

Sequential and Sparse Index File

Organización de Archivos

Profesor Heider Sanchez

P1. Sequential File: Usando como base la siguiente estructura de registro:

```
struct Registro
{
    char codigo [5];
    char nombre [20];
    char carrera [15];
    int ciclo;
};
```

Se le pide diseñar los siguientes algoritmos bajo el esquema de archivo ordenado.

- a) Algoritmo de inserción en bloque el cual recibe como parámetros una lista de registros que deben ser guardados en un archivo de forma ordenada. Para ello seleccione el campo **codigo** como campo de ordenación (search-key).

void insertAll(vector<Registro> registros)

- b) Algoritmo para agregar un nuevo registro al archivo. 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 al campo **nombre**. Asegúrese de mantener los punteros actualizados.

void add(Registro registro)

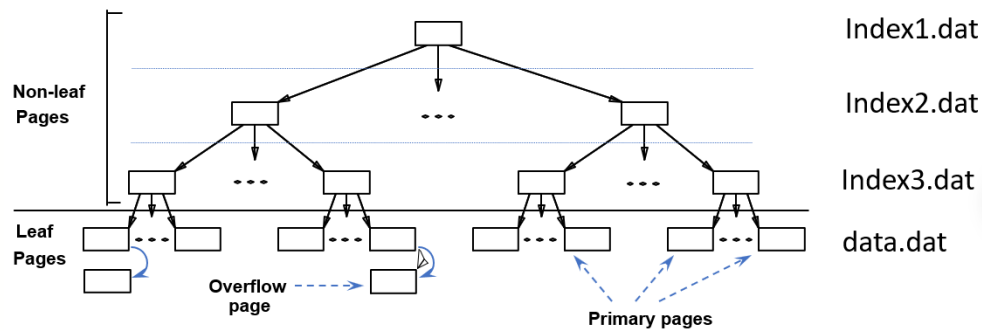
- c) Algoritmo de búsqueda puntual dado una clave de búsqueda (se puede obtener más de un registro):

Registro search(string key)

- d) Algoritmo de búsqueda por rango entre dos claves de búsqueda:

vector<Registro> search(string begin, string end)

P2. ISAM – Sparse Index File: Diseñe los siguientes algoritmos del Multilevel Sparse Index:



- Algoritmo para insertar un nuevo registro
void add(Registro registro)
- Implementar la función de búsqueda.
vector<Registro> search(string key)
- Implemente la búsqueda por rango.
vector<Registro> search(string begin-key, string end-key)

Nota:

- El índice crece hasta el nivel 3. El índice es estático y para nuevas inserciones se producen overflow pages (encadenamiento de páginas).
- Define el factor de bloque tanto en las páginas de datos como en las páginas del índice.

Entregable:

- Diseño e implementación de los algoritmos y el análisis de acceso a memoria secundaria