

# Fundamentos de Organización de Datos

Clase 7

# Agenda

## Hashing

- Definición
- Tipos
- Propiedades

## Propiedades

- Función de hash
- Densidad / tamaño nodo
- Tratamiento del overflow

## Dispersión

- Estática
- Dinámica

# Hashing (Dispersión) → Introducción

Necesitamos un mecanismo de acceso a registros con una lectura solamente

- **Secuencia:**  $N/2$  accesos promedio
- **Ordenado:**  $\log_2 N$
- **Árboles:** 3 o 4 accesos



Clave Primarias → características

- **No se repiten**
- **El resto de las claves actúan a través de ella**
- Cuando se aprenda a modelar, tendrán más características que las hacen especiales

# Hashing (Dispersión) → Definición

Técnica para generar una dirección base única para una llave dada. La dispersión se usa cuando se requiere acceso rápido a una llave



Técnica que convierte la llave del registro en un número aleatorio, el que sirve después para determinar donde se almacena el registro.

Técnica de almacenamiento y recuperación que usa una función de hash para mapear registros en dirección de almacenamiento.

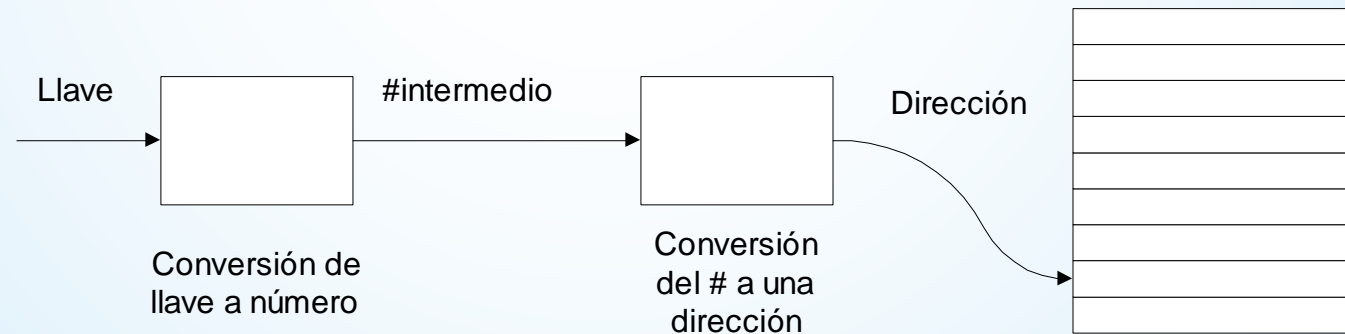
# Hashing (Dispersion) → definición

- Archivos secuenciales indizados
  - Archivo de datos
  - Archivo con índice primario
  - Archivos con índices únicos o secundarios
- Archivos directos
  - UN ACCESO
  - No puede haber estructuras adicionales
  - Se organiza EL archivo de datos
  - Solo puede organizarse por un UNICO criterio
  - Ese criterio es la clave primaria

# Hashing (Dispersión) → Definición

## Atributos del hash

- No requiere almacenamiento adicional (índice)
- Facilita inserción y eliminación rápida de registros
- Encuentra registros con muy pocos accesos al disco en promedio



# Hashing (Dispersión) → Definición

## Costo

- No podemos usar registros de longitud variable
- No puede haber orden físico de datos
- No permite llaves duplicadas

## Para determinar la dirección

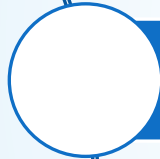
- La clave se convierte en un número casi aleatorio
- # se convierte en una dirección de memoria
- El registro se guarda en esa dirección
- Si la dirección está ocupada → colisión/overflow (tratamiento especial)

# Hashing (Dispersión) → Parámetros

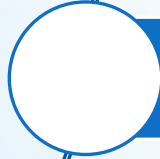
E  
F  
I  
C  
I  
E  
N  
C  
I  
A



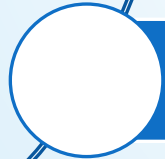
Función de hash



Tamaño de los nodos



Densidad de empaquetamiento



Método de tratamiento de desbordes



# Hashing (Dispersión) → Parámetros

## 1. Función de hash

- Caja negra que a partir de una clave se obtiene la dirección donde debe estar el registro.
- Diferencias con índices
  - Dispersión no hay relación aparente entre llave y dirección
  - Dos llaves distintas pueden transformarse en iguales direcciones (colisiones)

# Hashing (Dispersión) → parámetros

## Colisión:

- Situación en la que un registro es asignado a una dirección que está utilizada por otro registro

## Overflow



- Situación en la que un registro es asignado a una dirección que está utilizada por otro registro y no queda espacio para este nuevo

## Soluciones

- Algoritmos de dispersión sin colisiones o que estas colisiones nunca produzcan overflow (perfectos) (imposibles de conseguir)
- Almacenar los registros de alguna otra forma, esparcir

# Hashing (Dispersión) → Parámetros

## Soluciones para las colisiones

- Esparcir registros: buscar métodos que distribuyan los registros de la forma más aleatoria posible
- Usar memoria adicional: distribuir pocos registros en muchas direcciones, baja la densidad de empaquetamiento:
  - Disminuye el colisiones y por ende overflow
  - Desperdicia espacio
- Colocar más de un registro por dirección: direcciones con N claves, mejoras notables

# Hashing (Dispersión) → Parámetros

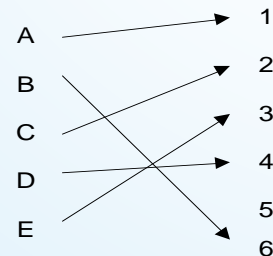
## Algoritmos simples de dispersión

- Condiciones
  - Repartir registros en forma uniforme
  - Aleatoria (las claves son independientes, no influyen una sobre la otra)

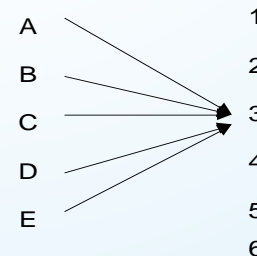


## Tres pasos

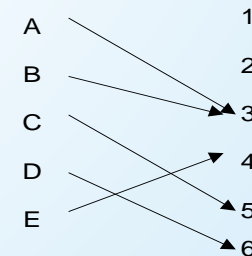
- Representar la llave en forma numérica (en caso que no lo sea)
- Aplicar la función
- Relacionar el número resultante con el espacio disponible



uniforme



peor



aceptable

# Hashing (Dispersión) → Parámetros

## 2. Tamaño de las cubetas


- Puede tener más de un registro
- A mayor tamaño
  - Menor overflow
  - Mayor fragmentación
  - Búsqueda más lenta dentro de la cubeta (este concepto realmente afecta al problema?)

# Hashing (Dispersión) → Parámetros

## 3. Densidad de empaquetamiento

- Proporción de espacio del archivo asignado que en realidad almacena registros
- $DE = \frac{\text{número de registros del archivo}}{\text{capacidad total del archivo}}$
- Densidad de empaquetamiento menor
  - Menos overflow
  - Más desperdicio de espacio

# Hashing → estimacion del Overflow

- Es necesario analizar el comportamiento de un archivo directo
- Cuando encontrar un registro requiere un solo acceso y cuando requiere mas cantidad de accesos
- Estimar el Overflow 
  - Analizar probabilisticamente si la insercion de un registro genera o no colision
  - Analizar si la colisión genera o no overflow
- Es necesario
  - Conocer elementos básicos de probabilidades
  - Vamos a utilizar la distribucion de Poisson



# Hashing (Dispersión) → Parámetros

## Estimación del overflow → sabiendo que

- N # de cubetas,
- C capacidad de nodo,
- R # reg. Del archivo
- $DE = \frac{R}{C \times N}$
- Probabilidad que una cubeta reciba I registros (distribución de Poisson)

$$P(I) = \frac{R!}{I! * (R - I)!} * \left(\frac{1}{N}\right)^I * \left(1 - \frac{1}{N}\right)^{R-I}$$



# Hashing (Dispersión) → Parámetros

Por que?Cuál es la justificación de la fórmula anterior?

- Supongamos que
  - A: no utilizar un cubeta particular
  - B: utilizar una cubeta en particular
- $P(B) = 1/N$        $P(A) = 1 - P(B) = 1 - 1/N$
- Si tenemos dos llaves?
  - $P(BB) = P(B) * P(B) = (1/N)^2$  (porque se puede asegurar esto?)
  - $P(BA) = P(B) * P(A) = (1/N) * (1 - 1/N)$
  - $P(AA) = P(A) * P(A) = (1 - 1/N)^2$

# Hashing (Dispersión) → Parámetros



Si la secuencia fuera de tres claves

- $P(BBB) = P(B) * P(B) * P(B) = (1/N)^3$
- $P(BAA) = P(B) * P(A) * P(A) = (1/N) * (1-1/N)^2$
- $P(AAA) = P(A) * P(A) * P(A) = (1-1/N)^3$
- Cuantas combinaciones? → 8

# Hashing (Dispersión) → Parámetros

- En general → si fueran R claves

- $P(A...AB...B)$  siendo la suma de A y B igual a R

- Que nos interesa → que I registros vayan a un nodo

- ESTO QUE SIGNIFICA



- B...B (I veces)

- A...A (R-I veces)

- $P(B)$  i veces

- $P(A)$  R-I Veces

- $(1/N)^i * (1-1/N)^{R-i}$

# Hashing (Dispersión) → Parámetros

- Ahora analicemos la siguiente situación
- $P(B..B A..A)$  siendo  $I$  la cantidad de  $B$  y  $R-I$  la cantidad de  $A$ 
  - $(1/N)^i * (1-1/N)^{R-I}$
- $P(B A.. A B.. B)$  siendo  $I$  la cantidad de  $B$  y  $R-I$  la cantidad de  $A$ 
  - $(1/N)^i * (1-1/N)^{R-I}$
- $P(A.. A B.. B)$  siendo  $I$  la cantidad de  $B$  y  $R-I$  la cantidad de  $A$ 
  - $(1/N)^i * (1-1/N)^{R-I}$
- Todas las anteriores combinaciones dan la misma probabilidad
  - Cuantas combinaciones se pueden hacer
  - $R$  tomadas de a  $I$   $\left( \frac{R!}{I! * (R-I)!} \right)$



# Hashing (Dispersión) → Parámetros

$$P(I) = \frac{R!}{I! * (R - I)!} * \left(\frac{1}{N}\right)^I * \left(1 - \frac{1}{N}\right)^{R-I}$$

En general la secuencia de R llaves, que I caigan en un nodo es la probabilidad

$$\left(\frac{1}{N}\right)^I * \left(1 - \frac{1}{N}\right)^{R-I}$$

Cuántas formas de combinar esta probabilidad hay (R tomadas de a I combinaciones)

$$\frac{R!}{I! * (R - I)!}$$

Función de Poisson: (probabilidad que un nodo tenga I elementos) R, N, I con la definición ya vista

$$P(I) = \frac{(R/N)^I * e^{-(R/N)}}{I!}$$

# Hashing (Dispersión) → Parámetros

## Análisis numéricos de Hashing

- En general si hay  $n$  direcciones, entonces el # esperado de direcciones con  $l$  registros asignados es  $N * P(l)$ .
- Las colisiones aumentan con al archivo más "lleno"
- Ej:  $N = 10000$     $R = 10000$     $DE = 1$     $100\%$



|                 |   |       |      |
|-----------------|---|-------|------|
| $P(0) = 0.3679$ |   | 3679  |      |
| $P(1) = 0.3679$ | * | 10000 | 3679 |
| $P(2) = 0.1839$ |   | 1839  |      |
| $P(3) = 0.0613$ |   | 613   |      |

qué significa?

$$\text{overflow} = 1839 + 2 * 613 = 3065 \quad (\text{alto})$$

# Hashing (Dispersión) → Parámetros

Ahora supongamos que el problema es

- $R = 500$      $N = 1000$      $DE = 50\%$

$$P(0) = 0.607 \quad 607$$

$$P(1) = 0.303 \quad * 1000 \quad 303$$

$$\text{saturación} = N * [ 1 * P(2) + 2 * P(3) + 3 * P(4) + 4 * P(5) ] = 107$$

- **Saturación menor**

| densidad | overflow |
|----------|----------|
| 10%      | 4.8%     |
| 50%      | 21.4%    |
| 100%     | 36.8%    |

- **los números bajos de overflow (baja densidad) → muchas cubetas libres**



# Hashing (Dispersión) → Parámetros

Que pasa si mantenemos la DE pero cambiamos ciertos valores

- EJ:

$R = 750$

$N = 1000$

$C = 1$



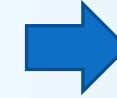
$DE = 75\%$

$R/N = 0,75$

$R = 750$

$N = 500$

$C = 2$



$DE = 75\%$

$R/N = 1,5$

deben influir en la función de Poisson

saturación

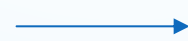
$c = 1$



222

cubetas

$c = 2$



140

cubetas

- Cual es el tamaño de la cubeta?



# Hashing (Dispersión) → Parámetros

| DE   | 1    | 2    | 5    | 10   | 100 |
|------|------|------|------|------|-----|
| 10%  | 4.8  | 0.6  | 0.0  | 0.0  | 0.0 |
| 20%  | 9.4  | 2.2  | 0.1  | 0.0  | 0.0 |
| 30%  | 13.6 | 4.5  | 0.4  | 0.0  | 0.0 |
| 40%  | 17.6 | 7.3  | 1.1  | 0.1  | 0.0 |
| 50%  | 21.3 | 10.4 | 2.5  | 0.4  | 0.0 |
| 60%  | 24.8 | 13.7 | 4.5  | 1.3  | 0.0 |
| 70%  | 28.1 | 17.0 | 7.1  | 2.9  | 0.0 |
| 75%  | 29.6 | 18.7 | 8.6  | 4.0  | 0.0 |
| 80%  | 31.2 | 20.4 | 10.3 | 5.3  | 0.1 |
| 90%  | 34.1 | 23.8 | 13.8 | 8.9  | 0.8 |
| 100% | 36.8 | 27.1 | 17.6 | 12.5 | 4.0 |



# Hashing (Dispersión) → Parámetros

## Tratamiento de Colisiones con Overflow

- Hemos visto que el % de overflow se reduce, pero el problema se mantiene dado que no llegamos a 0%

## Algunos métodos

- Saturación progresiva
- Saturación progresiva encadenada
- Doble dispersión
- Área de desborde separado

# Hashing (Dispersión) → Parámetros

## Saturación progresiva:

- Cuando se completa el nodo, se busca el próximo hasta encontrar uno libre.
- Búsqueda?
- Eliminación, no debe obstaculizar las búsquedas

# Hashing (Dispersión) → Parámetros

## saturación progresiva encadenada

- similar a saturación progresiva, pero los reg. de saturación se encadenan y “no ocupan” necesariamente posiciones contiguas
- Ejemplo

# Hashing (Dispersión) → Parámetros

## Dispersión doble:

- saturación tiende a agrupar en zonas contiguas, búsquedas largas cuando la densidad tiende a uno
- Solución almacenar los registros de overflow en zonas no relacionadas.
- esquema con el cual se resuelven overflows aplicando una segunda función a la llave para producir un  $N^{\circ} C$ , el cual se suma a la dirección original tantas veces como sea necesario hasta encontrar una dirección con espacio.

# Hashing (Dispersión) → Parámetros

## Encadenamiento en áreas separadas:

- No utiliza nodos de direcciones para los overflow, estos van a nodos especiales
- Ejemplo:
- Se mejora el tratamiento de inserciones o eliminaciones. Empeora el TAP.
- Ubicación del desborde
  - A intervalos regulares entre direcciones asignadas
  - Cilindros de desborde

# Hashing (Dispersión)

## Hash con espacio de direccionamiento estático

- Necesita un número de direcciones fijas, virtualmente imposible
- Cuando el archivo se llena
  - Saturación excesiva
  - Redispersar, nueva función, muchos cambios

## Solución → espacio de direccionamiento dinámico

- Reorganizar tablas sin mover muchos registros
- Técnicas que asumen bloques físicos, pueden utilizarse o liberarse.



# Hashing (Dispersión) → espacio dinámico

## Varias posibilidades

- Hash virtual
- Hash dinámico
- Hash Extensible

## Hash Extensible

- Adapta el resultado de la función de hash de acuerdo al número de registros que tenga el archivo, y de las cubetas necesitadas para su almacenamiento.
- Función: Genera secuencia de bits (normalmente 32)



# Hashing (Dispersión) → espacio dinámico

## Como trabaja

- Se utilizan solo los bits necesarios de acuerdo a cada instancia del archivo.
- Los bits tomados forman la dirección del nodo que se utilizará
- Si se intenta insertar a una cubeta llena deben reubicarse todos los registros allí contenidos entre el nodo viejo y el nuevo, para ello se toma un bit más.
- La tabla tendrá tantas entradas (direcciones de nodos) como  $2^i$ , siendo  $i$  el número de bits actuales para el sistema.

# Hashing (Dispersión) → espacio dinámico (ejemplo)

| Clave   | Secuencia de bits |
|---------|-------------------|
| Alfa    | .... 0011 0011    |
| Beta    | .... 0110 0101    |
| Gamma   | .... 1001 1010    |
| Epsilon | .... 0111 1100    |
| Delta   | .... 1100 0001    |
| Tita    | .... 0001 0110    |
| Omega   | .... 1111 1111    |
| Pi      | .... 0000 0000    |
| Tau     | .... 0011 1011    |
| Lambda  | .... 0100 1000    |
| Sigma   | .... 0010 1110    |

# Elección de organización

## Archivos

- Acomodar datos para satisfacer rápidamente requerimientos
- Accesos: resumen

| Organización | Acc.un reg. CP | Todos reg. CP |
|--------------|----------------|---------------|
| Ninguna      | Lento          | Lento         |
| Secuencial   | Lento          | Rápido        |
| Index sec.   | Buena          | Rápida        |
| Hash         | Rápido         | lento         |

# Elección de organización

## Elección de organización

- Captar los requerimientos de usuario
- Que examinar
  - Características del archivo
    - Número de registros, tamaño de registros
  - Requerimientos de usuario
    - Tipos de operaciones, número de accesos a archivos
    - Características del hard
      - Tamaño de sectores, bloques, pistas, cilindros, etc.
    - Parámetros
    - Tiempo (necesario para desarrollar y mantener el soft, para procesar archivos)
    - Uso promedio (# reg. Usados/ #registros)