Video34 (1)

Conceptos Principales

1. a
2. b
3. c

Notas

Resumen general

El video explica cómo funcionan los **archivos binarios de acceso directo e indexado**, su estructura interna, cómo simulan punteros en disco y cómo manejan problemas de **fragmentación** (espacios vacíos al eliminar registros).

---

1. Archivos de acceso directo (organización directa)

Concepto

- Los registros tienen **tamaño fijo**.
- A diferencia del acceso secuencial, **se pueden leer o escribir registros en cualquier posición del archivo** sin recorrer todos los anteriores.
- Se utilizan funciones como `fseek()` para moverse a una posición específica (en bytes).

Ejemplo: Lista enlazada simple en disco

- Cada registro tiene:
  - **Datos (verde)**
  - **Puntero al siguiente (amarillo)**
- El puntero no es una dirección de memoria, sino un **número de byte o posición en disco**.
  - Ejemplo:

```
Registro 1 → posición 54
Registro 2 → posición 72
Registro 3 → posición 36
```

- El valor `1` indica **nulo** (no hay siguiente).

Uso típico

- Cuando los datos son **muy grandes** y no caben todos en memoria.
- Permite **cargar solo una parte** del archivo según se necesite.
- Ideal para operaciones **ocasionales** de lectura o actualización.

🗃️ 2. Archivos de organización indexada

🔹 Estructura

- Se usan **dos archivos**:
  1. **Archivo de datos** → contiene la información real.
  2. **Archivo índice** → indica en qué posición (byte) está cada registro y su tamaño.
- Gracias al índice:
  - Los registros **pueden tener distinto tamaño**.
  - El índice **sí usa registros de tamaño fijo** (por ejemplo, 12 bytes: ID + posición + tamaño).

🔹 Ejemplo práctico

- Similar al sistema de archivos **FAT (File Allocation Table)** de los discos duros:
  - El índice guarda el número de bloque y la ubicación del siguiente bloque del archivo.
  - Existen versiones **FAT16** y **FAT32**, según el tamaño (bits) de los punteros usados.

🧩 3. Manejo de huecos y fragmentación

🔹 Problema

- Al eliminar registros, quedan **espacios libres** (“huecos”) en el archivo.
- Esos huecos deben manejarse para evitar desperdicio de espacio y lentitud.

🔹 Métodos para tratarlos

1. **Marcado simple (método feo pero fácil):**
   - Se marca el espacio como libre (por ejemplo, todo en 0).
   - Al insertar un nuevo registro, se busca desde el inicio el primer hueco disponible.
   - Simple, pero **lento e ineficiente** si hay muchos huecos.
2. **Archivo de huecos (método eficiente):**
   - Se mantiene un **archivo aparte** que guarda las posiciones libres.
   - Cuando se elimina algo, su posición se agrega al archivo de huecos.
   - Al insertar, se revisa si hay huecos disponibles antes de escribir al final.
   - Es **rápido, organizado y escalable**.
3. **Ignorar huecos (método del sistema operativo):**
   - Se siguen agregando registros al final sin preocuparse por los huecos.
   - Luego se realiza un proceso de **desfragmentación**, que:
     - Compacta el archivo.
     - Actualiza todos los punteros.
   - Muy **lento y costoso**, por eso solo se usa en sistemas operativos (como Windows al “desfragmentar disco”).

📊 4. Comparación rápida

Tipo de acceso	Estructura	Ventajas	Desventajas
Secuencial	Bloques consecutivos	Simple y rápido para lecturas ordenadas	No se puede saltar registros
Directo	Registros fijos + punteros en disco	Permite saltar a cualquier posición	Requiere gestión precisa de offsets
Indexado	Archivo de datos + archivo índice	Soporta tamaños variables	Complejo de mantener, más espacio
Con huecos	Maneja espacios libres	Reutiliza espacio	Puede fragmentarse si no se optimiza

🧠 Conclusión

- Los **archivos de acceso directo** permiten moverse por posiciones específicas del disco, simulando punteros.
- Los **archivos indexados** separan datos e índices, facilitando el acceso y la variación de tamaño.
- La **fragmentación** es un problema natural del manejo dinámico de archivos, y existen distintas estrategias (búsqueda, índice de huecos, desfragmentación) para controlarla.