

## Tarea 3: Índices

Carlos Tonatihu Barrera Pérez  
Profesor: Hernández Contreras Euler  
Bases de Datos  
Grupo: 2CM1

31 de marzo de 2017

# Índice

<b>1. Introducción</b>	<b>1</b>
<b>2. Desarrollo</b>	<b>2</b>
2.1. Primario . . . . .	2
2.2. Secundario . . . . .	2
2.3. Agrupamiento . . . . .	3
2.4. Árbol $B^+$ . . . . .	4
2.5. Múltiples niveles . . . . .	6
2.6. Tablas hash . . . . .	6
<b>3. Conclusiones</b>	<b>8</b>
<b>Referencias</b>	<b>8</b>

## 1. Introducción

Los Índices surgen debido a que al hacer muchas consultas estas pueden hacer referencia a una pequeña porción de los registros de un archivo por lo que se genera un gasto adicional en la búsqueda de estos registros por lo que al implementar índices esto se puede mejorar.

Los índices en las bases de datos cumplen la misma función que el índice de un libro el cual es evitar búsquedas innecesarias para encontrar la información que se desea de la manera más rápida posible.

En este caso al recuperar algún registro con base en su identificador el sistema de bases de datos buscaría en un índice para encontrar el bloque de disco donde se ubica el registro correspondiente para después extraer el bloque y en seguida obtener el registro que se busca.[1]

Existen dos tipos de índices en la base de datos:

- **Índices ordenados.** Basados en una disposición ordenada de los valores.
- **Índices asociativos.** Basados en una distribución uniforme de los valores a través de cajones (buckets). El valor de cada cajón esta determinado por una función de asociación (hash función).

No existe una técnica universal que se deba de utilizar sino que esto depende de la base de datos. Para realizar la eleccion indicada se debe de tomar en cuenta:

- Tipos de acceso
- Tiempos de acceso
- Tiempo de inserción
- Tiempo de borrado
- Espacio adicional requerido

## 2. Desarrollo

### 2.1. Primario

El índice primario es aquel en el cual el archivo que contiene los registros está ordenado secuencialmente y el índice cuya clave de búsqueda especifica el orden secuencial del archivo es el índice de agrupación (índice primario) además de que es un índice no denso. Índice primario hace alusión a un índice según la clave primaria aunque esto no es necesario.

Barcelona	C-217	150.000	
Daimiel	C-101	100.000	
Daimiel	C-110	120.000	
Madrid	C-215	140.000	
Pamplona	C-102	80.000	
Pamplona	C-201	180.000	
Pamplona	C-218	140.000	
Reus	C-222	140.000	
Ronda	C-305	70.000	




Figura 1: Ejemplo de un índice primario ordenado secuencialmente con base en su clave.

Este tipo de índices está disponible en sistemas gestores de bases de datos como MySQL, SQLServer, PostgreSQL, Oracle y IBM DB2.

### 2.2. Secundario

El índice secundario es aquel en el que las claves de búsqueda especifican un orden diferente del orden secuencial del archivo. Este tipo de índice mejora el rendimiento de las consultas que utilizan otras claves de búsqueda distinta de la del índice primario. Sin embargo, estos implican un gasto adicional en la modificación de la base de datos.

Barcelona		→	C-217	Barcelona	750	
Daimiel		→	C-101	Daimiel	500	
Madrid		→	C-110	Daimiel	600	
Pamplona		→	C-215	Madrid	700	
Reus		→	C-102	Pamplona	400	
Ronda		→	C-201	Pamplona	900	
		→	C-218	Pamplona	700	
		→	C-222	Reus	700	
		→	C-305	Ronda	350	

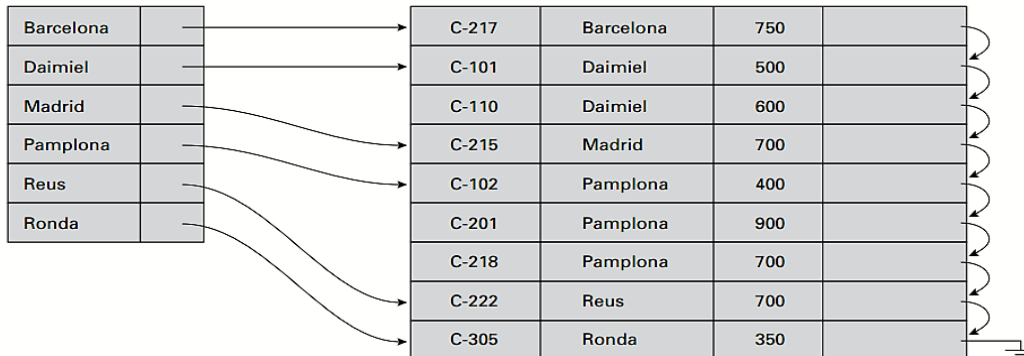


Figura 2: Índice secundario sobre el campo clave (denso).

En el caso de los índices secundarios sobre un campo no clave se tienen las siguientes opciones.

- Existe una entrada por cada registro (índice denso).
- Registros de longitud variable. Índice no denso y el campo de la dirección contiene una lista de punteros.
- O registros de longitud fija. Índice no denso con un nivel extra de indirección para manejar punteros múltiples

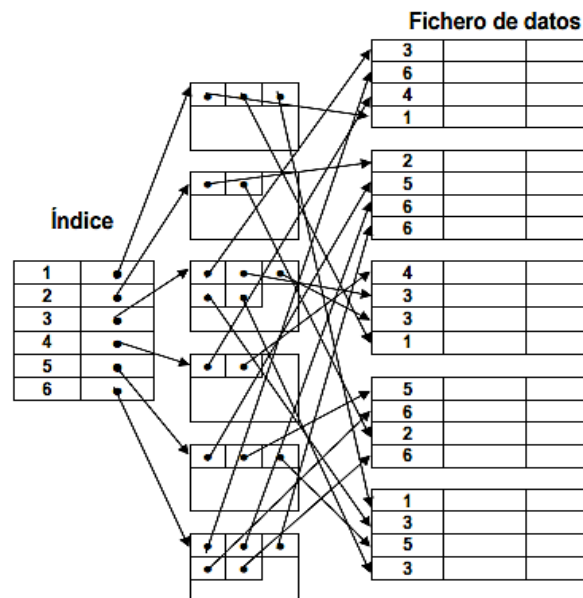


Figura 3: Índice secundario sobre un campo no clave.

Este tipo de índices está disponible en sistemas gestores de bases de datos como MySQL, PostgreSQL, Oracle y IBM DB2.

## 2.3. Agrupamiento

Un índice de agrupamiento es un índice no denso que tiene una entrada por cada valor distinto del campo de indexación. Sus entradas son registros de longitud fija, el campo de indexación que utilizan es un campo no clave de ordenación del fichero de datos, además, es un índice no denso. Es importante mencionar que en un fichero ordenado por un campo no clave solo se puede definir un índice de agrupamiento.

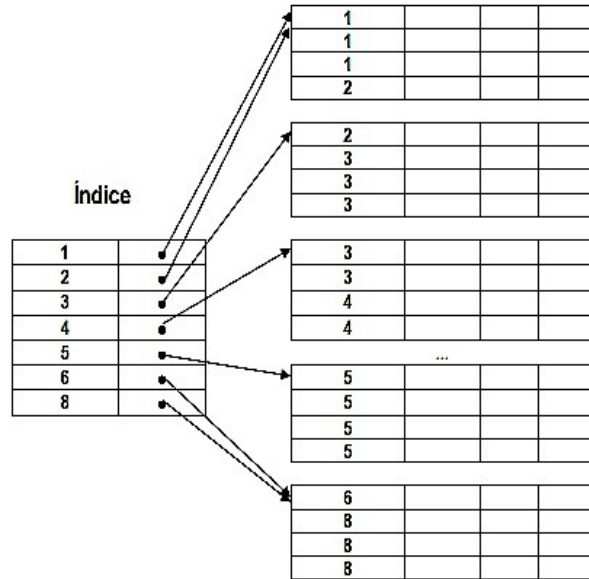


Figura 4: Ejemplo de un índice de agrupamiento.

Otro aspecto importante de este tipo de índices es que son registros de longitud fija en donde las entradas son una por cada valor distinto del campo de agrupamiento. Y el puntero apunta al primer bloque que contiene un registro con dicho valor.

## 2.4. Árbol $B^+$

Un índice de árbol  $B^+$  tiene forma de un árbol equilibrado en el que cada camino de la raíz a las hojas tiene la misma longitud. Su altura es proporcional a el logaritmo base N del número de registros de la relación.

Este tipo de árbol es más corto que los arboles binarios por lo que se necesitan menos accesos a disco para localizar los registros. Es por esto que las búsquedas son directas y eficientes pero al momento de realizar una inserción y borrado el trabajo se complica pero sigue manteniendo su eficiencia.

Se utiliza para evitar el principal inconveniente de la organización del archivo secuencial indexado el cual es que el rendimiento disminuye conforme el archivo crece.

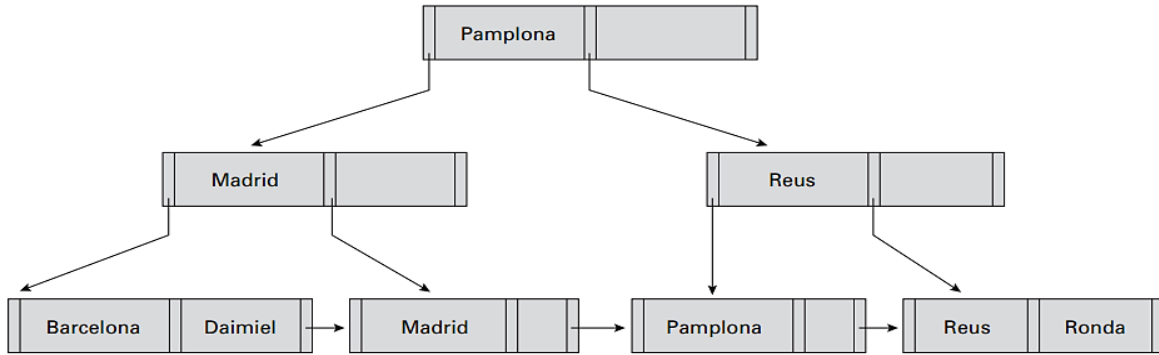


Figura 5: Estructura de un árbol  $n=3$ .

En un nodo de este tipo se tiene que  $K_i$  representa los valores de la clave de búsqueda y  $P_i$  son los punteros a los hijos (si se trata de un simple nodo) o a los registros para los nodos hoja. Como se puede observar en la figura 6 en un nodo las claves de búsqueda están ordenadas de la forma  $k_1 < k_2 < k_3 < \dots < k_{n-1}$ .

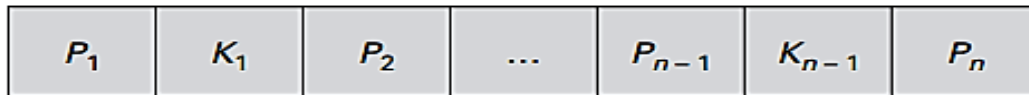


Figura 6: Estructura de un nodo del árbol  $n=3$ .

Este tipo de índices está disponible en sistemas gestores de bases de datos como MySQL, PostgreSQL, SQLServer, Oracle y IBM DB2.

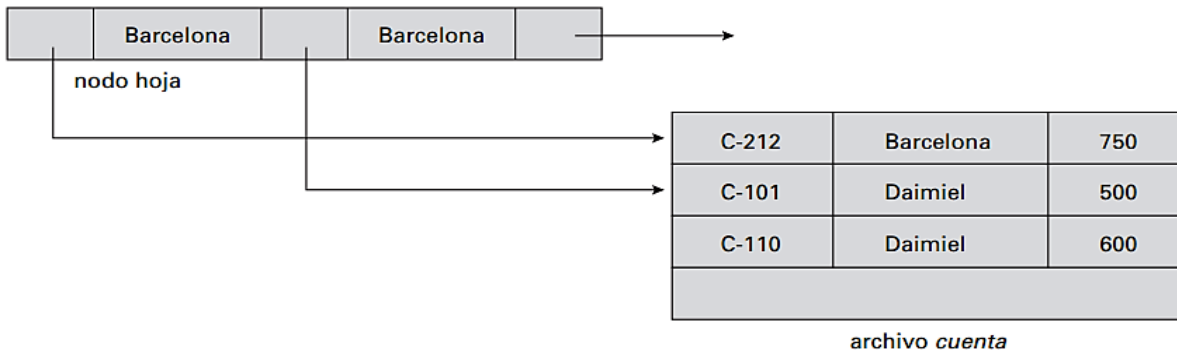


Figura 7: Ejemplo de un nodo hoja que apunta a un registro.

## 2.5. Múltiples niveles

El objetivo de los índices multinivel es reducir más que en la búsqueda binaria el fragmento de índice en donde se busca. Esto se realiza gracias a su estructura en donde el primer nivel es un fichero ordenado con entradas de tamaño fijo y un valor distinto del campo de indexación en cada una.

En los siguientes niveles hay índices primarios sobre el primer nivel esto quiere decir que para un número de registros por bloque  $r$  en el primer bloque se tienen  $i_1$  entradas, en el siguiente nivel hay  $i_2 = \frac{i_1}{r}$  entradas y en el tercer nivel  $i_3 = \frac{i_2}{r}$  por lo que se necesita un nivel más si el anterior ocupa más de un bloque.

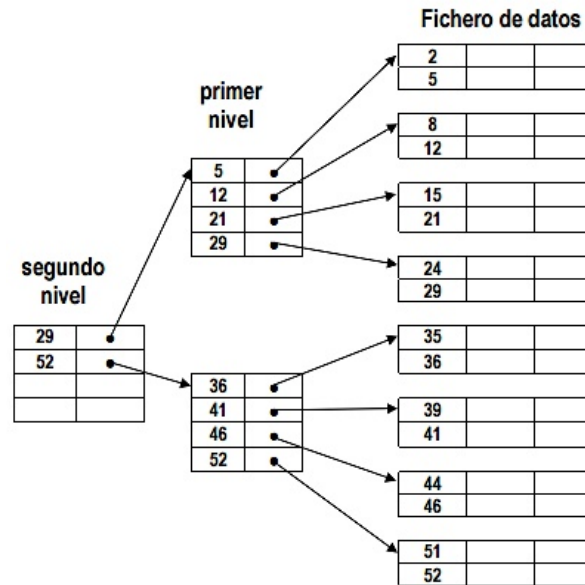


Figura 8: Estructura de un índice multinivel de dos niveles.

## 2.6. Tablas hash

Un índice de tabla hash consta de una colección de cubos organizados en una matriz. Además, una función hash asigna las claves de índice a los cubos correspondientes en el índice hash. Como se muestra en el ejemplo de la figura 9 las tres claves de índice se asignan a tres cubos distintos en el índice hash. [2]

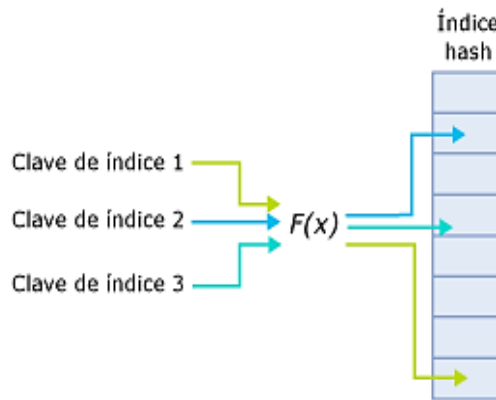


Figura 9: Asignación mediante tabla hash.

La función hash debe de tener las siguientes características:

- La función hash es determinista. La misma clave de índice se asigna siempre al mismo cubo en el índice hash.
- Se pueden asignar múltiples claves de índice al mismo depósito de hash.
- La función hash está equilibrada, lo que significa que la distribución de los valores de clave de índice en los depósitos de hash sigue normalmente una distribución de Poisson

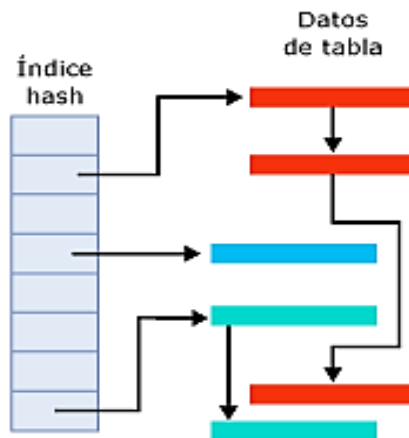


Figura 10: Tabla hash en memoria.

En memoria la estructura de índice hash consta de una matriz de punteros de memoria. Cada cubo asigna un desplazamiento en esta matriz. Cada cubo de la matriz señala a la primera fila del cubo de hash. Cada fila del cubo señala a la siguiente fila, por lo que genera una cadena de filas para cada cubo de hash como se observa en la figura 10.

Este tipo de índices está disponible en sistemas gestores de bases de datos como MySQL, PostgreSQL, SQL Server, Oracle y IBM DB2.



### 3. Conclusiones

Este tema es bastante extenso por lo que se debe de conocer a fondo para elegir la técnica que mejores resultados nos brinde a la hora de implementar una base de datos, además de que se tiene que elegir la tecnología correcta para implementar dicha técnica. Además, se puede observar que la mayoría de este tipo de índices se puede utilizar en diversos sistemas gestores de bases de datos. Por ultimo, para algunas de estas técnicas considero que se requiere tener conocimientos de algoritmos y estructuras de datos para obtener resultados más óptimos y entender que esta sucediendo.

### Referencias

- [1] A. Silberschatz, *Fundamentos de Diseño de Bases de Datos*. McGraw-Hill, 2007.
- [2] MSDN, “Índices hash.” [https://msdn.microsoft.com/es-es/library/dn133190\(v=sql.120\).aspx](https://msdn.microsoft.com/es-es/library/dn133190(v=sql.120).aspx), 2013. [Consultado: 2017-03-31].