Índice general

Capítulo 1

Nivel Interno

Recordemos que un sistema gestor de bases de datos (SGBD) tiene tres niveles distintos: interno, conceptual y lógico. En este tema nos vamos a centrar en el primero de ellos. El nivel interno está formado, a su vez, por otros dos niveles distintos:

(a) Comunicación SO: asociado al sistema operativo donde se ejecute el SGBD.

(b) Gestión de la información: asociado el SGBD en sí.

1.1. Medidas para evaluar un sistema de archivos

Al final, la información está almacenada en un disco duro, que tienen sistemas de ficheros asociados que les dan estructura. Hay que compararlos para elegir el más eficiente, para ello, tenemos una serie de 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 actuar un registro
T_X	el tiempo necesario para leer el archivo
T_Y	el tiempo necesario para reorganizar el archivo

1.2. Registros y bloques

Un SGBD almacena la información en tablas. La estructura de las tablas vienen determinadas por las columnas. La información de una tabla se almacena en filas. EL SGBD provee de una serie de tipos de datos para las columnas de una tabla (numéricos, enteros, etc).

Los ficheros en los que se almacenan las tablas de una base de datos se componen de un conjunto de bloques, que a su vez se componen de un conjunto de registros, que a su vez están compuestos por campos, y en cada campo se almacena un valor.

Los registros, según su longitud (R), pueden clasificarse en:

(a) Longitud fija: todos los campos tienen longitud fija conocida a priori. Si denotamos por V_i a la longitud del campo $i-$ésimo, entonces la longitud del registro es:

$$R = \sum_i V_i$$

(b) Longitud variable: hay por lo menos un campo del registro que tiene longitud variable. Con la siguiente notación

   a) A: longitud media de los nombres de atributo.

   b) V: longitud media de los valores de atributo.

   c) a': número medio de atributos.

   d) s: número de separadores por atributo.

   tenemos que la longitud del registro es:

$$R = a'(A + V + s) \tag{1.1}$$

Ejemplo 1.1. Calcular la longitud del siguiente registro de longitud variable:

Factura=325;	Linea=1;	Concepto=Análisis;	Cant=1;	Precio=300;

$$\left.\begin{array}{c} A = (7+5+8+4+6)/5 = 6 \\ V = (2+2+8+2+6)/5 = 4 \\ s = 2 \end{array}\right\} \Rightarrow R = a'(6+4+2) = 12a'$$

Los **bloques** son las unidades de transferencia de información del disco a la memoria o viceversa (aunque en memoria se denominan *buffers*). El tamaño del bloque es fijo para toda la base de datos y es múltiplo del tamaño del bloque físico del SO (por eficiencia).

Según su estructura, los bloques se pueden clasificar en:

(a) Estructura homogénea: todos los registros tienen la misma estructura, es decir, todos los registros son del mismo tipo y tienen el mismo número de campos.

(b) Estructura heterogénea: los registros tienen distinta estructura (los registros tienen distinta longitud). Para usar este tipo, también necesitamos almacenar información sobre la propia estructura del registro.

La relación entre los registros en disco, los bloques del SO y los bloques del SGBD es:

$$\text{Registro disco} \leftrightarrows \text{Bloque SO} \leftrightarrows \text{Bloque SGBD}$$

LLamamos **factor de bloqueo** (Bfr), al número de registros que caben en un bloque y depende del tamaño del mismo bloque, B, y del tamaño de los registros, R. Este valor puede ser fijado a priori por el administrador del SGBD. Siempre incluye una cabecera **C**, con información útil para el sistema (referencias, fecha de actualización, etc), que tiene que restarse a B.

Denominamos **bloqueo** a la forma en la que se ajustan los registros a un bloque. Hay dos tipos de bloqueo:

(a) Fijo o entero: se rellena el bloque con tantos registros como sea posible.

a) Registro longitud fija:

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

b) Registro de longitud variable:

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

M es el tamaño de las marcas de separación, ya que al ser de longitud variable, las necesitamos para diferenciar entre los registros.

En ambos se redondea hacia abajo ya que si no cabe un registro entero, no podemos guardar una fracción de él.

(b) Partido o desencadenado: se escriben registros en un bloque hasta que no quede espacio. Cuando vayamos a insertar el último registro, puede caber entero o partirse en dos partes, cada una en un bloque distinto. Debido a eso, es necesaria la existencia de una referencia del bloque conteniendo el primer trozo, al bloque conteniendo el otro.

a) Registro longitud fija:

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

b) Registro de longitud variable:

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

P es el tamaño del puntero al siguiente bloque.

Llamamos **espacio desperdiciado (W)** al espacio que se pierde en marcas, referencias, espacio en el que no cabe un registro, etc.

(a) Bloqueo partido con registros de longitud variable:

$$W = \frac{(P + Bfr \cdot M)}{Bfr} = \frac{P}{Bfr} + M$$

Esa fórmula se puede ver como el porcentaje de espacio del bloque que no es dedicado a almacenar registros. Como es bloque partido, necesitamos un puntero P al siguiente bloque, y como son registros de longitud variable, necesitamos un separador por *cada* registro que almacenamos en ese bloque, es decir, usamos $Bfr \cdot M$ espacio para separadores.

Una vez sabemos medir cuánto espacio se desperdicia en un bloque, podemos saber más o menos que bloqueo es más eficiente teniendo en cuenta el tamaño de los registros:

(a) El bloqueo fijo es más eficiente para registros pequeños.

(b) El bloqueo partido es más eficiente para registros grandes.

1.3. Organización de archivos y métodos de acceso

1.3.1. Archivo Secuencial Físico (ASF)

En este tipo de archivo suponemos que los registros tienen estructura y longitud variable, es decir, necesitamos *dos* separadores: el primero para separar los campos dentro de un registro y el otro para separar el nombre el campo de su valor. Por ejemplo:

Factura=325;	Linea=1;	Concepto=Análisis;	Cant=1;	Precio=300;

Tamaño de registro

Si recordamos la expresión (1.1), tenemos que: (porque $s = 2$):

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

Recuperación de un registro

$$T_F = \sum_{i=1}^{n} \frac{i}{n} T = \frac{n+1}{2} T \approx \frac{n}{2} T \tag{1.2}$$

Siguiente registro (por clave)

$$T_N = T_F$$

Como es por clave, el siguiente registro no tiene porqué estar justo después del que tenemos, por lo que hay buscarlo entero de nuevo.

Inserción de registro

$$T_I = T_W$$

Simplemente hay que abrir el archivo, ir al final e insertarlo.

Actualización de registro

Hay que distinguir dos casos:

(a) Si el tamaño del registro no cambia, lo único que hay que hacer es encontrarlo y sustituir el nuevo valor:

$$T_U = T_F + T_W$$

(b) Si el tamaño del registro cambia, entonces primero tenemos que encontrar el registro (T_F), marcarlo como inválido (T_W) e insertarlo al final (T_I):

$$T_U = T_F + T_W + T_I$$

Lectura de fichero

Hay que distinguir dos casos en función de si queremos que el resultado esté ordenado:

(a) Independientemente del contenido del registro: como están seguidos, simplemente hay que leerlos todos:

$$T_X = nT$$

(b) Lectura ordenada según el valor de un atributo: hay leer el primero, buscar el siguiente, leerlo, buscar el siguiente, es decir: (ojo, al buscarlo ya lo has leído):

$$T_X = nT_F$$

Reorganización del fichero

Llamemos O al número de registros añadidos y d al número de registros que hay marcados para borrar. Para reorganizarlo, primero debemos leer todos los registros que había antes más los añadios ($n + O$, en n, van incluidos también los que hay marcados para borrar). Después, hay que escribirlos todos otra vez, excepto los marcados para borrar. Importante, no consideramos ningún orden, por lo que no hay que tener en cuenta ningún tiempo de ordenación. Además, como lo insertamos en un archivo totalmente nuevo, los podemos escribir directamente (T_W).

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

1.3.2. Archivo Secuencial Lógico (ASL)

En este tipo de archivo, los registros se encuentran ordenados por una **clave física**, que puede tener uno o varios campos. Además, los registros los consideramos de longitud fija, es decir, tienen los mismos campos y aparecen en el mismo orden. La estructura de los registros se incluye en la cabecera del fichero. Este tipo de registros son muy útiles para hacer estrategias tipo *merge*.

Como los registros están ordenados por una clave, tenemos que insertalos necesariamente en alguna parte del interior del archivo (no al final, en la mayoría de los casos). Esto es terriblemente costoso porque tendríamos que mover todos los registros existentes un lugar hacia abajo para insertar el nuevo registro. Por esta razón, se usa una **zona de desbordamiento** no ordenada, de tipo ASF. Cuando esta zona de desbordamiento crece, tenemos que reconstruir el fichero haciendo un merge del original con este y reescribiéndolo.

Tamaño de registro

Como tienen estructura y longitud fija, simplemente tenemos que considerar lo que ocupa el valor de cada registro (porque la cabecera solo aparece al principio y despreciamos su tamaño):

$$R = \sum_i V_i$$

Recuperación de un registro

Dos casos:

(a) Si el valor de búsqueda no es clave, entonces no podemos aprovechar la única ventaja que nos da la estructura de ASL, es decir, sería igual que el ASF (1.2):

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

(b) Si además de no ser clave, hay registros en la zona de desbordamiento, hay que buscarlo en la zona de desbordamiento, es decir, buscarlo en dos ASF de tamaño n y O (O es el tamaño de la zona de desbordamiento):

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

(c) Si el valor de búsqueda es clave, entonces la eficiencia dependerá del algoritmo de búsqueda que usemos. En el mejor de los casos:

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

Siguiente registro (por clave)

Hay que tener en cuenta donde se encuentra el siguiente registro:

(a) Si el siguiente valor de clave está en el propio fichero, entonces va a estar justo después del registro que tenemos porque los valores están ordenados, entonces solo tenemos que leerlo:

$$T_N = T$$

(b) Si el siguiente valor de clave está en la zona de desbordamiento, entonces leemos nuestro registro más todos los de la zona de desbordamiento para buscarlo:

$$T_N = T + OT$$

Inserción de registro

Como el archivo está ordenado, la inserción es más complicada. Primero tenemos que buscar el sitio que le corresponde al registro, escribirlo y leer y reescribir todos los que haya después de ese:

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

Si la inserción es de varios registros, es más eficiente insertarlos en el zona de desbordamiento directamente:

$$T_I = T_W$$

Pero como después, en algún punto, habrá que insertarlos en el fichero maestro:

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

esta formula no la entiendo muy bien

Actualización de registro

Hay que distinguir dos casos dependiendo de si se cambia el valor de la clave o no:

(a) Si no se cambia el valor de la clave, entonces simplemente hay que encontrarlo y escribir los nuevos valores:

$$T_U = T_F + T_W$$

(b) Si se cambia el valor de la clave, entonces hay que encontrarlo, marcarlo como inválido, e insertarlo:

$$T_U = T_F + T_W + T_I$$

Lectura de fichero

(a) Si sólo se usa archivo maestro:

$$T_X = nT$$

(b) Si se usa un fichero de desbordamiento, primero hay que ordenar ese fichero ($T_C(O)$) y después leer ambos ficheros usando *merge* ($(n+O)T$):

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

Reorganización del fichero

La reorganización de un ASL es muy fácil: consiste en leer el fichero en merge, y reescribir los registros eliminando los borrados:

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

Importante, si no hay zona de desbordamiento no tiene sentido reorganizar el fichero porque ya está ordenado de por sí.

1.3.3. Archivo Secuencial Indexado (ASI)

Este tipo de archivo acelera el acceso por clave. Se busca el valor de clave en el **índice** y éste irá acompañado de una posición en el fichero. Si la posición a la que apunta el índice es un registro, entonces se llama **índice denso**, si apunta a otro índice, se llama **índice no denso**.

La estructura de este fichero usa ASLs, descritos en la sección anterior. Manejamos dos ficheros:

(a) Un fichero de datos: de tipo ASL con una posible área de desbordamiento (pero que se gestiona de forma distinta al ASL).

(b) Un fichero de índice: otro ASL con registros de longitud fija (y estructura uniforme) que contiene valor de clave y dirección del fichero de datos.

Tamaño de registro

Al igual que en el ASL, los registros son de longitud fija y estructura uniforme, luego hay que sumar las longitudes de cada atributo. Peo también hay que tener en cuenta el tamaño del índice, que está formado por la clave (V_K) y el puntero al registro (P):

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

Recuperación de un registro

La recuperación es muy sencilla, buscar el índice y leer el registro:

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

Siguiente registro (por clave)

Como los índices sí están ordenados, simplemente hay que leer el siguiente índice y luego leer el registro al que apunta su puntero, es decir, dos lecturas sólo:

$$T_N = T + T = 2T$$

Inserción de registro

Actualización de registro

Lectura de fichero

Reorganización del fichero

1.3.4. Archivo de Acceso Directo (AAD)

Tamaño de registro

Recuperación de un registro

Siguiente registro (por clave)

Inserción de registro

Actualización de registro

Lectura de fichero

Reorganización del fichero

1.4. Evaluación del sistema