1. Función de dispersión (hashing)

* 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.

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



Función del hash

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

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 esta utilizada por otro registro y no queda espacio para este nuevo

Solución 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:
  1. Disminuye las colisiones y por ende overflow
  2. Desperdicia espacio
* Colocar más de un registro por dirección: direcciones con N claves, mejoras notables

Tamaños de las cubetas:

* Puede tener más de un registro
* A mayor tamaño
  1. Menor overflow
  2. Mayor fragmentación
  3. Búsqueda más lenta dentro de la cubeta

Densidad de empaquetamiento:

* Proporción de espacio del archivo asignado que en realidad almacena registros
* DE = número de registros del archivo  
   capacidad total del archivo
* Densidad de empaquetamiento menor
  1. Menos overflow
  2. Más desperdicio de espacio

Datos a tener en cuenta en el hashing:

* N # de cubetas,
* C capacidad de nodo,
* R # reg. Del archivo
* DE = R  
   C x N
* Probabilidad que una cubeta reciba I registros (distribución de Poisson)

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
* P(BA) = P(B) \* P(A) = (1/N) \* (1 – 1/N)
* P(AA) = P(A) \* P(A) = (1 – 1/N)2
* 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

Tratamiento de Colisiones con Overflow:

* Saturación progresiva
  + Cuando se completa el nodo, se busca el próximo hasta encontrar uno libre.
  + Búsqueda?
  + Eliminación, no debe obstaculizar las búsquedas
* Saturación progresiva encadenada
  + similar a saturación progresiva, pero los reg. de saturación se encadenan y “no ocupan” necesariamente posiciones contiguas
* Doble dispersión
  + 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 Nº C, el cual se suma a la dirección original tantas veces como sea necesario hasta encontrar una dirección con espacio.
* Área de desborde separado
  + 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

Hash con espacio de direccionamiento estático

* Necesita un número de direcciones fijas, virtualmente imposible
* Cuando el archivo se llena
  1. Saturación excesiva
  2. 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.

Hay varias posibilidades para solucionarlo

* Hash virtual
* Hash dinámico
* Hash Extensible

El 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)
* 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 2i, siendo i el número de bits actuales para el sistema.