Preguntas de repaso de FOD:

- Un archivo directo:

1. Acceso directo y orden de búsqueda lineal
Acceso directo significa que puedes ir directamente a cualquier registro sin tener que pasar por todos los registros anteriores.
Orden de búsqueda lineal implica que, para encontrar un elemento, potencialmente necesitas revisar cada elemento uno por uno.
Estas dos características son contradictorias. Si tienes acceso directo, no necesitas revisar los elementos uno por uno, así que la búsqueda no sería lineal sino constante (O (1)).

2. Acceso directo y orden de búsqueda logarítmica
Acceso directo (como se explicó anteriormente) implica acceso inmediato a cualquier registro.
Orden de búsqueda logarítmica generalmente se refiere a estructuras de datos como árboles balanceados, donde la búsqueda se realiza dividiendo repetidamente el espacio de búsqueda en mitades.
La búsqueda logarítmica se asocia más comúnmente con estructuras que requieren algún tipo de acceso jerárquico o dividido, no con acceso directo. Si el acceso es directo, no hay necesidad de un orden logarítmico.

3. Acceso directo y orden de búsqueda exponencial
Orden de búsqueda exponencial implica que el tiempo necesario para encontrar un elemento crece exponencialmente con el tamaño del conjunto de datos.
Esto no tiene sentido en combinación con acceso directo. El acceso directo debe ser eficiente, y un orden exponencial es extremadamente ineficiente.

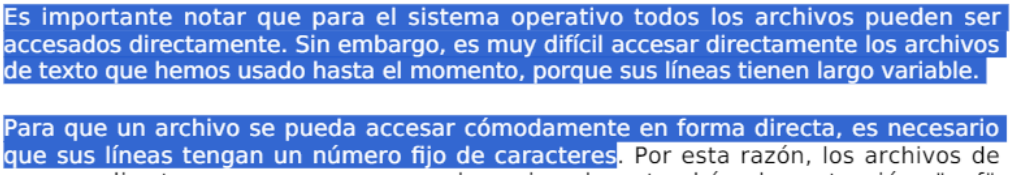
4. Acceso secuencial indizado y orden de búsqueda lineal
Acceso secuencial indizado implica que tienes índices que te permiten localizar ciertas posiciones, y luego accedes secuencialmente desde ahí. Orden de búsqueda lineal sugiere que podrías tener que recorrer secuencialmente a través de un conjunto de datos.
Sin embargo, los índices están diseñados para mejorar la eficiencia del acceso secuencial, por lo que la combinación de acceso secuencial indizado y búsqueda lineal no tiene mucho sentido. El propósito de los índices es precisamente evitar una búsqueda lineal completa.
Así que ninguna respuesta es correcta.

- Si se realiza a baja de un registro en un archivo directo:

b. Es más eficiente una baja lógica que física considerando el espacio ocupado. No es cierto esto ni ninguna de las anteriores ya que:
En términos de espacio ocupado, baja física es más eficiente porque libera el espacio inmediatamente.
Baja lógica puede ser más eficiente en términos de tiempo y simplicidad de implementación, pero a costa de un mayor espacio ocupado y posible fragmentación lógica a lo largo del tiempo.
La elección entre baja lógica y baja física depende del contexto y de las prioridades específicas de rendimiento y gestión de espacio de tu aplicación.

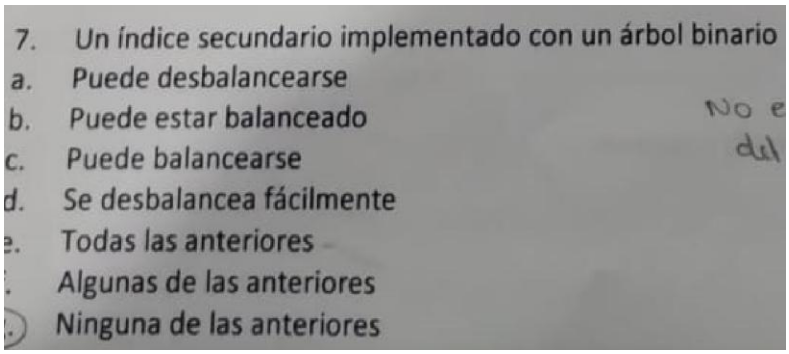
- Un proceso de merge de n archivos, para que sea lo más eficiente posible en términos de rendimiento se debe aplicar con todos los archivos ordenados por EL MISMO (cuidado porque ponen “algunos”) criterio.

- En la pregunta de archivo directo con registros de longitud variable:



- Para realizar un archivo de bajas (eliminar registros) sobre un archivo de datos, no es estrictamente necesario que los registros estén ordenados por algún criterio específico. La capacidad de realizar operaciones de eliminación generalmente depende más del método de acceso y manipulación de datos que del ordenamiento de los registros. Aquí te explico más detalladamente.

- Las claves primarias SÍ identifican inequívocamente un elemento del archivo.

- 

  En este son respuestas del árbol, pero ninguna del índice. Por eso ninguna es cierta.

- Índices de archivos con registros de longitud variable.

Los registros de longitud fija en los índices proporcionan una estructura optimizada y eficiente para facilitar operaciones rápidas y consistentes en la gestión de datos, siendo esta la razón principal por la cual se prefieren en la mayoría de las implementaciones.

La presencia de un índice en un archivo no está directamente relacionada con la longitud de los registros que contiene.

1. **Árboles B y B+**:
   - **Árbol B**: Es una estructura de datos en la que cada nodo puede contener múltiples claves y múltiples hijos. Los datos pueden estar tanto en los nodos internos como en las hojas.
   - **Árbol B+**: Es una variante del árbol B en la que los datos están solo en las hojas, no en los nodos internos. Las hojas están enlazadas, facilitando las búsquedas secuenciales y el recorrido de todos los datos.
2. *Árboles B y B+**:
   - **Árbol B***: Es otra variante del árbol B, diseñada para mejorar el rendimiento al permitir una mayor capacidad de almacenamiento en los nodos internos y una mayor cantidad de punteros a hijos.
   - **Árbol B+**: Como se mencionó anteriormente, este tipo de árbol es una versión específica del árbol B que optimiza el acceso secuencial y el rendimiento de búsqueda.

Aclaración sobre la Imposibilidad de Ser Ambos Tipos Simultáneamente:

- **Árbol B y B+**: Cada uno tiene características únicas y diferentes en términos de cómo manejan los datos y estructuran los nodos. No pueden ser "B y B+" al mismo tiempo porque implicaría combinar características mutuamente excluyentes, como tener datos en los nodos internos y solo en las hojas.
- *Árbol B y B+**: Similarmente, no pueden ser ambos a la vez porque cada uno tiene estructuras y características específicas diseñadas para propósitos distintos.

La saturación progresiva encadenada es una técnica utilizada en el contexto de tablas hash para manejar colisiones. Aquí te explico cómo se aplica:

- **Opción 2: En caso de colisión y saturación.** Si hay saturación es porque hay colisión. SE DEBE usar en saturación.

Explicación:

- **Colisión**: Se refiere al caso en el que dos claves distintas producen el mismo valor hash y deben ser almacenadas en la misma posición de la tabla hash.
- **Saturación**: Se refiere al llenado de las posiciones de la tabla hash con datos.

Una función de hash uniforme y aleatoria distribuye las claves de forma equitativa y aleatoria entre las posiciones de una tabla hash. Esto significa que cada posición tiene la misma probabilidad de contener cualquier clave.

Cuando buscas una clave:

La función de hash te dice dónde buscar directamente en la tabla.

Debido a la distribución uniforme, en promedio solo necesitas un acceso directo para encontrar la clave, porque las claves están bien distribuidas por toda la tabla.

En resumen, la función de hash uniforme y aleatoria hace que el tiempo promedio de búsqueda sea aproximadamente 1 acceso, porque distribuye las claves de manera justa y aleatoria en la tabla hash.

- Densidad de empaquetamiento en cuanto a hashing dinámico.

El hashing estático tiene una densidad de empaquetamiento que varía con la cantidad de elementos insertados, mientras que el hashing extensible permite ajustes dinámicos para mantener una densidad de empaquetamiento óptima y estable.

- Inserción en hashing estático.

Cuando se produce una inserción de hash estático, sobre el archivo de datos puede haber más de una operación de lectura y escritura. Se realiza al menos UNAoperación de cada una.

- Colisión con hashing dinámico.

Si hubo colisión con hashing dinámico, PUEDE ocurrir overflow o no. No siempre, quizás había un registro previo entonces la clave inserta al lado.

- Un algoritmo de búsqueda en un archivo:

Aunque la opción a ("Es más eficiente si el archivo está ordenado") podría parecer correcta, la opción b ("Puede ser más eficiente si se considera como precondición que el archivo está ordenado") es más precisa, ya que reconoce que la eficiencia depende de la asunción de que el archivo está ordenado, no simplemente de si está ordenado o no en general.

- El proceso de alta de un registro por ajuste óptimo:

El proceso de alta de un registro por ajuste óptimo no está directamente relacionado con si los registros son de longitud fija o variable. Más bien, se refiere a la estrategia utilizada para insertar registros de manera que se optimice el uso del espacio disponible en la estructura de datos.

- La operación assign vincula el archivo lógico con el archivo físico.

La operación "assign" vincula el nombre lógico que un programa utiliza para referirse a un archivo con su ubicación física en el sistema de almacenamiento. Esto permite al programa acceder y manejar correctamente los datos del archivo.

- **Número de Registro Relativo (NRR)**: Es un número que indica la posición relativa de un registro dentro de un archivo. Este número se utiliza para identificar y acceder rápidamente a un registro específico dentro de una secuencia de registros almacenados en un archivo.

Opciones dadas:

a. Se puede acceder a un registro de un archivo fragmentado en un sólo acceso: Esto puede ser posible si el archivo fragmentado utiliza un esquema de direccionamiento que permita acceso directo a registros específicos utilizando el NRR como referencia.

b. Se puede acceder a un registro de un archivo no fragmentado en un sólo acceso: En archivos no fragmentados, donde los registros están almacenados de manera contigua, se puede acceder directamente a un registro utilizando su NRR si se implementan métodos de acceso directo o mediante índices.

c. Se puede acceder a un registro de un archivo en un sólo acceso: Esto generalmente se refiere a la capacidad de acceder a cualquier registro del archivo de manera eficiente, lo cual puede lograrse con técnicas como el direccionamiento directo utilizando el NRR.

d. Se puede lograr acceso directo a un archivo: Acceder directamente a un archivo significa poder ubicar y recuperar rápidamente registros individuales utilizando métodos que aprovechen el NRR para evitar búsquedas secuenciales.

Todas son ciertas.

- Un "archivo serie" puede ser:

  Un archivo que almacena datos de manera secuencial, uno tras otro, sin un orden específico impuesto.

  Un archivo que organiza los datos en orden cronológico, basado en el tiempo de registro.

  Un archivo que simplemente contiene datos almacenados en una secuencia, sin necesariamente seguir un orden específico definido por criterios como el tiempo o un índice.

Así que puede ordenarse, pero no requiere estarlo o no.

- Dado un archivo con 1000 registros…

No siempre se puede llevar a memoria RAM para hacer búsquedas más eficientes es la respuesta correcta.

Las otras respuestas son demasiado absolutas como para ser ciertas. No es siempre posible lo que indican:
1. Siempre se puede llevar a memoria RAM para hacer búsquedas más eficientes
2. Siempre se puede realizar búsqueda dicotómica
3. No puede realizarse búsqueda dicotómica

- Un archivo ordenado:

Lo mismo sucede con esto. Puede desordenarse es la respuesta correcta; no tiene porqué convenir mantenerse ordenado, ni tampoco no conviene ordenarlo. Además, sí puede desordenarse.

- Cuál de las siguientes definiciones corresponde a archivo

Para mí, la respuesta según la definición de cátedra es: “Colección de registros que abarca un conjunto de entidades con ciertos aspectos en común para un propósito particular” y “Colección de registros del mismo tipo almacenados en un dispositivo de memoria secundaria”.

Puede ser una colección de registros semejantes almacenados en disco rígido porque el almacenamiento secundario no solo hace referencia a disco rígido. Sin embargo, como solo una respuesta es correcta y dos ya lo son, considero que d es correcto.

- Un archivo que maneja registros de longitud fija necesita:

No se necesita un indicador de longitud porque la longitud de cada registro es constante y predefinida. Debido a esta característica, no es necesario utilizar delimitadores ni indicadores de longitud, ya que la posición de cada campo y cada registro se puede calcular directamente conociendo la longitud fija de los mismos. Por lo tanto, nada de esto es correcto.

- El proceso de baja en un archivo con registros de longitud variable:

Puede recuperar el espacio disponible con nuevas altas:

En registros de longitud variable, cuando se elimina un registro, puede quedar un espacio vacío que podría ser reutilizado por nuevos registros si estos caben en ese espacio.

Puede recuperar el espacio disponible compactando periódicamente el archivo:

Este es un método común para manejar el espacio disponible. La compactación periódica reorganiza los registros, eliminando los espacios vacíos y consolidando los registros restantes para recuperar espacio.

Puede recuperar el espacio disponible compactando el archivo ante cada baja:

Compactar el archivo cada vez que se elimina un registro es técnicamente posible, pero suele ser ineficiente debido al costo computacional de reordenar los registros con cada eliminación.

Todas son correctas.

- El procedimiento de alta de información en un archivo:

a. Siempre agrega información al final del archivo:

No es siempre cierto, ya que algunos sistemas reutilizan el espacio dejado por registros eliminados.

b. Puede recuperar espacio dado de baja físicamente:

Si la baja es física, el espacio ya se ha eliminado definitivamente y no se puede reutilizar directamente.

c. Siempre recupera espacio dado de baja lógicamente:

La baja lógica marca los registros como eliminados, permitiendo que el espacio sea reutilizado posteriormente, pero no necesariamente siempre.

d. Ninguna de las anteriores:

Esta es la correcta.

- El proceso de baja lógica (sin ningún agregado de otras operaciones) de un archivo:

La baja lógica marca los registros como eliminados, pero no elimina físicamente los datos del disco. Esto significa que el espacio ocupado por estos registros no se libera inmediatamente para su reutilización. La baja lógica deja los registros en el archivo hasta que se realice una operación de compactación u otro tipo de mantenimiento para recuperar ese espacio.

- Dado un archivo

a. Siempre se necesita tener un índice asociado:

Esto no es cierto. No todos los archivos necesitan un índice asociado. Los índices se utilizan para optimizar la búsqueda y recuperación de datos, pero no son estrictamente necesarios para el funcionamiento básico de un archivo.

b. Un índice asociado le permite optimizar las operaciones de alta:

Un índice asociado generalmente se utiliza para optimizar las operaciones de búsqueda, lectura y, en algunos casos, las operaciones de actualización y eliminación. Sin embargo, no necesariamente optimiza las operaciones de alta (inserción). De hecho, mantener índices puede agregar una sobrecarga a las operaciones de inserción, ya que el índice también debe actualizarse.

c. Siempre debe estar ordenado:

Esto no es cierto. No todos los archivos necesitan estar ordenados. Algunos archivos pueden ser simplemente secuenciales sin ningún orden específico, dependiendo de los requisitos de la aplicación.

- Una clave candidata

a. Admite repeticiones de valores:

Esto no es cierto. Una clave candidata debe tener valores únicos para poder identificar registros de manera unívoca.

b. Admite repeticiones de campos:

Esto tampoco es cierto. Los campos que conforman una clave candidata deben ser únicos.

c. Podría haber sido elegida clave primaria:

Esta es cierta. Una clave candidata es un conjunto de campos que pueden identificar registros de manera única y, por lo tanto, podría haber sido elegida clave primaria.

d. Tiene exactamente las mismas características que la clave primaria:

Una clave candidata no es la clave primaria, pero tiene las características necesarias para serlo.

- El proceso de compactación de un archivo tiene sentido ser aplicado:

a. Luego de realizar una operación de alta:

No es necesario compactar después de una operación de alta, ya que agregar datos no deja espacios vacíos en el archivo que necesiten ser eliminados.

b. Luego de realizar una operación de baja lógica:

Esto tiene sentido. La baja lógica marca registros como eliminados, pero no libera el espacio que ocupan en el archivo. Compactar el archivo después de varias bajas lógicas consolidará los registros y liberará espacio.

c. Luego de realizar una operación de baja físicamente:

En este caso, los registros ya se han eliminado físicamente y el espacio se ha liberado, por lo que no es necesario compactar el archivo.

d. Luego de realizar una operación de modificación:

Las modificaciones no necesariamente dejan espacios vacíos a menos que involucren cambios significativos en el tamaño de los registros. Por lo general, no se necesita compactar después de una modificación.

- El acceso promedio para recuperar un dato en un archivo desordenado

a. Tiene orden lineal -Esta es la correcta**-**

En un archivo desordenado, el acceso promedio generalmente implica revisar cada registro secuencialmente hasta encontrar el dato buscado. Esto es un orden lineal, O(n), donde n es el número de registros en el archivo.

b. Tiene orden logarítmico:

Un acceso logarítmico, O (log n), sería característico de estructuras de datos ordenadas como árboles de búsqueda balanceados (por ejemplo, árboles AVL o árboles B). No es aplicable a archivos desordenados.

c. Tiene orden constante (1):

El acceso constante, O (1), implica que el tiempo de acceso no depende del tamaño de los datos. Esto solo sería posible en casos donde se tiene un acceso directo mediante un índice o una tabla hash, pero no es común en archivos desordenados.

d. No tengo datos suficientes para responder:

A menos que se conozca el método específico de acceso (como un índice o estructura de datos utilizada), es posible que no se pueda determinar el orden de acceso de manera definitiva. Sin embargo, en archivos desordenados típicos, el acceso promedio tiene un orden lineal.

- **La política de primer ajuste** es un método utilizado en la gestión de memoria y almacenamiento de archivos para asignar y recuperar espacio después de que se han eliminado registros o datos.

a. Sólo se aplica en registros de longitud fija: Esto no es cierto. El primer ajuste puede aplicarse tanto en registros de longitud fija como en registros de longitud variable.

b. Sólo se aplica en registros de longitud variable: Esto tampoco es cierto. La política de primer ajuste no está restringida por la longitud de los registros.

c. Genera fragmentación interna: **-Esto es correcto-.** La fragmentación interna ocurre cuando el espacio asignado a un registro es más grande que el registro mismo, dejando espacio no utilizado dentro de ese bloque. Ahí sería propicio recuperar espacio borrado.

d. Genera fragmentación externa: **-Esto también es correcto-.** La fragmentación externa ocurre cuando hay suficiente espacio total disponible para satisfacer una solicitud, pero está dividido en bloques más pequeños que no están contiguos y no se pueden usar juntos.

- A partir de un archivo con registros de long fija y luego de algunas operaciones con el mismo:

a. Nunca genera fragmentación:

Esta afirmación no es cierta. Incluso en archivos con registros de longitud fija, la eliminación de registros puede llevar a la fragmentación, ya sea interna o externa.

b. Puede generar fragmentación interna:

Esto es posible. La fragmentación interna ocurre cuando el espacio asignado a un registro es más grande que el registro mismo, dejando espacio no utilizado dentro de ese bloque.

c. Puede generar fragmentación externa:

Incorrecto en el contexto de registros de longitud fija. La fragmentación externa no suele ser un problema en archivos con registros de longitud fija, **ya que no hay bloques de espacio libre divididos en fragmentos pequeños.**

Fragmentación Interna: Sucede cuando el espacio asignado a un registro es mayor que el tamaño real del registro. Por ejemplo, si tenemos un registro de 100 bytes y solo utilizamos 50 bytes de ese espacio, los otros 50 bytes dentro de ese bloque quedan como espacio no utilizado.

- El acceso secuencial a un archivo es:

a. Acceso a los registros uno tras otro y en el orden físico en el que están guardados:

Esta definición describe correctamente el acceso secuencial. Cuando se accede secuencialmente a un archivo, se lee cada registro en el orden en que están almacenados físicamente en el archivo, uno tras otro.

b. Acceso a los registros de acuerdo con el orden establecido por otra estructura:

Esto describe más bien el acceso basado en algún tipo de índice o estructura adicional que organiza los registros de manera no necesariamente secuencial.

c. Acceso a un registro determinado sin necesidad de haber accedido a los predecesores:

Esto describe el acceso directo o aleatorio, no el acceso secuencial.

- Un borrado lógico en un archivo de datos

a. Recupera inmediatamente el espacio borrado, dejando el archivo de tamaño menor:

Esto no es cierto para el borrado lógico. El borrado lógico marca los registros como eliminados, pero no libera el espacio físico inmediatamente. Por lo tanto, el tamaño del archivo no disminuye de inmediato.

b. No se puede aplicar con registros de longitud variable:

Esto no es correcto. El borrado lógico puede aplicarse tanto a registros de longitud fija como a registros de longitud variable. La capacidad de realizar borrado lógico no está limitada por el tipo de longitud de los registros.

c. Sólo se aplica con registros de longitud fija:

Esto es incorrecto por la misma razón que la opción b. El borrado lógico puede aplicarse a ambos tipos de registros.

d. Permite recuperar el espacio con nuevas altas:

Esta es la afirmación correcta. Aunque el borrado lógico no libera espacio físico de inmediato, sí permite que el espacio marcado como borrado lógicamente sea reutilizado para nuevas inserciones (altas), dependiendo de cómo se implemente el archivo y se maneje el espacio.

- Un archivo físicamente ordenado

1. **Permite búsqueda binaria sólo si las altas mantienen el archivo ordenado:**

Esta afirmación es correcta. Para mantener la capacidad de realizar búsquedas binarias eficientes, es crucial que las operaciones de alta mantengan el orden del archivo. Si las altas no se realizan de manera que mantengan el orden, la búsqueda binaria podría no ser efectiva.

2. Es más fácil de recorrer:

La facilidad de recorrido también puede depender de otros factores como la disponibilidad de índices adecuados.

3. Permite búsqueda binaria: Si el primero es cierto, se contradicen.
4. No permite búsqueda binaria si hay bajas lógicas:
   Las bajas lógicas no afectan directamente la capacidad de realizar búsqueda binaria en un archivo físicamente ordenado, ya que la estructura de ordenamiento inicial de los registros no se ve alterada por estas operaciones.

- En un archivo con registros de longitud variable

  a. Puede utilizar "$" como delimitador de fin de registro:
  Esto es correcto.
  b. Cuando un registro se modifica utiliza el mismo espacio:
  Esto no siempre es cierto. La modificación de un registro de longitud variable puede resultar en un uso diferente del espacio. Por ejemplo, si se modifica para reducir su tamaño, el espacio ocupado puede disminuir. La respuesta b no es siempre verdadera.
  c. Se puede utilizar cualquier lugar libre de un archivo para insertar un registro:
  Es necesario que haya espacio disponible en el archivo para insertar el nuevo registro. Si hay suficiente espacio libre contiguo o fragmentado, se puede insertar un nuevo registro de longitud variable en cualquier lugar del archivo.
  d. Siempre se utiliza la política de mejor ajuste para recuperar espacio:
  Esto no es cierto. La política de recuperación de espacio puede variar según la implementación del sistema de archivos. No siempre se utiliza la política de mejor ajuste; otras políticas como el primer ajuste, peor ajuste, o políticas específicas del sistema pueden ser aplicables.

- Para aplicar un algoritmo de merge

  a. Es necesario más de un archivo: -Esto es correcto-, ya que el algoritmo de merge combina dos o más conjuntos de datos en uno solo.

  b. Es necesario que el archivo maestro esté ordenado: No es estrictamente necesario que el archivo maestro esté ordenado. La ordenación del archivo maestro puede depender del algoritmo específico de merge utilizado, pero no es un requisito absoluto en todos los casos.

  c. Es necesario que los detalles estén ordenados: No es siempre necesario que los detalles estén ordenados. Algunos algoritmos de merge pueden funcionar eficientemente con detalles desordenados, aunque la ordenación puede mejorar la eficiencia del proceso en ciertos casos.

  d. No es necesario que el archivo maestro esté ordenado: -Esto es correcto-. No es imprescindible que el archivo maestro esté ordenado para aplicar un algoritmo de merge.

  e. No es necesario que los archivos detalles estén ordenados: -Esto es correcto-. Para aplicar un algoritmo de merge, no siempre es necesario que los archivos detalles estén ordenados. Algunos algoritmos pueden manejar eficientemente datos desordenados durante el proceso de fusión.

- La función EOF

  a. Puede devolver verdadero después de un Reset:

  Esto es cierto. Después de hacer un Reset (o reinicio) en la lectura de un archivo, EOF puede devolver verdadero si se ha alcanzado el final del archivo después de haberse leído completamente.

  b. Puede devolver falso después de una lectura:

  Esto es correcto. Después de una operación de lectura, EOF puede devolver falso si aún no se ha alcanzado el final del archivo y hay más datos por leer.

  c. Puede devolver verdadero después de una lectura:

  Esto es correcto. Si la lectura de un archivo ha alcanzado el final durante la operación de lectura, EOF puede devolver verdadero para indicar que ya no hay más datos para leer.

  d. Devuelve verdadero si estás al final del archivo:

  Esto es correcto. EOF devuelve verdadero cuando el puntero de lectura ha alcanzado el final del archivo, indicando que no hay más datos válidos para leer.

  e. Devuelve falso si no estás al final del archivo:

  Esto es correcto. Mientras haya datos disponibles para leer en el archivo, EOF devolverá falso para indicar que no se ha llegado al final.

- Es posible aplicar la búsqueda binaria en

a. Archivos desordenados con registros de longitud variable: Incorrecto. La búsqueda binaria no puede aplicarse en archivos desordenados.

b. Archivos desordenados con registros de longitud fija: Incorrecto por la misma razón que la opción anterior.

c. Archivos ordenados con registros de longitud variable: Incorrecto. En archivos con registros de longitud variable, no se puede garantizar que se pueda acceder de manera directa a cada elemento debido a la variabilidad en la longitud de los registros. Esto significa que la búsqueda binaria no puede aplicarse eficazmente en estos casos porque no se puede dividir el conjunto de datos de manera uniforme para realizar búsquedas precisas.

d. Archivos ordenados con registros de longitud fija: Correcto. En archivos ordenados con registros de longitud fija, la búsqueda binaria es perfectamente aplicable porque se puede acceder a cada registro de manera directa utilizando un índice.

- Un archivo organizado con registros de longitud variable

a. No permite realizar bajas lógicas:

Esto es incorrecto. Un archivo organizado con registros de longitud variable puede permitir operaciones de baja lógica, donde un registro se marca como eliminado lógicamente, pero permanece físicamente en el archivo. Esto ayuda a mantener la integridad de los datos sin necesidad de eliminar físicamente el registro.

b. Optimiza la utilización de espacio en disco:

Esto es cierto. Los archivos con registros de longitud variable pueden optimizar el uso del espacio en disco al permitir que los registros se expandan o contraigan según sea necesario, en lugar de reservar espacio fijo para cada registro como en los archivos de longitud fija.

c. No permite realizar bajas físicas:

Esto es incorrecto. En un archivo con registros de longitud variable, se pueden realizar bajas físicas donde el espacio ocupado por un registro eliminado se libera para su reutilización. Esto implica la eliminación física del registro del archivo.

d. Sólo acepta altas al final del archivo:

Esto es incorrecto. Aunque algunos sistemas de archivos pueden preferir agregar nuevos registros al final por eficiencia, no es una restricción inherente de los archivos con registros de longitud variable. Se pueden agregar nuevos registros en cualquier lugar donde haya espacio disponible en el archivo.

- La eficiencia promedio de búsqueda en un archivo sin orden es

En un archivo sin orden específico, la eficiencia promedio de búsqueda suele seguir un orden lineal. Esto significa que, en promedio, la búsqueda requerirá recorrer aproximadamente la mitad de los registros para encontrar el dato deseado, ya que no hay ninguna estructura o criterio que permita acotar la búsqueda más rápidamente.

- Una colisión

a) Siempre genera saturación: Esto es incorrecto. La saturación en archivos generalmente se refiere a la ocupación total de espacio disponible en disco o memoria, lo cual no es necesariamente generado por una operación específica como la de alta de registros.

b) Requiere utilizar saturación progresiva: Esta afirmación no tiene sentido claro en el contexto proporcionado. "Saturación progresiva" no es un término estándar en relación con operaciones de archivos o bases de datos.

c) Puede producir saturación**: Esto es cierto**. La operación de alta de registros en un archivo puede eventualmente llevar a la saturación del espacio disponible si no se gestiona adecuadamente la asignación y liberación de espacio.

d) Más de una es correcta: Esta opción es incorrecta, ya que hasta el momento solo hemos identificado una afirmación que podría ser cierta.

- Un índice secundario asociado a un archivo:

1. Siempre debe referenciar al índice primario: Esto no es necesariamente cierto. Un índice secundario puede referenciar directamente a los registros del archivo sin necesidad de pasar por el índice primario. Por lo tanto, esta afirmación es incorrecta.

2. Siempre debe permitir acceder a los registros del archivo: Esto es incorrecto. Dependiendo de cómo esté estructurado el índice secundario, puede o no proporcionar acceso directo a todos los registros del archivo.

3. A veces debe permitir acceder a los registros del archivo: Esto es correcto. La utilidad de un índice secundario puede variar según el diseño y el propósito del índice. A veces puede permitir el acceso directo a registros y otras veces puede requerir acceso a través del índice primario.

4. Puede referenciar al índice primario: ¿¿¿¿Esto puede ser????

- La política de Mejor Ajuste

Fragmentación Interna: No ocurre en registros de longitud fija porque cada registro se almacena en un bloque de tamaño fijo y siempre utiliza todo el espacio asignado. No hay espacio no utilizado dentro del bloque.

Fragmentación Externa: Tampoco es común en registros de longitud fija si se gestiona adecuadamente el espacio entre los bloques de registros. Esto significa que, si los bloques de registros están contiguos y no hay espacios libres desperdiciados entre ellos, no hay fragmentación externa.

En archivos con registros de longitud fija, la fragmentación es mínima o no existe, siempre y cuando se gestione correctamente el espacio entre los bloques de registros.

Registros de Longitud Fija: En estos archivos, cada registro ocupa un espacio fijo predefinido. No hay variabilidad en el tamaño de los registros, por lo tanto, no hay necesidad de buscar el "mejor ajuste" para un tamaño específico. Los bloques asignados a cada registro se utilizan por completo sin dejar espacio no utilizado (fragmentación interna). Además, la fragmentación externa no es un problema significativo si los bloques de registros están contiguos y bien gestionados.

Política de Mejor Ajuste: Esta política se usa típicamente en contextos donde los registros tienen longitudes variables y se busca el bloque más pequeño que pueda contener un registro específico para minimizar la fragmentación interna y externa. En registros de longitud fija, no se aplicaría esta política porque no hay variabilidad en el tamaño de los registros que requiera una optimización de ajuste.

- Con respecto a un archivo con registros de longitud fija: ¿Cuáles de estas alternativas son correctas?

a. **Utilizar NRR para tener acceso directo**: NRR (Relative Record Number) se utiliza comúnmente para acceder directamente a registros en archivos secuenciales indexados. Sin embargo, en archivos de registros de longitud fija, donde cada registro tiene un tamaño fijo y están almacenados uno tras otro, NRR no es necesario ni comúnmente utilizado para acceso directo. Por lo tanto, esta afirmación es incorrecta para archivos de longitud fija.

b. **Utilizar un indicador de longitud al inicio de cada registro**: En archivos de longitud fija, todos los registros tienen la misma longitud predeterminada. No es necesario tener un indicador de longitud al inicio de cada registro porque la longitud es conocida de antemano. Este enfoque se utiliza más comúnmente en archivos de longitud variable para determinar la longitud de cada registro. Por lo tanto, esta afirmación no es correcta para archivos de longitud fija.

c. **Utilizar un segundo archivo con la información de la dirección del byte de inicio de cada registro**: Esta técnica podría ser utilizada para implementar acceso directo a registros en archivos de longitud fija. Un segundo archivo que almacena las direcciones de inicio de cada registro podría facilitar el acceso directo a registros específicos sin necesidad de recorrer el archivo secuencialmente. Por lo tanto, esta afirmación es correcta como una técnica válida para acceso directo en archivos de longitud fija.

d. **Ninguna de las anteriores**: Esta opción no es correcta porque la afirmación c es válida como una técnica para acceso directo en archivos de longitud fija.

- La operación seek en un archivo

Permite posicionarse directamente al final del archivo: Esto es correcto. La operación seek permite mover el puntero de posición al final del archivo, lo que es útil para realizar operaciones como añadir nuevos datos al final del archivo o consultar su tamaño total.

Permite posicionarse directamente en el primer registro del archivo: Esto es correcto también. Con seek, se puede mover el puntero de posición al inicio del archivo, lo que facilita comenzar a leer desde el primer registro o realizar operaciones al principio del archivo.

Permite conocer la posición actual dentro del archivo: Esto también es correcto. La operación seek permite no solo mover el puntero a ubicaciones específicas como el inicio o el final del archivo, sino también obtener la posición actual del puntero dentro del archivo.

- Con respecto a los buffers de E/S:

Ocupan lugar en memoria RAM: Esto es correcto. Los buffers de entrada/salida (E/S) son áreas de memoria reservadas en RAM para almacenar temporalmente datos que se están leyendo desde un dispositivo de entrada o que se están escribiendo hacia un dispositivo de salida. Estos buffers ayudan a mejorar la eficiencia al reducir la frecuencia con la que se accede al dispositivo de E/S directamente.

El SO está encargado de manipularlos: Esto también es correcto. El sistema operativo (SO) es responsable de gestionar los buffers de E/S. Esto incluye la asignación de memoria para los buffers, la gestión de su contenido y la sincronización de las operaciones de lectura/escritura entre los buffers y los dispositivos de E/S correspondientes.

Mejoran la performance en lecturas y escrituras: Esto es correcto. Los buffers de E/S actúan como almacenamiento intermedio entre los dispositivos de E/S (como discos duros o redes) y la memoria principal (RAM). Esto ayuda a mejorar el rendimiento al reducir los tiempos de espera asociados con las operaciones de E/S, ya que permite que los datos se transfieran en bloques más grandes y se gestionen de manera más eficiente.