

# WUOLAH



bebesitaaaaa

[www.wuolah.com/student/bebesitaaaaa](http://www.wuolah.com/student/bebesitaaaaa)



2813

## Resumen-tema-5.pdf

RESUMEN TEMAS 1-5



2º Fundamentos de Bases de Datos



Grado en Ingeniería Informática



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

**Como aún estás en la portada, es  
momento de redes sociales.  
Cotilléanos y luego a estudiar.**



# WUOLAH

# TEMA 5: NIVEL INTERNO

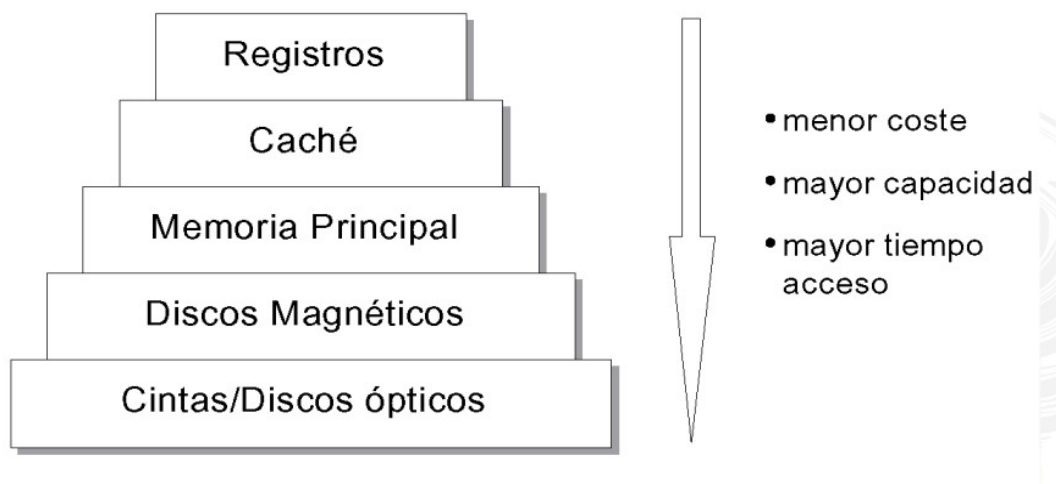
## 1. Conceptos básicos

Una BD sirve para almacenar de forma permanente grandes cantidades de datos, con el propósito principal de gestionar de forma eficiente los datos y su almacenamiento. Esto tiene sus consecuencias tanto en la organización lógica de los datos, como en su organización física.

El **nivel interno** expresa las operaciones sobre los datos en términos de actuación sobre unidades mínimas de almacenamiento denominadas páginas o bloques de BD. Provee al administrador de mecanismos para optimizar el almacenamiento y el acceso a los datos. Se encuentra implementado en el SGBD.

El **nivel físico** se encuentra implementado en el SO. Proporciona al SGBD una capa de abstracción sobre el hardware. Realiza el acceso a los medios de almacenamiento mediante llamadas a los servicios del sistema de archivos proporcionado por el SO.

## 2. Dispositivos de almacenamiento



### **Memoria principal**

Es el dispositivo de almacenamiento primario de los ordenadores.

- Hace trabajos de caché de la porción de la BD de uso más reciente.
- Elemento de almacenamiento intermedio que ubica de forma temporal los datos afectados por las operaciones.
- Como es rápida y cara, el nivel interno debe optimizar su uso para acelerar el procesamiento de los datos.
- Tanto el disco duro como la memoria principal utilizan distintos niveles de caché para acelerar el acceso a los datos.

### **Discos Duros**

Dispositivo de almacenamiento más usado en BD, constituido por un conjunto de discos magnéticos con dos caras. Cada cara tiene un conjunto de pistas concéntricas y cada pista se divide en sectores con la misma capacidad de almacenamiento (bloque). Localización de un bloque:

- Cilindro.
- Superficie de disco.
- Sector



Formación  
Manuel  
Pozo

# ACADEMIA UNIVERSITARIA MP

Academia especializada en grados universitarios.  
Cursos intensivos y clases particulares.

Profesores especializados en más de 150 asignaturas.  
Consulta todas tus asignaturas.

Matemáticas

Química

Física

Bilología

Bioquímica

Ambientales

Geología

Óptica

Estadística

Tecnología

Farmacia

Nutrición

Ingeniería

Economía


Medicina

Odontología

Psicología

Magisterio.

 [www.formaciónmanuelpozo.com](http://www.formaciónmanuelpozo.com)

 615 14 96 76

 Granada



## Medidas de rendimiento

- Tiempo medio de acceso ( $t_a$ ): tiempo medio transcurrido entre una instrucción y la obtención de la información.
- Tiempo medio de búsqueda ( $t_b$ ): tiempo medio de posicionamiento en pista.
- Tiempo de latencia rotacional ( $t_l$ ): tiempo medio de posicionamiento en sector.
- $t_a = t_b + t_l$
- Tiempo medio entre fallos (MTBF)

## 3. Métodos de acceso a la BD almacenada

**¿Cómo se transforma un registro almacenado en una representación física en el almacenamiento secundario?**



Para que el gestor de almacenamiento pueda localizar un registro almacenado, se utiliza el RID (RecordIdentifier).

Cada registro almacenado tiene:

- Cabecera: número y tipo de columnas que lo integran.
- Datos: contenido de las columnas.

Las páginas o bloques de la BD deben tener un tamaño múltiplo de las páginas del SO. Para recuperar un registro almacenado hay que determinar la página de BD que lo contiene y entonces recuperar los bloques de disco que lo integran. Hay que organizar la estructura de almacenamiento y los métodos de acceso, de forma que se optimice la interacción con los dispositivos de almacenamiento secundario.

### Gestor de disco del SO

Normalmente el SGBD interactúa con la BD almacenada en el sistema de almacenamiento secundario a través del gestor de disco del SO. El gestor de disco organiza los datos en conjuntos de bloques o archivos de SO. Una BD puede valerse de uno o varios de estos archivos para almacenar su contenido. También se encarga de gestionar el espacio libre en el disco. Las funciones del gestor de disco del SO son:

- Crear un nuevo archivo de SO.
- Eliminar un archivo de SO existente.
- Añadir un bloque nuevo al conjunto de bloques c.
- Eliminar el bloque b del conjunto de bloques c.
- Devolver el bloque b del conjunto de bloques c.
- Reemplazar el bloque b dentro del conjunto de bloques c.

## Gestor de archivos del SGBD

Componente del SGBD que se encarga de:

- Hacer la transformación entre campos, registros y archivos almacenados a bloques y conjuntos de bloques que pueda entender el gestor de disco.
- Organizar los datos de manera que se minimice el tiempo de recuperación. Minimizar las E/S a disco.

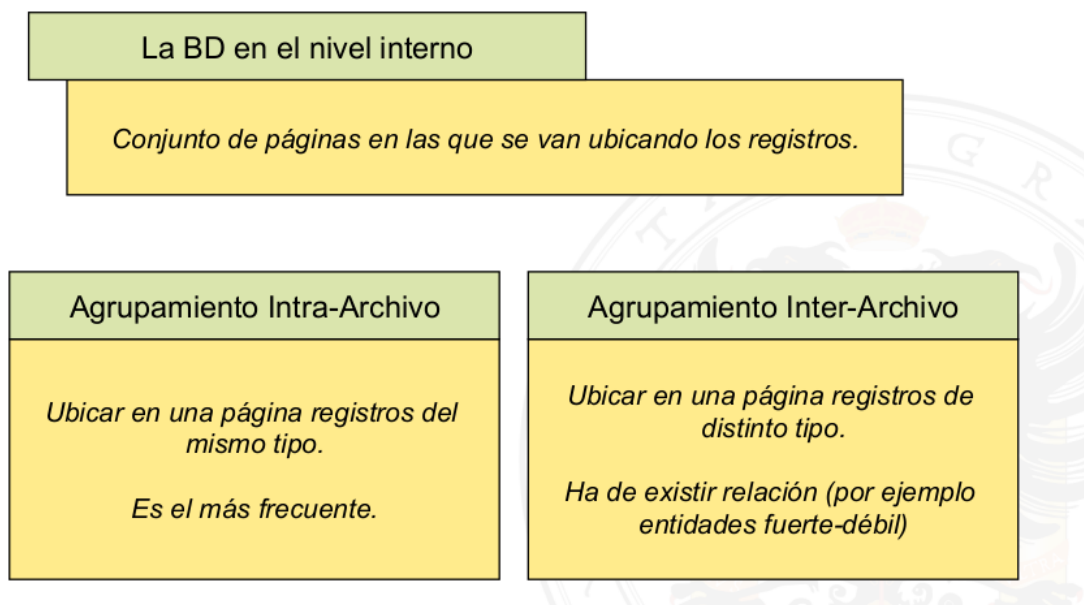
Las funciones del gestor de archivos del SGBD son:

- Crear un nuevo archivo almacenado. Asociar al archivo un conjunto de páginas o bloques de la BD.
- Eliminar un archivo almacenado.
- Recuperar el registro almacenado *r* del archivo almacenado *a*. Normalmente, el SGBD proporciona el RID. Sólo hay que obtener en memoria la página que contiene el registro para extraerlo.
- Añadir un nuevo registro almacenado al archivo almacenado *a*. Hay que localizar la página de BD más apropiada de las pertenecientes al archivo almacenado. Si no se pudiera, se solicita una nueva página. Se devuelve al SGBD el RID nuevo.
- Eliminar el registro *r* del archivo almacenado *a*. Hay que recuperar la página de BD que contiene dicho registro y marcar el espacio ocupado por el registro en dicha página como disponible.
- Actualizar el registro *r* en el archivo almacenado *a*. Recupera la página de la BD que contiene el registro que se desea actualizar. Trata de sustituir la información. Si no puede, se intenta ubicar en otra página.

## 4. Representación de la BD en el nivel interno

La BD se representa de diferentes formas en los diferentes niveles de la arquitectura del SGBD. Su representación en el nivel interno no tiene por qué coincidir con su representación en el nivel conceptual. Cada conjunto de registros del mismo tipo no tiene por qué ser un mismo archivo. El nivel interno debe traducir las estructuras del nivel conceptual a otras estructuras intermedias más cercanas al almacenamiento real de los datos (nivel físico).

### Agrupamiento



La organización descrita es un ejemplo general. Cada SGBD utiliza su variante concreta, aunque la idea general es la misma. No existe una relación directa fichero-almacenado/fichero-físico, ya que todos los conjuntos de páginas irán almacenados, con toda probabilidad, en uno o varios ficheros físicos.

## 5. Organización y métodos de acceso

El objetivo es minimizar el número de accesos a disco y por tanto minimizar la cantidad de páginas de BD involucradas en una operación de BD. Ninguna de las organizaciones presentadas es mejor en términos absolutos. Criterios básicos para medir la calidad de una organización son:

- **Tiempo de acceso** a los datos requeridos.
- Porcentaje de **memoria ocupada** por los datos requeridos con respecto a las páginas de BD que los contiene.

Trabajaremos a dos niveles:

- Organización de **registros** de datos a nivel de almacenamiento.
- Adición de **estructuras** complementarias para acelerar el acceso a dichos registros.

## 6. Organización secuencial

Fichero de acceso secuencial:

- Aquel donde los registros están almacenados consecutivamente.
- Para acceder a un registro determinado debemos pasar obligatoriamente por los registros que le preceden.
- Los registros suelen estar ordenados por una clave.

Inserción de un nuevo registro:

- Buscar el bloque que le corresponde. Si hay sitio, se inserta el nuevo registro. En caso contrario, o bien se opta por crear un nuevo bloque o bien se crea un bloque de desbordamiento.
- Es recomendable dejar espacio vacío en los bloques para evitar los problemas de reorganización.

Borrado de un registro:

- Buscar el registro.
- Puede implicar una reorganización local de los registros de un bloque.

En resumen, las operaciones suponen:

- Escritura del bloque del registro que se inserta o borra.
- Creación o liberación de bloques de datos en el fichero secuencial.
- Creación o liberación de bloques de desbordamiento.
- Reorganización de registros entre bloques contiguos, lo que implica la escritura de los bloques implicados en el desplazamiento.

Como puede verse, de esta forma de organizar los registros no está exenta de grandes inconvenientes. Pueden subsanarse mediante el uso de estructuras adicionales que nos permitan:

- Acelerar la localización de los datos.
- Disminuir el número de bloques de disco transferidos.

Entre las técnicas más usadas se encuentra:

- Índices.
- Acceso directo.

## 7. Indexación

Tiene por objetivo disminuir el tiempo de acceso a los datos por una clave de búsqueda. Similar a la idea de un índice de un libro.

### **Ficheros indexados**

Partimos de un fichero secuencial pero disponemos de una estructura adicional, un fichero índice.

- Sus registros poseen:
  - **Campo clave** (la clave de búsqueda).
  - **Campo** de referencia que contiene **RIDs** de registros.
- Son más pequeños que los del fichero de datos, aunque el número de ellos es el mismo en ambos ficheros.
- **Índice primario:** la clave de búsqueda es el mismo campo clave por el que está ordenado el fichero de datos.
- **Índice secundario:** contruidos sobre otros campos que no sean la clave física del fichero de datos.
- Proceso de consulta:
  - Consulta por un valor de la clave:
    - Sobre el índice localizamos la clave (recorrido secuencial).
    - Obtenemos el RID del registro requerido.
    - Vamos a disco para recuperar el bloque de datos donde se encuentra el registro señalado por el RID.
    - La búsqueda en el índice es más rápida.
  - Consulta por rango de valores:
    - Búsqueda en el índice por valor de clave de la cota inferior.
    - Recorrido de las entradas del índice que están en el intervalo, recuperando los registros correspondientes gracias a su RID.
- Inserción en un nuevo registro:
  - Mismas operaciones que en el fichero secuencial.
  - Hay que actualizar también el índice.
- Borrado de un registro:
  - Borrado de un registro en el fichero de datos.
  - Borrado de una entrada en el índice.
- Se puede montar un índice sobre más de un campo de un registro.
  - **Clave:** concatenación de los campos indicados.

En resumen, los índices aceleran el acceso a los datos, pero ralentizan las otras operaciones. Por lo tanto, hay que considerar la conveniencia de crear cada índice teniendo en cuenta la frecuencia de las consultas y la frecuencia de las operaciones de mantenimiento de los datos.

## 8. Índices no densos

En realidad los índices siguen siendo muy grandes, porque contienen todos los registros del fichero que indexan. Para reducir el tamaño aparecen los índices no densos:

- Registros compuestos por:
  - La clave de búsqueda.
  - La dirección de comienzo del bloque donde puede encontrarse el registro deseado.
- El número de registros se reduce al número de bloques del fichero de datos. El acceso secuencial al índice no denso se acelera.

Las diferencias en el proceso de búsqueda son:

- Una vez encontrado el bloque donde podría encontrarse el registro:
  - Hay que cargarlo en memoria.
  - Hay que hacer una búsqueda secuencial. No tiene costes en términos de acceso a disco.
- No se tiene garantía alguna de encontrar el registro deseado hasta consultar el bloque de datos leído.

El mantenimiento de un índice no denso es menos costoso:

- Inserción y borrado menos frecuentes.
- Sólo ocurren cuando la operación afecta al valor representativo del bloque.
- Los índices no densos sólo se pueden definir sobre la clave física.

## **9. Índices jerárquicos**

Volvemos al objetivo de disminuir el tiempo necesario para recorrer el índice en busca de un registro. Para ello se crean **índices sobre índices**, generando varios niveles en el acceso a los datos.

Un índice multinivel está formado por:

- Un índice de primer nivel sobre el fichero de datos.
  - Puede ser denso o no dependiendo de la clave.
- Otros índices, no densos, contruidos sucesivamente unos sobre otros.

El tamaño de los bloques se establece con la idea de optimizar cada una de las operaciones de acceso al disco físico. Se reduce el número de accesos a disco para localizar un registro. Se complica el mantenimiento del índice.

## **10. Árboles B+**

Son una generalización de los árboles binarios balanceados en la que los nodos pueden tener más de dos hijos. Todos los valores de la clave se encuentran almacenados en los nodos hoja.

Un Árbol B+ de orden M (el máximo número de hijos que puede tener cada nodo) es un árbol con la siguiente estructura:

- Nodo de nivel superior: raíz.
- Nodos del nivel inferior: hojas.
- Cada nodo distinto de las hojas tiene como máximo M hijos.
- Cada nodo (excepto raíz y hojas) tienen como mínimo  $(M+1) / 2$  hijos.
- La raíz tiene al menos 2 hijos si no es un nodo hoja.
- Todos los nodos hoja aparecen al mismo nivel.
- Las claves contenidas en cada nodo nos guiarán hasta el siguiente nodo del nivel inmediatamente inferior.
- Un nodo no hoja con n hijos contiene:
  - n-1 valores de clave almacenados.
  - n punteros  $P_i$  que apuntan a un nodo hijo.

### **Restricciones dentro de los nodos**

- Los valores de la clave  $C_i$  están ordenados dentro del nodo.
- Los valores x del subárbol apuntado por  $P_i$  cumplen:
  - $C_{i-1} \leq x < C_i$
  - Excepto para:
    - $i = 1$ , donde  $x < C_1$
    - $i = v$ , donde  $x \geq C_v$

### **Nodos hoja**

- Tienen una estructura diferente:
  - Pareja clave – RID
  - Punteros al siguiente nodo hoja.
  - Algunas variantes también tienen punteros al nodo hoja anterior.
- La lista concatenada de nodos hoja, tiene gran utilidad a la hora de hacer consultas por intervalos.



## Restricciones en nodos hoja

- Las claves aparecen ordenadas en cada nodo.
- Todas las claves han de ser menores que las del siguiente nodo en el conjunto secuencia.
- Los nodos han de estar como mínimo rellenos hasta la mitad.
- Todos los nodos hoja se encuentran en el mismo nivel.
  - Árbol equilibrado.
  - Todos los caminos desde la raíz a un nodo hoja tienen la misma longitud.

## Proceso de consulta

- Localización de un registro:
  - Navegamos desde la raíz, bajando niveles.
  - Buscamos el registro en el nodo hoja y, en su caso, recuperamos el registro del fichero de datos gracias al RID.
- Consultas por rango:
  - Se localiza el nodo hoja que contiene el valor inferior.
  - Se recorren los nodos hoja hasta alcanzar el superior, recuperando los registros pertinentes del fichero de datos.

## Inserción y borrado

Se utilizan algoritmos que garantizan que el árbol resultante sea equilibrado.

## 11. Árboles B

Los Árboles B son una variante de B+ en la que no almacenan todos los valores de la clave en los nodos hoja, si no que algunos valores se van almacenando en los nodos intermedios conforme se crea el árbol.

## 12. Uso de Árboles B+ en BD

Son variaciones del Árbol B+, de orden elevado, en la que se procura que cada nodo tenga una capacidad de almacenamiento similar al tamaño de un bloque de datos. Esto reduce los accesos a disco que suelen ser los que determinan el rendimiento de las búsquedas en BD. En los nodos intermedios sólo están los rangos de los valores de la clave y los punteros a los nodos hijo correspondientes.

En los nodos hoja se encuentran todos los valores de la clave ordenados junto con los RIDs que apuntan a las tuplas que contienen ese valor de la clave. Los nodos hoja, que forman el conjunto secuencia, se encuentran enlazados para poder recuperar por búsquedas secuenciales, a veces se encuentran doblemente enlazados, para facilitar búsquedas ascendentes y descendentes por el valor de la clave.

## Tablas Organizadas por Índice (IOT)

Las hojas contienen las tuplas en lugar del RID. Una IOT sólo puede estar organizada de esta forma mediante una clave, aunque sea pueden definir índices adicionales basados en otras claves.

Tabla Normal	Index-Organized Table
Identificador único ROWID (RID)	Identificado por la clave primaria
ROWID implícito Soporta varios índices	No tiene ROWID Soporta índices secundarios
Full Scan devuelve las tuplas desordenadas	Una recuperación completa devuelve tuplas orden. Según CP
Restricciones Unique permitidas	No soporta restricciones Unique
Soporta distribución, replicación y particionado	No soporta distribución, replicación ni particionado

### Índice por Clave Invertida (reverse index)

Invierte los datos del valor de la clave. Este índice es adecuado para búsquedas basadas en predicados “=”. Con este índice se reducen los embotellamientos en el índice cuando se están introduciendo los datos de forma ascendente para los valores de la clave, puesto que todos irían a la misma entrada del índice.

## 13. Índices BITMAP

Para cada valor que toma la clave almacena una secuencia de bits (tantos como tuplas contenga la tabla), el bit 1 indica que ese valor está presente en la tupla, el 0 que no lo está.

B-tree	BITMAP
Adecuado para columnas que tomen muchos valores	Adecuado para columnas que tomen pocos valores
Actualizaciones sobre las claves no muy costosas	Actualizaciones sobre las claves muy costosas
Ineficiente para consultas usando predicados OR	Eficiente para consultas usando predicados OR
Útil para OLTP	Útil para DSS

## 14. Acceso directo

Otra forma de acceder a un registro almacenado:

- No hay estructura adicional.
- Se usa un algoritmo que nos indique directamente la posición del registro deseado.

El **acceso directo** es calcular directamente la dirección de un registro mediante la aplicación, a un campo determinado del mismo, de algún algoritmo o función. El campo debe identificar unívocamente al registro.

Normalmente no es posible establecer una clave física que sea totalmente correlativa y única para cada registro. Hay que buscar un algoritmo que transforme los valores de un cierto campo en una dirección. Posee una entrada (campo clave) y una salida (valor entero positivo fácilmente transformable en RID).

Los algoritmos de direccionamiento no suelen mantener el orden de la clave. Los registros no están almacenados según el orden de su clave física. Problemas con la recuperación por intervalos.

Hay una gran variedad de algoritmos:

- Depende del tipo de clave:
  - Si la clave es alfanumérica, hay que transformarla a un valor numérico.
- Suelen estar basados en un mecanismo de generación de números pseudoaleatorios:
  - Cuadrados centrales:** se eleva la clave al cuadrado y se eligen tantos dígitos centrales como sea necesario.
  - Congruencias:** se divide la clave por M (suele ser primo) y se toma el resto.
  - Desplazamiento:** se superponen adecuadamente los dígitos binarios de la clave y luego se suman.
  - Conversión de base:** se cambia base de numeración y se suprimen algunos dígitos resultantes.

Salvo que el campo clave se diseñe para ello, es prácticamente imposible encontrar una transformación que dé un valor entero positivo en un rango de valores limitado tal que no haya dos valores distintos de clave que den lugar al mismo número (colisiones).

Los algoritmos producen huecos, zonas vacías del rango de salida, no asignadas por el algoritmo. Se traducen en huecos en el fichero de datos.

Para gestionar colisiones y huecos se combina el acceso directo con una gestión mediante listas de colisiones:

- Zona de desbordamiento.
- Colisión:
  - El registro problemático se almacena en la zona de desbordamiento.
  - Los sinónimos (registros con claves que producen colisión) se conectan mediante una lista.

Si crecen las listas de sinónimos el acceso directo puro no resulta adecuado.

- Mantener listas.
- Zona de desbordamiento casi como el fichero original.

Han aparecido técnicas más sofisticadas como el **hashing**.

## **15. Hashing básico**

Si el problema principal es que los valores de las claves no están uniformemente distribuidos en el intervalo  $[0, M]$ , se acumulan en una parte de este intervalo. La solución es asignar más espacio a esa parte del intervalo.

La técnica empleada es:

- Se divide el espacio del fichero en “cubos”.
- El algoritmo de direccionamiento asigna cubos, no direcciones concretas.
- En cada cubo puede haber más de un registro.
- Ciertos rangos de valores tienen asignados más cubos que otros.
- Se complementa con el uso de cubos de desbordamiento.

Los parámetros empleados son:

- Número de cubos.
- Tamaño de los cubos.
- La transformada clave/dirección, que debe tener en cuenta la distribución de la clave según rangos para que unos cubos no se llenen mucho y otros se queden muy vacíos.

Para insertar un registro:

- Transformar la clave.
- Localizar el cubo correspondiente.
- Si hay sitio se inserta el registro y se termina.
- Si no hay sitio, se sitúa el registro en un cubo de desbordamiento conectándolo con el cubo que realmente le corresponde mediante punteros.

El proceso de búsqueda:

- Transformar la clave.
- Localizar el cubo correspondiente.
- Realizar una búsqueda secuencial dentro del cubo:
  - Si hemos encontrado el registro, el proceso termina.
  - En caso contrario, se impone un barrido por punteros a través de los cubos de desbordamiento.

## **16. Hashing dinámico**

El hashing básico sigue teniendo problemas. Es necesario conocer la distribución previa de las claves para asignar adecuadamente los cubos. En otro caso siguen apareciendo huecos/colisiones. Al aumentar el número de registros, aumentan los registros en páginas de desbordamiento. Se hacen necesarias las reorganizaciones. La solución a esto es trabajar de forma dinámica.

Se parte de una configuración uniforme y de pocos cubos. Los restantes se van generando conforme se necesiten. Se asignan a los rangos conforme la afluencia de registros lo demanda.

## Técnica

- El valor transformado del campo clave nos lleva a la entrada de una tabla índice que se almacena en memoria.
- Allí está la dirección del cubo donde se encuentran los registros que tienen asociado este valor transformado.
- Puede ocurrir que varias entradas de la tabla conduzcan al mismo cubo.
- El proceso que sigue:
  - Inicialmente, todas las entradas apuntan al mismo cubo.
  - A medida que vamos insertando registros, se van generando nuevos cubos y cambiando las salidas de la tabla índice.

## Algoritmo de Hashing Dinámico

Datos de partida:

- $k$  = clave física para direccionar.
- $k' = h(k)$  valor entero entre 0 y  $M$ .
- $n$  = número de bits que tiene  $k'$  en binario.
- $d \leq n$ , los  $d$  primeros dígitos de  $k'$  seleccionan el cubo donde está el registro y se llaman pseudollave.
- $d < d \leq n$ , inicialmente el archivo tiene  $2^b$  cubos distintos, como máximo tendrá  $2^d$ .

## Algoritmo

- Se considera una tabla índice en memoria con  $2^d$  filas.
- En la primera columna de esta tabla se sitúan todas las posibles sucesiones de  $d$  dígitos binarios ( $d$  es la profundidad global de la tabla).
- En principio, todas las entradas cuyos  $b$  primeros dígitos son iguales apuntan al mismo cubo. Allí se almacenan los registros cuyo valor de  $k'$  tiene esos  $b$  primeros dígitos.
- Todos los cubos tienen en principio profundidad local igual a  $b$ .
- Cuando se llena un cubo se divide en 2, poniendo en uno de ellos los registros con el dígito  $b+1$  de  $k'$  a 0 y en el otro los que lo tienen igual a 1. La profundidad local de estos cubos aumenta una unidad.

El hashing dinámico supera los problemas clásicos del acceso directo. También tiene sus inconvenientes, pues utiliza una tabla índice adicional. El tamaño máximo de la tabla depende de “ $n$ ” y, por tanto, de la función de dispersión que se elija.