño de dos estructuras de almacenamiento de datos en memoria secundaria utilizando un dataset de empleados:

1. Archivo Secuencialmente Ordenado (Sequential File)
2. Archivo organizado como Árbol Binario de Búsqueda Balanceado (AVL File)

Los estudiantes analizarán el tiempo de acceso y eficiencia en las operaciones fundamentales utilizando como clave de búsqueda principal el **ID de empleado** (`Employee_ID`):

- insert(record)
- search(key)
- remove(key)
- rangeSearch(init_key, end_key)

Requerimientos de implementación:

- La implementación de ambos métodos será en **Python**.
- Utilizar **archivos binarios** con registro de **longitud fija**.
- Medición y comparación de los **tiempos de acceso** en ambos métodos.
- Usar la siguiente **estructura de registro** para representar un empleado:

Campo	Tipo de Dato	Tamaño (bytes)
Employee_ID	`int`	4
Employee_Name	`string`	30
Age	`int`	4
Country	`string`	20
Department	`string`	20
Position	`string`	20
Salary	`float`	4
Joining_Date	`string (DD/MM/YYYY)`	10

Nota: Los tamaños de los campos string corresponden al máximo de caracteres permitidos, rellenando con espacios si es necesario.

2. Desarrollo

P1 (6 puntos): Implementación del Archivo Secuencial

1. Carga de datos desde un archivo csv (`employee.csv`).
2. Función para **insertar** nuevos registros usando espacio auxiliar. El archivo original debe reconstruirse con el espacio extra cuando este último exceda k registros.
3. Función de **búsqueda secuencial** por `Employee_ID` .
4. Función para **eliminar** un registro marcándolo como eliminado (por ID de empleado). En la reconstrucción del archivo de datos no se deben considerar los registros eliminados lógicamente.
5. Función para la **búsqueda por rango** que retorne todos los empleados entre un rango de `Employee_ID` especificado.

P2 (6 puntos): Implementación del Archivo AVL

1. Carga de datos desde un archivo csv (`employee.csv`).
2. Función para **insertar** nuevos registros actualizando correctamente los punteros de jerarquía.
3. Función para **buscar** un empleado específico utilizando la estructura del AVL (por `Employee_ID`).
4. Función para **eliminar** un registro y reestructurar el árbol en el archivo (por ID de empleado).
5. Función para la **búsqueda por rango** que retorne todos los empleados entre un rango de `Employee_ID` especificado.

P3 (8 puntos): Evaluación de Desempeño

- Medir el tiempo de acceso a memoria secundaria para:
  i. **Inserción de registros**
  ii. **Búsqueda de empleados específicos**
  iii. **Búsqueda por rango de empleados**
  iv. **Eliminación de registros**
- Comparar los tiempos de acceso y **analizar los resultados**, utilizar gráficos.
- Analizar en qué escenarios conviene usar cada método.

3. Entregable

- **Código fuente en Python** con las implementaciones.
- **Informe con los resultados experimentales** incluyendo el analisis y la discución.