Fundamentos de Base de Datos, Silberschatz, Korth. Capitulos 11 y 12 – desde pag 263

Fundamentos de Sistemas de Base de Datos Elmasry y Navathe, Wesley, tercera edición. Capítulos 5 y 6 – desde pag 63

1. Comente las ventajas y desventajas de utilizar
   1. Un archivo no ordenado
   2. Un archivo ordenado
   3. Un archivo con direccionamiento calculado estático con cubetas (cajones) y encadenamiento.

Para cada caso indique que operaciones resultan más eficientes con respecto a los tiempos de acceso y al espacio utilizado.

1 – Archivos de registros no ordenados (montículo):

Ventajas:

-Sirve para reunir y almacenar registros de datos para el futuro.

-La inserción de un registro es muy eficiente.

-Se puede organizar por extendida y no extendida, además de poder tener los registros tanta longitud fija como variable.

-(longitud fija y no extendidos) fácil de tener acceso a cualquier registro por su posición.

Desventajas:

-La búsqueda de un registro en el archivo es un proceso muy costoso por ser secuencial.

-La eliminación de un registro deja espacio desocupado, si se hace con una gran cantidad de registro se desperdicia espacio.

-(longitud variable) Al modificar requiere eliminar un registro antiguo y la inserción del registro modificado, porque este es posible que no quepa en el espacio que ocupa.

Operaciones Eficientes en acceso y espacio:

-**La inserción**: Utilizando el último bloque del archivo en disco se copia en un buffer, se añade el nuevo registro, y se reescribe el bloque en el disco.

Archivos de registros ordenados:

Ventajas:

-La lectura de los registros en orden según los valores del campo de ordenación son más eficientes que los no ordenados.

-Encontrar el registro que sigue al actual en orden según el campo de ordenación casi nunca requiere accesos a bloques adicionales, porque el siguiente registro está en el mismo bloque(a excepción que este en el último registro del bloque).

-Si usamos una condición de búsqueda basada en el valor de un campo clave de ordenación, se tiene un acceso más rápido mediante técnica de búsqueda binaria.

(Búsqueda binaria tiene acceso a log2(b) bloques, se tiene acceso a (b/2) bloques).

-Las búsquedas que tiene las condiciones <=,<,>=,>.

-La lectura de los registros del archivo en orden según el campo de ordenación es muy eficiente (si se hace omiso de los registros de desborde).

-Si se utiliza índice primario, que es un camino de acceso adicional que permite mejorar el tiempo de acceso aleatorio sobre un campo clave.

Desventajas:

-El ordenamiento no ofrece ninguna ventaja para el acceso aleatorio u ordenado a los registros con base en los valores de un campo.

-La inserción y la eliminación de registros son operaciones costosas.

Operaciones Eficientes en acceso y espacio:

-**Lectura**: Los registros en orden según los valores del campo de ordenación resulta en extremo eficiente, ya que no es necesario ordenarlos adicionalmente.

-**Búsqueda:** Si usamos una condición de búsqueda basada en el valor de un campo clave de ordenación, tendremos acceso más rápido mediante la técnica de búsqueda binaria.

-**Modificaciones (Depende):** Si la condición de búsqueda se basa en el campo clave de ordenación, podremos localizar el registro realizando una búsqueda binaria.

Archivo con direccionamiento calculado estático con cubetas (cajones):

Ventajas:

-Menor número de colisiones usando cubetas.

-Ofrece acceso más rápido posible para obtener un registro arbitrario.

-Adecuada para leer registros en orden.

Desventajas:

-(acceso a registro arbitrario) No resulta muy util cuando se requiere otras aplicaciones del mismo archivo.

-La cantidad fija de espacio asignada al archivo dependiendo la cantidad de registros, se puede desaprovechar el espacio y si se excede abra una gran cantidad de coaliciones.

-La búsqueda de un registro con base en el valor de un campo distinto del de dispersión es tan costosa como en el casi del no ordenado.

Operaciones Eficientes en acceso y espacio:

-**Modificación:** Si la condición de búsqueda es una comparación de igualdad sobre el campo de dispersión, la localización del registro es muy eficiente.

-**Eliminación:** se puede implementar sacando el registro de su cubeta. Si ésta tiene una cadena de desborde, podemos pasar uno de los registros de desborde a la cubeta para reemplazar el registro eliminado. Esto se logra con Y facilidad manteniendo una lista enlazada de posiciones de desborde desocupadas.

Archivos por encadenamiento:

Ventajas:

- Se agrega un campo apuntador a cada posición de registro.

- Se resuelve las coaliciones.

- Son más simples

- Mantienen las tablas de dispersión ocupadas en un 70 y 90 porciento.

Desventajas:

-Requiere su propio algoritmo para insertar obtener y eliminar.

-Los algoritmos de eliminación son bastante complicados.

Operaciones Eficientes en acceso y espacio:

-**La inserción**: Se lleva a cabo colocando el nuevo registro en una zona libre del fichero y reajusta los apuntadores.

-**La modificación (si es de longitud fija):** La reescritura se realiza en la zona del fichero si los registros son de longitud fija.

-**Eliminación:** Tan solo se supone el reajuste de los apuntadores, se pueden aprovechar los espacios desocupados (si se recuperan).

1. ¿Cuándo es preferible utilizar un índice denso en vez de un índice disperso?

Es preferible utilizar índice denso cuando lo que se busca es la velocidad de procesamiento, ya que en índice denso se sigue el puntero que va directo al primer registro con la clave de búsqueda. Se procesa el registro y se sigue el puntero de ese registro hasta localizar el siguiente registro según el orden de la clave de búsqueda, se seguiría procesando registros hasta encontrar uno cuya clave de búsqueda sea distinta.

1. ¿Cuál es la diferencia entre un índice primario y un índice secundario?

En los índices primarios los archivos están ordenados secuencialmente, estos archivos se llaman archivos secuenciales indexados. Hay dos tipos de índices ordenados, índice denso (crea un registro índice por cada valor de la clave de búsqueda en el archivo) y el índice disperso (solo se crea un registro índice para algunos de los valores). En cambio, en los índices secundarios deben ser densos, con una entrada en el índice por cada valor de la clave de búsqueda y un puntero a cada registro del archivo. En los índices secundarios si la clave de búsqueda no es la clave candidata, no seria suficiente apuntar solo al primer registro de cada valor de la clave, ya que los registros están ordenados según la clave de búsqueda del índice primario, por lo tanto, un índice secundario debe contener punteros a todos los registros. Los punteros de estos índices secundarios no apuntan directamente al archivo, cada puntero apunta a un cajón que contiene punteros al archivo. Los índices secundarios mejoran el rendimiento de las consultas que emplean claves que no son la de búsqueda del índice primario.

1. Dado un archivo de clientes ordenado por código de cliente que contiene 15000 registros de longitud fija de 64 Bytes, el archivo está almacenado en un dispositivo de almacenamiento magnético con bloques físicos de 512 Bytes.
   1. Calcular Factor de bloqueo: fbl =? registros por bloque.

r=15000. R=64 Bytes. B=512 Bytes

**fbl = (B/R)** = 8 registros por bloque

* 1. El número de bloques requerido para el archivo es: b = ? bloques.

**b = (r/fbl)** = 1875 bloques

* 1. Cantidad de accesos a bloques en una búsqueda binaria en el archivo de datos

**Log2b** = 11 accesos a bloques.

1. Dado un archivo de clientes con campo clave de ordenación código de cliente del archivo índice primario, y sabiendo que el archivo tiene 15000 registros de longitud fija de 64 Bytes, el archivo está almacenado en un dispositivo de almacenamiento magnético con bloques físicos de 512 Bytes, el campo clave de ordenación del archivo que tiene = 8 Bytes y el puntero a bloque de archivo tiene 4 Bytes.

Calcular:

* 1. El tamaño de cada entrada o registro del archivo índice primario.
  2. El factor de bloqueo del archivo índice.
  3. El número total de entradas o registros del archivo índice primario.
  4. El número de bloques requerido para el archivo índice.
  5. Cantidad de accesos en una búsqueda binaria en el archivo índice.
  6. Cantidad de accesos para encontrar un registro empleando el archivo índice.