TEMA 3

TRANSFORMACIÓN DE CLAVES (HASHING)

…o como encontrar, con el mínimo esfuerzo, una clave
dada dentro de un conjunto de elementos.

OBJETIVOS DE ESTE CAPITULO:

- Concepto de ‘Hashing’ (Dispersión).
- Colisiones y su tratamiento
- Cómo puede convertirse una clave dada en otra con funciones de trasformación
- Cómo optimizar el concepto de trasformación de claves

INDICE TEMA 3.

1. TRANSFORMACIÓN DE CLAVES

Sea un conjunto **C**:

Dada una clave $\mathbf{C_1}$ de este conjunto, un procedimiento para encontrarla -en el primer intento, si es posible- se basa en lo siguiente:

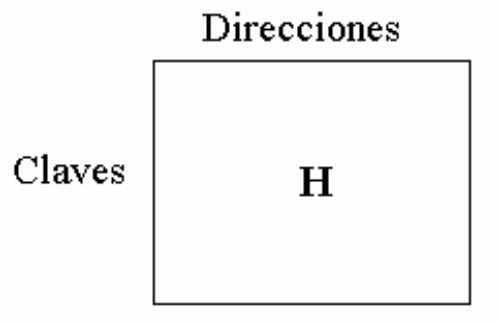
- Cada elemento del conjunto de claves va a representarse mediante una dirección en memoria

El procedimiento será entonces una aplicación :

$$\mathbf{H: \ C \rightarrow D}$$

Función de Transformación	Conjunto de claves	Conjunto de direcciones

El procedimiento puede verse entonces como una tabla:



- y de aquí el nombre de **Transformación de Claves**.

■ **HASH:**

$$\mathbf{H : C \rightarrow D}$$

Se emplea, generalmente, cuando el número de claves es muy grande comparado con el número real de registros que van a manejarse en el fichero.

- El método no es determinista!

Dados $C_1, C_2 \in C / C_1 \neq C_2$;

si $H(C_1) = H(C_2) \Rightarrow$ **Colisión**!

2. MANEJO DE COLISIONES

■ **Colisión**: El lugar en la tabla correspondiente a una cierta clave no contiene el elemento deseado; dos claves corresponden al mismo índice.

● El **tratamiento** de la colisión puede hacerse usando:

→ Almacenamiento externo: el espacio reservado para las colisiones está fuera de los límites de la tabla.

→ Almacenamiento interno: utiliza un método de reasignación para calcular un nuevo índice dentro de la tabla.

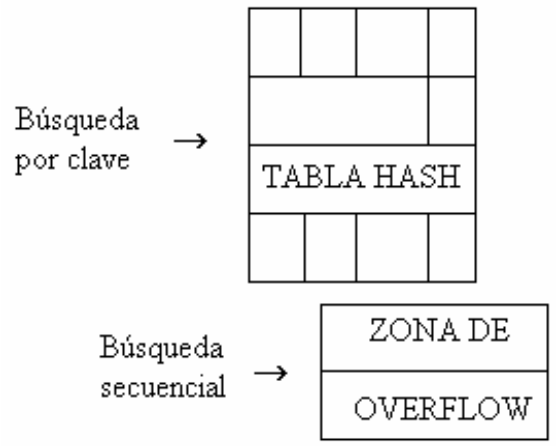
■ Tratamiento de colisiones **externo**

1. **Encadenamiento directo**

Se encadenan todos los elementos cuyas claves generan el mismo índice primario por medio de una lista externa a la tabla; un elemento adicional en cada elemento guarda el puntero a su lista de colisiones.

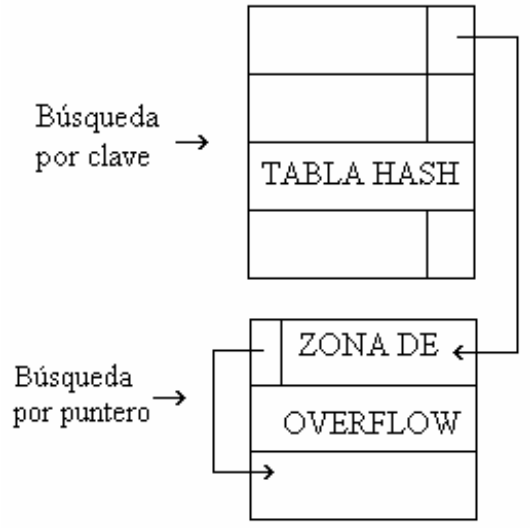
2. **Overflow**

Se reserva una parte de la tabla, llamada área de desbordamiento -aproximadamente un 10%- para almacenar aquellos elementos en colisión. En el área de desbordamiento la búsqueda se realiza de forma secuencial.



3. **Encadenamiento directo externo en zona de Overflow**

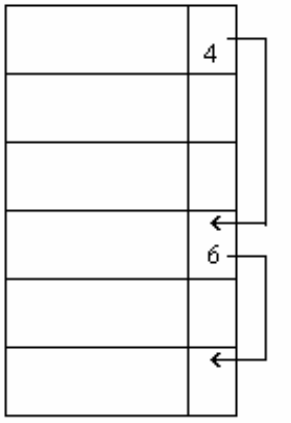
En el caso de producirse nuevas colisiones en el área de overflow, ésta se expande dinámicamente:



■ Tratamiento de colisiones **interno**

1. **Encadenamiento directo**

   Los elementos cuyas claves generan el mismo índice se enlazan a nuevas posiciones, ocupadas por estos elementos, dentro de la misma tabla. Hay que preveer espacio adicional en cada elemento de la tabla para guardar la dirección del siguiente elemento.



2. **Encadenamiento vacío**

   Se buscan otros lugares vacios dentro de la tabla.

   Se evitan enlaces, pero hay que realizar una búsqueda secuencial en la tabla.

   Asimismo, se produce un agrupamiento de claves secundarias detrás de las primarias.

   2.1. Inspección lineal

      Busca el siguiente lugar hasta encontrar el elemento buscado o una posición vacía.

      Los índices se generan de la siguiente forma:

$H (c) = d_0$

a continuación

$h_0 = c \rightarrow d_0 \rightarrow d_1 \rightarrow d_2 \rightarrow \ldots$

donde

$N \rightarrow$ tamaño de la tabla

$h_i = (h_{i-1} + 1) \text{ MOD } N$

o bien $h_i = (h_0 + i) \text{ MOD } N$

Termina cuando, para un d_i , se produce una de estas situaciones:

$T[d_i] = c$ -- se encuentra el elemento

$T[d_i] =$ vacío -- se encuentra una posición vacía

$d_i = d_0$ -- la tabla está llena

2.2 Inspección cuadrática

La generación de índices se efectúa:

$h_0 = c$

$h_i = d_0 + \text{colisiones}^2 \qquad (i > 0)$

3. ELECCIÓN DE LA FUNCIÓN DE TRANSFORMACIÓN

- Consideraciones a tener en cuenta:

  1- La función ha de distribuir las claves de manera uniforme sobre el conjunto de direcciones posibles.

  2- La función ha de ser rápida.

- Adicionalmente,

  3- Ha de producir pocas colisiones.

  4- Ha de ser fácil de calcular

  5- Ha de eliminar la información no distintiva de la clave original

- Tamaño de la tabla: un 10% mayor del tamaño necesario

Supongamos una tabla de **7.000 elementos**.

-Solo para claves numéricas:

1. Método del **centro de los cuadrados**

   Se multiplica la clave por sí misma, y se toman los dígitos centrales, ajustándolo al rango.

   Ej:

   clave $\rightarrow$ 172148
   cuadrado $\rightarrow$ 029634933904

   dígitos centrales ↑

   ajuste al rango $\rightarrow$ 3493 * 0,7 = **2445** Dirección

2. Método de la **división**

   Se divide la clave por, bien un número primo ligeramente inferior al tamaño de la tabla o bien por un nº que contenga factores primos menores que 20. La dirección resultante es el resto de la división.

   Ej:

   clave $\rightarrow$ 172148
   172148 MOD 6997 = 4220
   ajuste de rango $\rightarrow$ 4220 * 0,7 = **2954** Dirección

3. Método de **desplazamiento**

   Los dígitos exteriores se desplazan sobre los centrales, y se suman con éstos.

Ej.

clave → 542422241

2422
542
54
——
2717

ajuste de rango → 2717 * 0,7 = **1894** Dirección

4. Método de plegamiento

Se eligen varias columnas arbitrariamente, y de forma análoga al caso anterior, se pliega el resto sobre ellas, para hacer después una suma o un OR-exclusivo.

Ej:

clave → 542422241

2422
142
45
——
7064

ajuste de rango 7064 * 0,7 = **4944** Dirección

4. FACTOR DE CARGA

- Factor de carga, α: cociente entre el número de claves que hay en la tabla dividido por el número total de claves. Es una medida del rendimiento de la tabla hash, -número de colisiones que se producen-.

  - Para una tabla vacía:

$$\alpha = 0$$

  - Para una tabla vacía:

$$\alpha = \frac{n}{n + 1}$$

  - Es directamente proporcional al nº de colisiones.

5. BUCKETS (Compartimentos)

- 'Bloque de registros que se extraen en un acceso a disco cuando comparten la misma dirección'

  - Empleado en trabajo sobre archivos en disco; menos importante cuando se trabaja sobre memoria

  - Un compartimento está formado por uno o más sectores del disco

- Modo de trabajo:

  - Con la función de Trasformación se obtiene la dirección del compartimento 'base', y a partir de ahí, se continua la búsqueda mediante las técnicas conocidas.

- **Ventajas**:

  - Minimiza el número de colisiones (nueva colisión $\rightarrow$ cuando el bucket esté completo.

  - Minimiza el tiempo de acceso a las claves y el espacio utilizado en el almacenamiento.

  - El tamaño de la tabla Hash es menor.