Indices

El objetivo del índice es crear una estructura adicional que mantenga ordenados a los datos de una tabla a partir de una clave. Por ejemplo la creación de una PK o un índice unique

El índice puede estar en una tabla aparte o dentro de la tabla que ordena. A partir de la arquitectura 8086 esta en una tabla aparte.

Posicion relativa -> distancia al 0 en mi indice

Tipos de accesos a datos

* Secuencial -> el acceso se realiza en función al modo en que ingresaron los datos
* Secuencial indexado -> el acceso se realiza en función a una clave que fue definida
* Directo o random -> el acceso se realiza en forma directa a una clave que fue definida

En la practica existen 2 métodos para la creación de índices -> hashing y árbol binario (la primera es mas utilizada en arquitectura de mainframe y microcomputadoras mientras que la segunda se usa en arquitecturas 8086).

Hash

Las funciones hash crean una huella digital para un determinado dato a partir de como es este. Una buena función hash es aquella que:

* Evita colisiones -> para un conjunto de valores, los resultados de salida son todos distintos.
* Distribuye claves uniformemente -> distribuye uniformemente los valores de salida respecto a los de entrada.
* Fácil de calcular -> el tiempo de ejecución de la función hash debe ser de O(1)

El método crea una tabla (vector de dos dimensiones) donde en la primera dimensión colocará las claves y en la segunda dimensión las posiciones relativas de las mismas en la tabla donde se encuentran los datos correspondientes a esa clave. La función hash() recibe como entrada la clave a almacenar y devuelve un valor numérico entero que corresponde a la posición en la cual debería ir dicha clave en la tabla mencionada.

**Resolución de colisiones** -> si la función de hash me devuelve la misma clave ante 2 valores distintos estamos ante una colisión que hay que resolver. Los métodos que se usan son:

* Encaminamiento -> hago una lista con todos los valores que tienen la misma clave
* Direccionamiento abierto -> Busco otra posición dentro de la tabla. Los métodos para hacer esto son:
  + Sondeo lineal: busca secuencialmente en la tabla hasta encontrar una posición vacia
  + Sondeo cuadratico: empieza a buscar n posiciones por delante de la que dio originalmente
  + Hashing doble: aplica la función hash a la clave que dio y usa ese resultado como tamaño de salto para empezar a buscar.

Arboles M-arios

Se usan cuando se tiene un conjunto de datos muy grande y se trabaja con disco que también mueve bloques de datos muy grandes de entre 512 y 4096 bytes

El Árbol B es un tipo de árbol M-ario destinado a la creación de índices físicos para el acceso a la información. El objetivo principal es minimizar las operaciones de entrada y salida hacia el disco. Al imponer la condición de balance, el árbol es restringido de manera tal que se garantice que la búsqueda, la inserción y la eliminación sean todos de tiempo O(logn). El grado M del Árbol B varía entre 50 y 2000 y se determina en base al tamaño de las claves y del tamaño de la página del disco.

El árbol tiene 2 tipos de nodo nodo raíz o rama y nodo hoja

* Nodo hoja -> tiene una componente de dato donde van los valores de las claves ordenadas de menor a mayor y un componente puntero que contiene la posición relativa de los datos secuenciales correspondientes a esa clave
* Nodo raíz o rama -> tiene una componente de dato donde van los valores de las claves ordenados de menor a mayor y un componente puntero que apunta al nodo que contiene claves menores o iguales que ella.

Buscar en un árbol-B es muy parecido a buscar en un árbol binario de búsqueda, excepto que en vez de hacer una decisión binaria, o de dos caminos en cada nodo, hacemos una decisión multicamino en base al número de hijos del nodo.

Inserción: Para insertar un elemento x, comenzamos en la raíz y realizamos una búsqueda para él. Asumiendo que el elemento no está previamente en el árbol, la búsqueda sin éxito terminará en un nodo hoja. Este es el punto en el árbol donde el x va a ser insertado.

Split: Si ocurre que cuando se llega a la hoja no hay espacio para insertar el nodo se produce lo que se denomina split que es un proceso que divide el nodo en dos dejando la mitad de elementos en cada uno respetando el orden de menor a mayor, quedando la mitad de los elementos más chicos en un nodo y la mitad de los elementos mas grandes en el otro.

Eliminación: Para eliminar un elemento x, comenzamos en la raíz y realizamos una búsqueda para él. Asumiendo que el elemento existe, si existe se llegarà a la hoja donde esta y se borra, sino se dirà que no existe.

Fusión: Si ocurre que cuando se elimina el elemento x el nodo queda vacío, debe eliminarse el nodo, lo que puede generar una baja potencial en todos los antecesores de dicho nodo.

Factor de Split -> desperdicio espacio para mantener performance constante

Me indica a que porcentaje de llenado de un nodo lo tengo que splitear