Daniel54
www.wuolah.com/student/Daniel54

Tema 1 El Nivel Interno.pdf

Temas resumidos Nacho

3° Administración de Bases de Datos

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Tema 1. El nivel Interno.

1. Introducción.

Un **SGBD** es un sistema de software que almacena una cantidad de datos en diversos dispositivos. Dichos dispositivos estructuran los datos en ficheros en distintos sistemas de archivos.

Un **cluster** es un conjunto de bloques contiguos organizados verticalmente. Los archivos se almacenan en uno o varios clusters. Sin embargo, si el tamaño de archivo es menor que el tamaño del cluster, éste lo ocupa todo. El conjunto de cluster, que están a la misma distancia del centro de los discos, se denomina **cilindro**.

El sistema de archivos **FAT** tiene una entrada por cada fichero y una referencia a la primera entrada. Además, los bloques están enlazados entre sí.

La estructura en Linux que almacena la información sobre los ficheros se denomina **inodos.** Es una estructura que está distribuida, a diferencia de la estructura centralizada de Windows. Teniendo una estructura distribuida, es posible recuperar la información ya que todos los inodos están conectados entre sí (se puede recuperar información de los inodos que no estén fastidiados).

2. Evaluación de un Modo de Almacenamiento.

Para poder realizar la evaluación de un modo de almacenamiento, hay que tener en cuenta distintos parámetros.

Parámetro	Mide...
R	la memoria necesaria para almacenar un registro
T	el tiempo para encontrar un registro arbitrario
T_F	el tiempo para encontrar un registro por clave
T_W	el tiempo para escribir un registro cuando ya se tiene su posición
T_N	el tiempo para encontrar el siguiente registro a uno dado
T_I	el tiempo necesario para insertar un registro
T_U	el tiempo necesario para actualizar un registro
T_X	el tiempo necesario para leer el archivo
T_Y	el tiempo necesario para reorganizar el archivo

Estos parámetros nos permiten saber el tiempo de operaciones tales como recuperar un registro por valor de clave, obtener el siguiente registro, insertar un registro, etc.

Obtener el siguiente registro en un SA secuencial es una operación muy rápida ya que tienes un puntero apuntando al último registro introducido.


CUNEF
Lidera tu futuro.
Define tu éxito.
POSTGRADO EN
DATA SCIENCE
PARA FINANZAS
SÚMATE AL ÉXITO
Excelencia, futuro, éxito.
www.cunef.edu

3. REGISTROS.

UN SGBD almacena la información en tablas. Una **tabla** es una estructura bidimensional que tiene anchura y altura. En el modelo E/R, horizontalmente se le denomina esquemas y verticalmente se le denomina instancias.

La estructura de una tabla la determinan las columnas y la información de una tabla se almacena en filas. Cada fila es un **registro**.

Un SGBD provee de una serie de tipos de datos para las columnas de una tabla:

- **Numéricos:** enteros y reales.
- **Cadenas de caracteres:** CHAR, VARCHAR, VARCHAR2 (es un tipo de longitud variable, no reserva espacio).
- **Fecha:** DATE, TIME, TIMESTAMP.
- **Otros:** BLOB (Atributos capaces de contener hasta 2TB en un mismo registro (solo enteros)) y CLOB (BLOB pero con caracteres).

Tipo	Tamaño
CHAR(x)	*x* Bytes
VARCHAR2(X)	de 1 a *x*+1 Bytes
FLOAT	6 Bytes
BINARY_INTEGER	2 Bytes
...	...

Un **campo** es un atributo a nivel lógico, algo capaz de almacenar un valor. Un **registro** es un conjunto de campos. Al conjunto de registros, ordenados o no, se le denomina **bloque.** Y al conjunto de bloques, ordenados o no, se le denomina **fichero.**

Existen dos tipos de registros:

- **Registro de longitud fija:** Están compuestos por campos de longitud no variable. Se sabe el espacio que ocupa en todo momento.
- **Registros de longitud variable:** Cuya longitud depende de la longitud del dato que almacenan.

Una **estructura** es una secuencia de registros.

Existen dos tipos de estructuras:

- **Estructura homogénea:** Son bloques de estructura homogénea, es decir, todos los registros son de la misma estructura. Es decir, tienen los mismos campos.
- **Estructura heterogénea:** Son bloques de estructura heterogénea, es decir, almacenan registros de distinta estructura. Los registros tienen distinta cantidad de campos.

2.1. LONGITUD DE UN REGISTRO.

El tamaño de un **registro de longitud fija**, se determina calculando el tamaño total de todos sus campos.

Si el registro es de **longitud variable**, el tamaño de un registro no se puede calcular. Se puede **estimar**.

- A: longitud media de los nombres de atributo
- V: longitud media de los valores de atributo
- a': número medio de atributos
- s: número de separadores por atributo

$$R = a'(A+V+s)$$

El tamaño medio del registro variable depende de la longitud media de los nombres de atributo. Como aquí no hay un tamaño fijo, necesitamos **delimitadores** para poder separar un campo del siguiente.

Por cada uno de los campos, hay dos **separadores obligatorios** (el de identificador con el valor y el de entre campo y campo).

4. BLOQUES.

Un **bloque** es una unidad de información transferida por un dispositivo de almacenamiento masivo y almacenada en el área de trabajo de la memoria denominada **buffer**.

El tamaño de bloque siempre es **fijo** para la base de datos. Una vez establecido, no se puede tocar nunca más. El bloque de base de datos es siempre **múltiplo** del bloque físico del SO. A su vez, el bloque del SO es múltiplo del bloque de disco.

4.1. BLOQUEO.

El **bloqueo** es la forma en la que se almacenan registros dentro de un bloque. Un registro en disco tiene que pasar a un bloque de SO y, posteriormente, a un bloque de DB para ser tratado.

El **factor de bloqueo** es el número de registros que caben en un bloque que depende del tamaño del mismo bloque y del tamaño de los registros. Además, cada uno de los bloques necesita información del propio bloque.

Hay dos métodos de bloqueo:

- **Bloqueo fijo:** Se rellena el bloque con tantos registros como sea posible.
  - **Bloqueo fijo con registros de longitud fija:** (Se redondea hacia abajo el resultado)

$$Bfr = \left\lfloor \frac{B-C}{R} \right\rfloor$$

- **Bloqueo fijo con registros de longitud variable:** El siguiente registro cabe en el bloque si hay espacio. Las marcas de separación de registros ocupan espacio (caracteres especiales, distancia al comienzo del siguiente registro, tabla de posiciones de los campos (con inicial y final).

$$Bfr = \left\lfloor \frac{B-C}{R+M} \right\rfloor$$

- **Bloqueo partido:** Los registros que no quepan en el bloque, se meten en el siguiente bloque. Todos los bloques están enlazados entre sí. Los **problemas** del bloqueo partido son las **búsquedas secuenciales** (hay que leer dos bloques distintos) y la **actualización de ficheros** (los registros que estén separados, hay que unirlos para actualizarlos y volver a separarlos). Las **ventajas** es que no desperdicia espacio en los bloques y es la **única solución** cuando el tamaño del registro es mayor que el de un bloque.
  - **Bloqueo partido con registros de longitud variable:** Si se considera que todos los bloques tienen un registro partido al final, siempre va a haber una referencia al siguiente bloqueo. Entonces, además de la cabecera, también vamos a tener que almacenar un puntero al siguiente registro (P).

$$Bfr = \left\lfloor \frac{B-P-C}{R+M} \right\rfloor$$

5. ESPACIO DESPERDICIADO.

El espacio desperdiciado es aquel que se pierde en marcas, referencias, etc. Básicamente, es espacio en el que no cabe un registro.

En definitiva, el bloqueo fijo es más eficiente para registros pequeños ya que la tasa de utilización es muy alta. Y el bloqueo partido es más eficiente para registros grandes.

FALTA FOTO.

6. ORGANIZACIÓN DE ARCHIVOS Y MÉTODOS DE ACCESO.

Principalmente existen cuatro tipos de organización de archivos:

- **Archivo Secuencial Físico** (ASF). Los registros están almacenados de forma secuencial.
- **Archivo Secuencial Lógico** (ASL). Hay una parte en la que los registros están ordenados y otra parte en la que no (zona de desbordamiento).
- **Archivo Secuencial Indexado** (ASI).
- **Archivo de Acceso Directo** (AAD). A partir de un valor de clave, se obtiene una posición relativa (hash).

El resto de organización de archivos, son combinaciones de estas organizaciones.

Los ficheros almacenan registros y no se pueden escribir registros de forma individual.

6.1. Archivo Secuencial Físico.

Partimos de una hipótesis:

- Registros de estructura variable y longitud variable.
- Los campos estarán compuestos del identificador de atributo y el valor.
- Dos separadores: entre el id y el valor y entre campos.

6.1.1. Tamaño de un registro.

$$R = a' \cdot (A + V + 2)$$

6.1.2. Recuperación de un Registro.

$$T_F = \sum_{i=1}^{n} \frac{i}{n} \cdot T = \frac{n+1}{2} \cdot T \approx \frac{n}{2} \cdot T$$

6.1.3. Siguiente Registro (por Clave).

$$T_N = T_F$$

6.1.4. Inserción de un Registro.

Los registros se van almacenando por orden de llegada y no se puede alterar su posición. Es el mismo tiempo que se tarda en encontrar un registro por un valor específico en un archivo secuencial físico.

$$T_I = T_W$$

Hay que intentar no hacer búsquedas secuenciales.

El tiempo que se tarda en insertar es el mismo que se tarda en escribir.

6.1.5. Actualización de un Registro.

6.1.5.1. TAMAÑO DE REGISTRO NO CAMBIA.

Si el tamaño de registro no cambia, el tiempo que se tarda es la suma del tiempo que se tarda en encontrarlo y en sobreescribirlo.

$$T_A = T_F + T_W$$

6.1.5.2. TAMAÑO DE REGISTRO CAMBIA.

Si el tamaño de registro cambia, el tiempo que se tarda es el tiempo que se tarda en encontrarlo sumándole el tiempo de sobreescribirlo (marcarlo de alguna manera que está borrado) y sumándole el tiempo que se tarda en insertar la nueva versión.

$$T_A = T_F + T_W + T_I$$

6.1.6. Lectura de un Fichero.

6.1.6.1. INDEPENDIENTEMENTE DEL CONTENIDO DE REGISTRO.

Si no depende del contenido del registro, el tiempo es constante. Hay que leer los 'n' registros con un tiempo de lectura T.

$$T_X = n \cdot T$$

6.1.6.2. LECTURA ORDENADA SEGÚN EL VALOR DE UN ATRIBUTO.

Si se busca por valor de atributo, lo que se tarda es el tiempo que se tarda en encontrar cada uno de los registros y lo multiplico por los 'n' registros.

$$T_X = n \cdot T_F$$

6.1.7. Reorganización de Fichero.

El **número de registros añadidos** (O) es el número de registros añadidos desde su creación.

d es el número de registros marcados para borrar. T es el tiempo que tarda en leer un registro y Tw es el tiempo que tarda en escribir un registro.

Para poder reorganizar un fichero secuencial hay que crear otro. El tiempo de reorganización de fichero (Ty) es el tiempo que se tarda en leer todos los registros, comprobar los que no estén marcados como borrados y escribir los que no estén marcados en el nuevo fichero.

$$T_Y = (n+O) \cdot \dot{T} + (n+O-d) \cdot T_W$$

6.2. ARCHIVO SECUENCIAL LÓGICO (ASL).

En este tipo de organización, los registros están ordenados según una clave física. El ASL solo se utiliza en registros de longitud fija y por esta razón pierde mucho espacio.

Cuando se insertan nuevos registros, como los tenemos ordenados, es costoso abrir huecos. Por eso existe una zona que no está ordenada llamada **zona de desbordamiento** (de tipo ASF). Cuando la zona de desbordamiento sea superior a la zona ordenada, el ASL no sirve y tenemos que reconstruir el fichero entero realizando una mezcla entre la parte ordenada y la parte no ordenada obteniendo el archivo ordenado.

6.2.1. ESPACIO PARA UN REGISTRO.

$$R = \sum_i V_i$$

6.2.2. RECUPERACIÓN DE UN REGISTRO.

Si el valor de búsqueda no es clave, el tiempo de recuperación de un registro es secuencial.

$$T_F = \frac{n}{2} \cdot T$$

Si, además, hay registros en la zona de desbordamiento tenemos que recorrer secuencialmente la zona ordenada y la zona de desbordamiento.

$$T_F = \frac{n}{2} \cdot T + \frac{O}{2} \cdot T$$

Si el valor de búsqueda es clave, el tiempo que se tarda es el tiempo que tarda el algoritmo de búsqueda binaria.

$$T_F \approx \log_2(n) \cdot T$$

6.2.3. SIGUIENTE REGISTRO.

6.2.3.1. SIGUIENTE VALOR ESTÁ EN EL PROPIO FICHERO.

$$T_N = T$$

6.2.3.2. SIGUIENTE VALOR ESTÁ EN EL FICHERO DE DESBORDAMIENTO.

El siguiente registro no es el que se busca y hay que recorrer la zona de desbordamiento secuencialmente.

$$T_N = T + O \cdot T$$

6.2.4. INSERCIÓN DE REGISTRO.

El tiempo que se tarda en insertar un registro nuevo es lo que se tarda en localizar el punto de inserción (determinar si la zona ordenada está rellena. Si lo está, vamos al final de la zona de desbordamiento), y escribir el registro.

$$T_I = T_F + \frac{n}{2} \cdot T + \frac{n}{2} \cdot T_W [+ T_W]$$

6.2.4.1. INSERCIÓN DE VARIOS FICHEROS Y ZONA DE DESBORDAMIENTO.

Si hay varios registros para insertar y hay fichero de desbordamiento, podemos insertar todos los nuevos registros en él.

$$T_I = T_W$$

Pero después hay que insertarlos en el fichero maestro.

$$T_I = \frac{T_Y}{O}$$

6.2.5. ACTUALIZACIÓN DE REGISTRO.

6.2.5.1. SI NO SE CAMBIA EL VALOR DE LA CLAVE.

$$T_U = T_F + T_W$$

6.2.5.2. SI SE CAMBIA EL VALOR DE LA CLAVE.

Si se cambia el valor de la clave, hay que marcar el registro que queremos actualizar como borrado e introducir el nuevo.

$$T_U = T_F + T_W + T_I$$

6.2.6. LECTURA DE FICHERO.

Si solo leemos el fichero maestro hay que recorrer todos de forma secuencial.

$$T_X = n \cdot T$$

Si, además, se usa la zona de desbordamiento tenemos que ordenar la zona ordenada y la zona desordenada y leer una vez hecho la mezcla.

$$T_X = T_C(O) + (n+O) \cdot T$$

6.2.7. REORGANIZACIÓN DEL FICHERO.

Esta operación se realiza si hay desbordamiento. Hay que re-ordenar el fichero de desbordamiento y realizar la mezcla entre ambos ficheros (maestro y desbordamiento) en un nuevo fichero.

$$T_Y = T_C(O) + (n+O) \cdot T + (n+O-d) \cdot T_W$$

6.3. ARCHIVO SECUENCIAL INDEXADO (ASI)

En este tipo de organización todos los registros están desordenados pero hay una estructura adicional que se almacena en un archivo separado.

Tiene una estructura:

- Un fichero de datos: ASL con una posible área de desbordamiento.
- Un fichero de índice: ASL con registros de longitud fija que contiene valor de clave y dirección del fichero de datos.

Este método acelera el acceso por clave.

El fichero de índice es un ASL. Puede permitir que haya una zona de desbordamiento para no tener que estar ordenando constantemente. Mientras esté en memoria, el índice puede estar desordenado. Una vez deje de estar en memoria, se hace la mezcla entre la parte ordenada y desordenada de los índices.

Si la posición en memoria es de **registro**, se habla de **índice denso.** En este tipo, el número de registros del fichero de índice coincide con el número de registros del fichero maestro. Pueden haber otros índices sobre el mismo fichero maestro, llamados **secundarios**.

Si la posición en memoria es de **bloque**, se habla de **índice no denso.**

6.3.1. ESPACIO DE UN REGISTRO. ASI DENSO.

A la hora de calcular un tamaño de registro en un archivo denso es la suma del tamaño de los valores más el valor de la clave más la posición de dónde está el índice en memoria.

$$R=\sum_i V_i+(V_k+P)$$

6.3.2. RECUPERACIÓN DE REGISTRO. ASI DENSO.

En un archivo secuencial indexado con índice denso se tarda logaritmo (n) en encontrar el registro que queremos (al estar ordenado). Una vez tengamos el índice que queremos, hay que leerlo para ver si es que el realmente queremos. Y una vez lo tengamos leído, tenemos que volver a leer el registro.

$$T_F=\log_2(n)\cdot T+T$$

6.3.3. RECUPERACIÓN DEL SIGUIENTE REGISTRO. ASI DENSO.

Como el índice está ordenado, solo hay que leer el siguiente índice y el propio registro.

$$T_N=T+T=2\cdot T$$

6.3.4. INSERCIÓN DE REGISTRO. ASI DENSO.

TI1: Insertar registro de datos en el fichero maestro.

TI2: Insertar registro de índice en fichero de índice.

6.3.4.1. NO HAY ZONA DE DESBORDAMIENTO. ASI DENSO.

Si el fichero maestro no tiene desbordamiento, escribimos el registro en la zona de datos. Nos quedamos con la posición en la que lo hemos insertado e insertamos en el fichero índice la clave y la posición.

$$T_{I1}=T_{I2}=T_F+\frac{n}{2}\cdot T+\frac{n}{2}\cdot T_W[+T_W]$$

$$TI=T_{I1}+T_{I2}=2\cdot\left(T_F+\frac{n}{2}\cdot T+\frac{n}{2}\cdot T_W[+T_W]\right)$$

6.3.4.2. HAY ZONA DE DESBORDAMIENTO. ASI DENSO.

Si el fichero maestro tiene desbordamiento, escribimos el registro la zona de desbordamiento. Nos quedamos con la posición en la que lo hemos insertado e insertamos en el fichero índice la clave y la posición.

$$T_{I2}=T_F+\frac{n}{2}\cdot T+\frac{n}{2}\cdot T_W[+T_W]$$

$$TI=T_W+T_{I2}$$

6.3.5. ACTUALIZACIÓN DE REGISTRO. ASI DENSO.

6.3.5.1. SI NO SE CAMBIA EL VALOR DE LA CLAVE. ASI DENSO.

$$T_U=T_F+T_W$$

6.3.5.2. SI SE CAMBIA EL VALOR DE LA CLAVE. ASI DENSO.

$$T_U = 2 \cdot (T_F + T_W) + T_{I1} + T_{I2}$$

6.3.6. LECTURA DE UN FICHERO. ASI DENSO.

6.3.6.1. POR ÍNDICE PRINCIPAL. ASI DENSO.

Por índice principal, la lectura del fichero se realiza de forma secuencial.

$$T_X = n \cdot T$$

6.3.6.2. POR ÍNDICE SECUNDARIO. ASI DENSO.

Hay que leer cada elemento en el índice y luego leer el registro.

$$T_X = n \cdot T + n \cdot T = 2 \cdot n \cdot T$$

6.3.6.3. POR ÍNDICE SECUNDARIO Y DESBORDAMIENTO. ASI DENSO.

Si el elemento que estamos buscando está en la zona de desbordamiento, hay que buscarlo.

$$T_X = (n+O) \cdot T + n \cdot T + O \cdot T = 2 \cdot (n+O) \cdot T$$

6.3.7. REORGANIZACIÓN DEL FICHERO. ASI DENSO.

Hay que reorganizar el fichero de desbordamiento, realizar la mezcla con el fichero maestro y el fichero de desbordamiento y crear el índice de nuevo con todos los registros ordenados.

$$T_Y = T_C(O) + (n+O) \cdot T + (n+O-d) \cdot T_W + (n+O-d) \cdot T_W$$

Ocurre cuando la zona de desbordamiento es muy mayor (un 20%).

Se ordena la parte desordenada. Leemos todos los registros (partes ordenada y parte desordenada). Escribimos todos los registros menos los eliminados de la parte ordenada y desordenada. Y escribir en el índice.

6.3.8. ARCHIVO SECUENCIAL INDEXADO NO DENSO.

El problema de los índices densos es que son demasiado grandes para memoria. Por eso surgen los **índices no densos**. En vez de apuntar a un registro individual, apunta a una página de registros.

El fichero índice contiene tantas entradas como páginas contiene el fichero de datos.

Una clave dada está en una página (es mayor o igual que la clave de esa página y menor que la de la siguiente).

6.3.8.1. Entradas de Índice.

$$\frac{n}{Bfr}$$

6.3.8.2. DIFERENCIAS CON EL ÍNDICE DENSO.
El procedimiento de búsqueda en un índice no denso:

1. Se seleccionar en el índice la clave inmediatamente inferior a la buscada.
2. Se carga el bloque del maestro.
3. Se busca secuencialmente.

Si no está, hay que acceder al maestro. En un denso no es necesario.

6.3.8.3. CARACTERÍSTICAS.
En cada página del maestro se suele dejar un espacio para nuevos registros.

En maestros con fichero de desbordamiento, cada registro tiene un enlace al siguiente en el fichero de desbordamiento. En una página, hay un enlace a registro de desbordamiento (si la página está completa). En el fichero de desbordamiento, un registro puede tener enlace al siguiente.

6.3.9. ARCHIVO SECUENCIAL INDEXADO MULTINIVEL.
Este tipo de organización de archivo es efectiva cuando los archivos son grandes y/o los registros son grandes. Si el índice es grande, puede indexarse. A su vez, si el índice del índice es grande puede indexarse. A esto se le llama **índice multinivel**.

Si es de m niveles, todos los niveles son no densos. La estructura es de árbol y al primer nivel se le llama **raíz**. El nivel 1 es el más cercano al fichero maestro.

6.3.9.1. TAMAÑO DE PÁGINA DEL ÍNDICE.
El factor de bloqueo es el número de registros que caben en un bloque. B es el tamaño del bloque. C es la cabecera.

El bloque menos la cabecera es lo que llamamos **tamaño útil**.

Se redondea para abajo.

6.3.9.2. ESPACIO DE REGISTRO.
Mantenerlo ordenado es muy complicado. Siempre y cuando lo vayamos metiendo y estén ordenados, bien. Una vez que empiecen a estar desordenado, a tomar por saco.

En general, el fichero maestro tiene desbordamiento, entonces:

$$r = P + \sum_i V_i$$

El índice de primer nivel tiene una entrada por bloque maestro, luego el número de entradas es:

$$i_1 = \left\lceil \frac{n}{Bfr} \right\rceil$$

Los niveles superiores tienen su propio tamaño de bloque y su propio número de registros, con lo que el número de entradas será:

$$i_k = \left\lceil \frac{i_{k-1}}{y} \right\rceil$$

Y el número de bloques:

$$b_k = \left\lceil \frac{i_k}{y} \right\rceil = i_{k+1}$$

El espacio necesario para índices es de:

$$I = (b_1 + b_2 + \ldots + b_m) \cdot B$$

Y el espacio medio necesario por registro:

$$R = r + \frac{O}{n} \cdot r + \frac{I}{n}$$

6.3.9.3. RECUPERACIÓN DE REGISTRO.

$$T_F = T_M + (m-1) \cdot T_{F_i} + T'_F$$

$$T_F = T_M + (m-1) \cdot T_{F_i} + T'_F + T_{F\,Cadena}$$

6.3.9.4. SIGUIENTE REGISTRO.

$$T_N \approx T$$

6.3.9.5. INSERTAR REGISTRO.

Existen varios casos:

- Se tiene que insertar en el fichero maestro y queda espacio en el bloque. El tiempo que se tarda en insertar en un fichero específico.

$$T_I \approx T_F + \frac{1}{2} \cdot Bfr \cdot (T + T_W) + T_W$$

- Se tiene que insertar en el fichero maestro y no queda espacio en el bloque.

$$T_I \approx T_F + \frac{1}{2} \cdot Bfr \cdot (T + T_W) + 2 \cdot T_W$$

- Se tiene que insertar en el fichero de desbordamiento.

$$T_I \approx T_F + 2 \cdot T_W$$

6.3.9.6. ACTUALIZACIÓN DE REGISTRO.

Existen varios casos:

- Si no cambia el valor de la clave. Lo busca en todos los niveles del índice y cambiamos el valor.

$$T_U = T_F + T_W$$

- Si cambia el valor de la clave. Lo borramos y lo volvemos a meter.

$$T_U = T_F + T_W + T_I$$

6.3.9.7. LECTURA EXHAUSTIVA DE FICHERO.

$$T_X \approx (n+O)\cdot T$$

6.3.9.8. REORGANIZACIÓN DE FICHERO.

$$T_Y = T_X + (n+O-d)\cdot T_W + k\cdot I$$

Hay que intentar reorganizar absolutamente todo. Tiempo que se tarda en leer todo, en escribir y lo que tardamos en construir los distintos índices que teníamos.

6.3.9.9. CRITERIOS PARA SELECCIÓN DE ÍNDICES.

- Sólo son útiles si hay consultas que los utilicen.
- Los atributos candidatos son los involucrados en la cláusula WHERE (cuando se ven afectados por la igualdad o el uso de rangos) o aquellos atributos de reunión.
- Si el número de tuplas con valores repetidos para un campo supera el 5%, no debería crearse un índice sobre ese campo.
- Los campos de un índice multiatributo deben ordenarse según las consultas más frecuentes.
- No se deben indexar campos actualizados frecuentemente.

6.4. Archivo de Acceso Directo (AAD).

Este tipo de organización de archivos:

- Aprovecha el acceso directo de disco.
- Mediante la clave se obtiene la posición.
- Prevé nuevas entradas dejando espacio libre.
- Hay varios métodos de transformación.

Suponemos un campo clave de tamaño V (si es número, hay 10^V valores posibles; si es alfabético, hay 26^V valores posibles).

El fichero tendrá **m** celdas para **n** datos (incluyendo espacios libres) luego m >= n.

Si una clave tiene **b** símbolos distintos, entonces hay **b^V** posibles claves, que cumplirá **b^V >> m** y el algoritmo transforma cada clave en un valor entre 0 y m-1.

Puesto que b^V >> m:

- Se dan **colisiones**. Dos claves con la misma posición. Soluciones:
  - **Desbordamiento.**
  - **Direcciones de página.**
- Hay **huecos.** Ninguna clave da una posición. Solución:
  - Usar estos huecos para resolver colisiones.

Hay tres métodos de resolución de colisiones:

- **Direccionamiento cerrado.** El espacio de posiciones en un único archivo.

    - **Búsqueda lineal.** Una colisión se almacena en una posición libre del mismo bloque (secuencial en búsqueda de registro).
    - **Realeatorización.** Una colisión se almacena en otra posición mediante otra transformación de clave, y sucesivamente.
- **Direccionamiento abierto.** El espacio de posiciones en más de un fichero (principal y desbordamiento).
    - **Listas enlazadas.** Colisiones en fichero de desbordamiento (estructura ASI).
    - **Bloques de desbordamiento.** Colisiones en bloques del fichero de desbordamiento.
- **Hashing dinámico.** El espacio de posiciones y la transformación se adaptan dinámicamente.

6.4.1. ESPACIO DE UN REGISTRO.

$$r = P + \sum_i V_i$$

$$S_F = m \cdot r + c \cdot r = (m+c) \cdot r$$

$$R = \frac{S_F}{n} = \frac{(m+c)}{n} \cdot r$$

Hay que tener en cuenta los registros que están dentro del fichero ordenado por hash multiplicado por el número de cada uno de los registros más el tamaño de los que están en la zona de desbordamiento por el número de registro.

Se divide el tamaño entero entre n.

6.4.2. RECUPERACIÓN DE REGISTRO.

$$T_F = C + T + p \cdot T_{F\_Cadena}$$

C: El tiempo que se tarda en aplicar la función Hash y que te devuelva un registro.

TF_Cadena: El tiempo que se tarda en buscar una cadena.

P: Probabilidad de colisión.

6.4.3. INSERCIÓN DE UN REGISTRO.

$$T_I = C + 2 \cdot T_W$$

El tiempo que se tarda en calcular su posición y el tiempo que se tarda en escribir dos veces.

6.4.4. ACTUALIZACIÓN DE UN REGISTRO.

$$T_U = T_F + T_W$$

6.4.5. LECTURA EXHAUSTIVA DE FICHERO.

$$T_X \simeq (m+c) \cdot T$$

6.4.6. REORGANIZACIÓN DE FICHERO.

$$T_Y = T_X + T_{Carga}$$

$$T_{Carga} = \sum_{i=1}^{n} T_I(i)$$

Consiste en aplicarle una segunda función hash. Consiste leer todo el fichero e introducirlo en una nueva estructura.

6.5. Evaluación del Sistema.

Tenemos distintos parámetros para estimar la evaluación de un sistema.

6.5.1. ESTIMACIÓN DE USO DEL SISTEMA.

La carga por demandas de almacenamiento:

- **Se ve afectada por** el número de registros, número total de atributos, número medio de atributos por registro, longitud de campos, longitud de identificadores de atributos.
- **Carga por recuperación de información.** Se calcula teniendo en cuenta el número de solicitudes a un archivo en un conjunto de transacciones en un tiempo dado.
- **Carga por actualización.** Se calcula teniendo en cuenta el número de actualizaciones a la base de datos en base a la frecuencia de creación de registro, de actualización de registro, de eliminación y de ampliación.

6.5.2. ANÁLISIS DE BENEFICIOS.

La estimación de beneficios se basa en la probabilidad de las operaciones:

- Se tienen en cuenta los gastos del personal.
- Tiempo de respuesta bajos centran al personal. Se calcula en base a :
    - Tiempo de solicitud.
    - Tiempo de computación.
    - Tiempo de presentación de resultados.

En términos de uso del sistema:

- En general:
    - Tiempo medio para plantear una consulta.
    - Tiempo medio para describir una salida de datos.
    - Tiempo medio de procesamiento de operación.
    - Retraso medio en presentación de resultados.
- Hay que tener en cuenta el tipo de usuarios, porque incluye un costo adicional en el manejo.

En el mantenimiento de los datos, el mayor coste está asociado a la actualización. Esto conlleva un coste de detección y corrección de errores:

- **Antes de la actualización:** difícil y costoso.
- **Durante la actualización:** puede llevar a acciones no deseadas e incrementos de costes en el sistema.
- **Después de la actualización:** hay que rastrear el alcance y es mejor no cambiarlo.

6.5.3. NECESIDADES DE ALMACENAMIENTO.

Ante cambios frecuentes, será necesario prever más espacio reservado. La densidad mide la proporción de espacio ocupado (y depende del tipo de archivo).

$$D=\sum_{F_i} n_i \cdot \frac{R+W}{\mu_{D_i}}$$

Se utilizan componentes como discos, canales, controladores, etc. En general, mejor una sola unidad mayor que varias más pequeñas. La distribución utiliza varios procesadores y un dispositivo de almacenamiento.

6.5.4. COSTES DE EXPLOTACIÓN.

Para parámetros conocidos de carga y efectividad de archivos, podemos escoger entre varios enfoques:

- **Primer enfoque.** Modelar cada diseño físico y seleccionar.
  - **Es costoso.**
- **Segundo enfoque.** Reducir posibilidades. Primero, el hardware adecuado. Después el diseño físico para el uso estimado dados los recursos existentes.
  - Estimar la demanda.
  - Estimar recursos.
  - Comparar para encontrar una combinación satisfactoria.
  - Si no acoplan, ambas estimaciones se ajustan de forma más precisa.
  - Si la discordancia es importante, cambiar el equipo o el diseño físico y empezar de nuevo.
- **Procesos compartidos.** Existe competencia:
  - **A nivel de procesador.** Afecta al tiempo de respuesta.
  - **A nivel de disco.** Provoca recolocación constante del cabezal, lo que afecta al registro siguiente y a la lectura exhaustiva.
  - La programación concurrente permite bloquear un dispositivo hasta que una tarea concluya o ésta lo libere.

6.5.5. ANÁLISIS COSTE/BENEFICIO.

Beneficios:

- Cantidad de datos disponibles.
- Intensidad de uso de la BD.

Costes:

- Almacenamiento (inicial + ampliaciones).
- Procesamiento (escalabilidad de discos, controladoras o canales).