Actividad 1.2
BDD – 25/26
1º DAW – Jhonal Roca

1. **Investigue y explique con sus palabras en qué consiste el método de búsqueda binaria en ficheros.**

- La **búsqueda binaria** es un algoritmo que nos ayuda a realizar una búsqueda de manera eficiente para encontrar un valor específico dentro de un conjunto de datos ordenados. Una de sus **ventajas** a diferencia de la búsqueda secuencial es la velocidad en la que hace la búsqueda, mientras la búsqueda secuencial tiene que ir registro por registro la búsqueda binaria va dividiendo el conjunto de datos cada vez en mitades más pequeñas hasta encontrar el registro deseado.

2. **Realice un ejemplo de búsqueda secuencial y binaria en clase suponiendo que tiene que acceder a un valor dentro de un conjunto ordenado de valores. Compute y compare el número de lecturas en ambos procesos para varios valores de búsqueda.**

- **Búsqueda secuencial:**
  - Vamos a realizar una búsqueda secuencial en un lote que tiene 20 registros y deseamos obtener el registro 15 y el 9.

- [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

  - Como ya sabemos la búsqueda secuencial se basa en ir registro por registro por lo tanto para obtener los registros deseados se han tenido que hacer las siguientes lecturas.
    - Para el registro 15 se hicieron 15 lecturas.
    - Para el registro 9 se hicieron 9 lecturas.

- **Búsqueda Binaria:**
  - Si bien sabemos la búsqueda binaria se basa en dividir el lote e ir eliminando la parte que no

contiene el fichero deseado. Se desea obtener los ficheros 9 y 15.

- [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

- Como ya sabemos cómo funciona la búsqueda binaria vamos a empezar a dividir el lote para encontrar el fichero 9.

    - El elemento central es 10 y buscamos el fichero 9. Como el 9 es menor que 10, se descarta la mitad derecha (del 10 al 20) y buscamos otra vez en el lote del 1 al 9 y como sabemos el elemento central es el 5 por lo que 9 es mayor que 5 por lo que descartamos la mitad izquierda y volvemos a dividir y buscamos entre el 6 y el 9 por lo que el elemento central es el 7 y como 9 es mayor que 7 se elimina la mitad izquierda y buscamos entre el 8 y 9, ahora como el 9 es mayor que el 8 se descarta el 8 y tenemos como resultado el registro 9. **Por lo que el total para encontrar el registro 9 fueron 4 lecturas.**

- Ahora vamos en busca del registro 15.
    - El elemento central es el 10 y buscamos el registro 15. Como el 15 es mayor que 10 descartamos la mitad izquierda y volvemos a buscar (del 11 al 20) buscamos el elemento central que es el 15 y por lo tanto **ya se ha encontrado el registro deseado con tan solo 2 lecturas.**

3. ¿Cuántos índices primarios y de agrupamiento puede tener un fichero ordenado?

- El índice primario es aquel que aparece cuando el fichero esta ordenado por un campo clave único (Un campo que no tenga valores repetidos), ejemplo: el id de una tabla. Como el archivo solo puede estar ordenado por un único campo clave, solo puede haber un índice primario por fichero.

- El índice de agrupamiento es aquel que se da cuando el fichero esta ordenado por un campo no clave (con valores repetidos). En este caso, también el fichero solo puede

estar ordenado por un campo. Ejemplo: agrupar personas que vivan en Ubrique.

4. Comente ventajas e inconvenientes respecto a la actualización de datos en ficheros con organización tipo hash.

- En la organización por hash, la actualización de datos es muy rápida cuando no cambia el campo de dispersión, pero puede ser costosa si este campo se modifica o si existen muchas colisiones que obliguen a reorganizar o encadenar los registros, por eso veremos ventajas e inconvenientes de actualizar datos con hash.

    - **Ventajas:**
        - **Acceso directo y rápido:** Al aplicar la función de dispersión sobre el campo clave, se obtiene directamente la dirección del bloque → las operaciones de actualización (búsqueda y modificación) son muy eficientes.
        - **Actualización localizada:** Si se modifica un registro sin cambiar el valor del campo de dispersión, el acceso es inmediato y no se ve afectado el resto del fichero.
        - **Eficiencia en inserciones:** Insertar un nuevo registro es rápido porque basta con calcular la función hash y colocarlo en el bloque correspondiente (si no hay colisiones).

    - **Inconvenientes:**
        - **Problemas con colisiones:** Cuando dos claves distintas generan la misma dirección, hay que aplicar técnicas de resolución (encadenamiento, exploración lineal, etc.), lo que complica y ralentiza las actualizaciones.
        - **Dificultad si cambia el campo de dispersión:** Si la actualización afecta al campo usado en el hash, el registro debe reubicarse en otra posición → esto implica coste extra de acceso y reorganización.
        - **Inserciones y borrados menos predecibles:** Aunque rápidas en teoría, si hay muchas colisiones o el espacio no está bien

dimensionado, la eficiencia cae y se complica el mantenimiento.

- **No es eficiente para recorridos ordenados:** Si se requiere acceder a los datos en un orden distinto (por ejemplo, por rango), no es adecuado → lo que limita ciertas actualizaciones masivas.

5. **Si tenemos un archivo de datos de 2.000 jugadores con tamaño fijo de 80 bytes y un disco de tamaño de bloque igual a 1.024 bytes, determine el número de bloques requerido y el coste de una búsqueda binaria en cuanto a número necesario de accesos a bloques para encontrar un registro de datos.**

- Dados los datos:
  - Número de registro 2000
  - Tamaño de cada registro 80 bytes
  - Tamaño de bloque de disco 1024 bytes

- Factor de bloqueo
  - FBL = 1024 / 80 = 12,8

  Ahora vamos a realizar una comprobación para determinar el FBL

  80 x 12 = 960 (caben 12)

  80 x 13 = 1040 (No caben 13)

  Por lo que FBL = **12 registros por bloque**

- Calculamos el número de bloques necesarios para 2000 registros:
  - 2000 / 12 = 166
  - 12 x 166 = 1992
  - 2000 – 1992 = 8

  Como hay resto, se necesita un bloque extra por lo que quedaría:

  167 bloques