Instituto Politécnico Nacional

Escuela Superior de Cómputo

Bases de Datos

Tarea no. 3:

Índices de acceso (index)

Profesor: Euler Hernández Contreras

Alumno: Calva Hernández José Manuel

Grupo: 2CM12

Índice

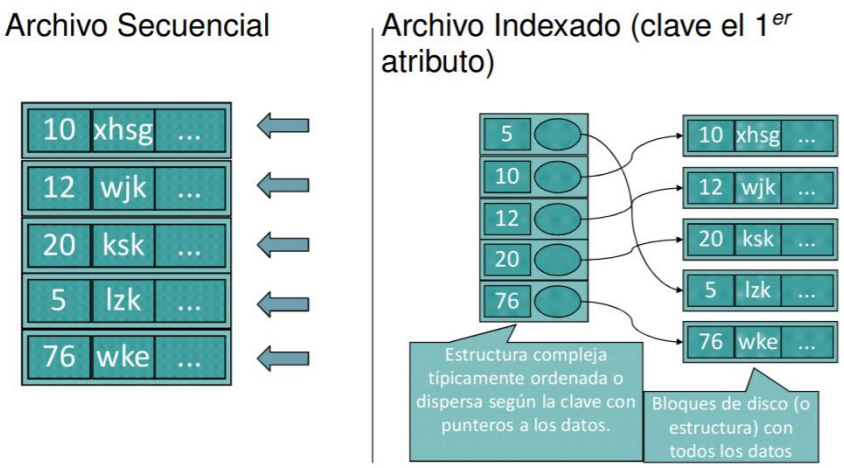
Introducción

Se conoce como índice, o index por su término en inglés, a una estructura de datos auxiliar utilizada para acelerar la recuperación de registros en respuesta a ciertas condiciones de búsqueda. Las estructuras de índice normalmente proporcionan rutas de acceso, que ofrecen formas alternativas de acceder a los registros sin que se vea afectada la ubicación física de los registros en el disco. Permiten un acceso eficaz a los registros basándose en la indexación de los campos que se utilizan para construir el índice. Básicamente, es posible utilizar cualquier campo del fichero para crear un índice y se pueden construir varios índices con diferentes campos en el mismo fichero. También son posibles varios tipos de índices; cada uno de ellos utiliza una estructura de datos en particular para acelerar la búsqueda. Para encontrar uno o varios registros del fichero basándose en ciertos criterios de selección de un campo indexado, primero hay que acceder al índice, que apunta a uno o más bloques del fichero donde están almacenados los registros requeridos. Los tipos de índices predominantes están basados en los ficheros ordenados (índices de un nivel) y en estructuras de datos en forma de árbol (índices multinivel, árboles B+). Los índices también pueden construirse basándose en la dispersión o en otras estructuras de datos de búsqueda.

Comúnmente el fichero de índice es mucho más pequeño que el fichero de datos, por lo que la búsqueda en el índice mediante una búsqueda binaria es razonablemente eficaz.

Indexación

La indexación es la principal herramienta para optimizar el rendimiento general de cualquier base de datos. Es una técnica para recuperar los datos contenidos en un fichero o una zona de memoria por medio de un índice que guarda la posición de los datos. Además, es muy común en bases de datos, ya que de no ser usada, es notable un bajo rendimiento en los servidores.



Índice Principal

Concepto

Un índice principal o primario se especifica en el campo clave de ordenación de un fichero ordenado de registros. Tiene entradas con registros de longitud fija con dos campos, esto quiere decir que sobre un fichero ordenado por clave, sólo puede definirse un índice primario.

Es un índice no denso, es decir, sólo tiene entradas para algunos de los valores de búsqueda, esto porque incluye una entrada por cada bloque de disco del fichero de datos y las claves de su registro ancla, en lugar de una entrada por cada valor de búsqueda (es decir, por cada registro).

Su principal ventaja es que debido a su estructura se facilita la búsqueda binaria sobre el índice, ya que se requiere de menos visitas sobre los bloques de disco.

Su mayor problema es el mismo hecho de ser un fichero ordenado, ya que la inserción y eliminación de registros van íntimamente ligados al índice principal. Esto se traduce en que tendremos que mover todos los registros cuando queramos añadir uno nuevo, además, ocasionalmente tendremos que cambiar algunas entradas del índice, puesto que al mover los registros modificaremos los registros ancla de algunos bloques.

Estructura

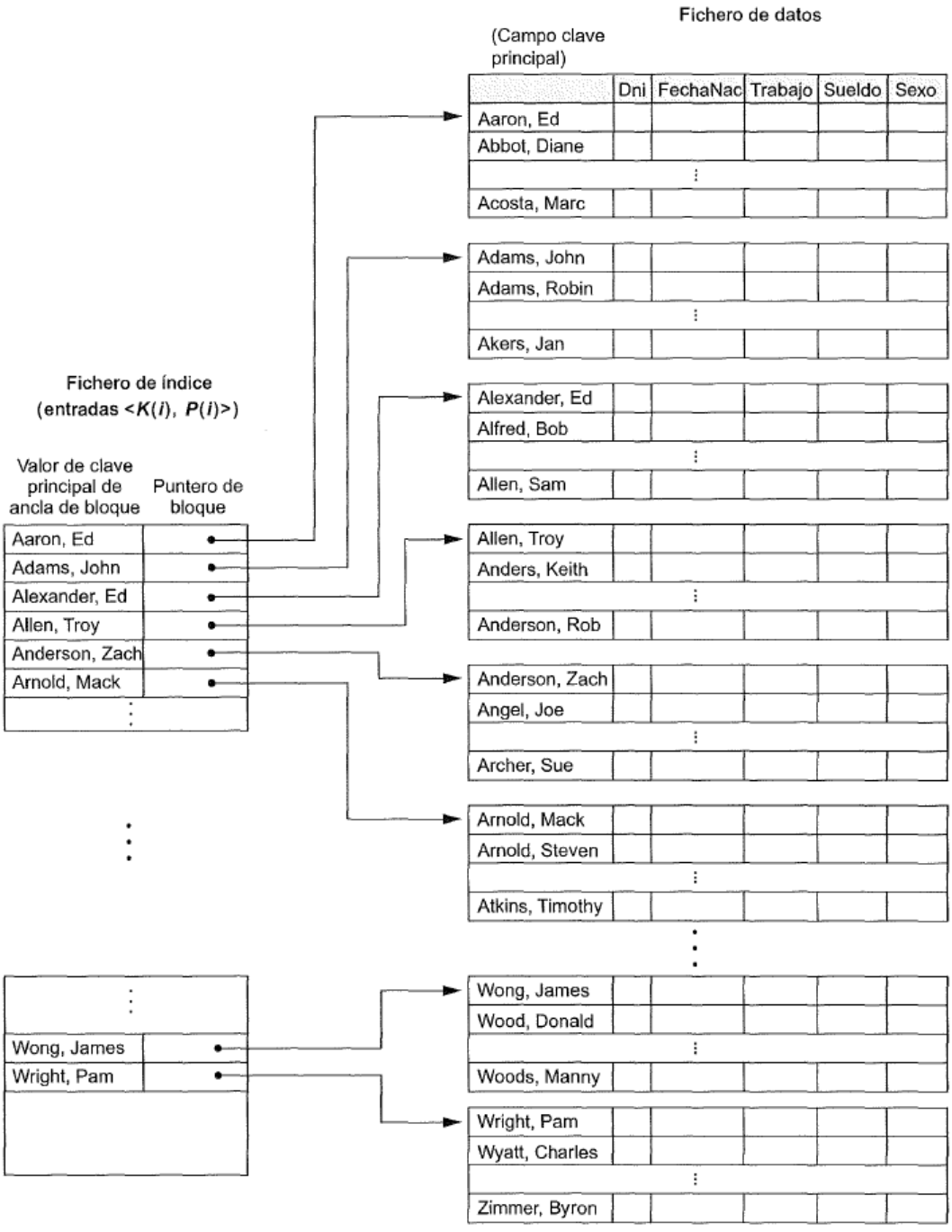
Los archivos indexados se componen normalmente de una dupla compuesta por:

- Clave de ordenación (search key): Es un atributo utilizado para buscar registros en un archivo.
- Archivo índice: Consiste en registros, también llamados entradas de índice (index entries), de la forma

Clave de ordenación	Puntero

El fichero de Índice para un índice principal necesita menos bloques que el fichero de datos, por dos razones. En primer lugar, hay menos entradas de índice que registros en el fichero de datos. En segundo lugar, cada entrada del índice tiene normalmente un tamai'ío más pequei'io que un registro de datos, porque sólo tiene dos campos; en consecuencia, en un bloque entran más entradas de índice que registros de datos.

Ejemplos


Fichero de datos
(Campo clave principal)
Dni
FechaNac
Trabajo
Sueldo
Sexo
Aaron, Ed
Abbot, Diane
Acosta, Marc
Adams, John
Adams, Robin
Akers, Jan
Alexander, Ed
Alfred, Bob
Allen, Sam
Allen, Troy
Anders, Keith
Anderson, Rob
Anderson, Zach
Angel, Joe
Archer, Sue
Arnold, Mack
Arnold, Steven
Atkins, Timothy
Wong, James
Wood, Donald
Woods, Manny
Wright, Pam
Wyatt, Charles
Zimmer, Byron
Fichero de índice
(entradas <K(i), P(i)>)
Valor de clave principal de ancla de bloque
Puntero de bloque
Aaron, Ed
Adams, John
Alexander, Ed
Allen, Troy
Anderson, Zach
Arnold, Mack
Wong, James
Wright, Pam

Índice Secundario

Concepto

Puede ser que nos interese tener un índice para claves que no sean primarias, o sea una clave para más de un registro, éstos son los llamados índices secundarios. Su principal característica es que, al contrario que en los primarios donde el direccionamiento pudiera ser real (posición exacta en el disco) o relativo (en función de la posición del fichero), en los secundarios se emplea el direccionamiento simbólico (la clave proporciona la clave primaria del registro, y no su dirección ni física, ni relativa, y el sistema emplea la clave primaria para localizar ese registro), en definitiva, emplea punteros indirectos.

La ventaja de este direccionamiento es que podemos hacer muchos índices secundarios y a la hora de modificar los ficheros, las direcciones físicas cambian, con lo que se deben cambiar también los índices primarios (actualizarlos); esta operación puede llevar mucho tiempo, sin embargo, al usar direccionamiento simbólico no es necesario modificar los índices, puesto que no tienen punteros a ningún sitio.

Hay una entrada de Índice por cada registro del fichero de datos, que contiene el valor de la clave secundaria para el registro y un puntero al bloque en el que está almacenado el registro o al propio registro. Por tanto, dicho Índice es denso, y estos proporcionan un ordenamiento lógico de los registros según el campo de indexación.

Cabe recalcar que existe la posibilidad de que sea un campo no denso, si el índice secundario se impone sobre un campo no clase, entonces podemos tener:

- Una entrada por cada registro: Índice denso
- Registro de longitud variable: Índice no denso y el campo de la dirección contiene una lista de punteros.
- Registro de longitud fija: Índice no denso con un nivel extra de dirección para manejar punteros múltiples.

Estructura

Los ficheros en los que se usa este tipo de direccionamiento comparten una estructura similar a esta:

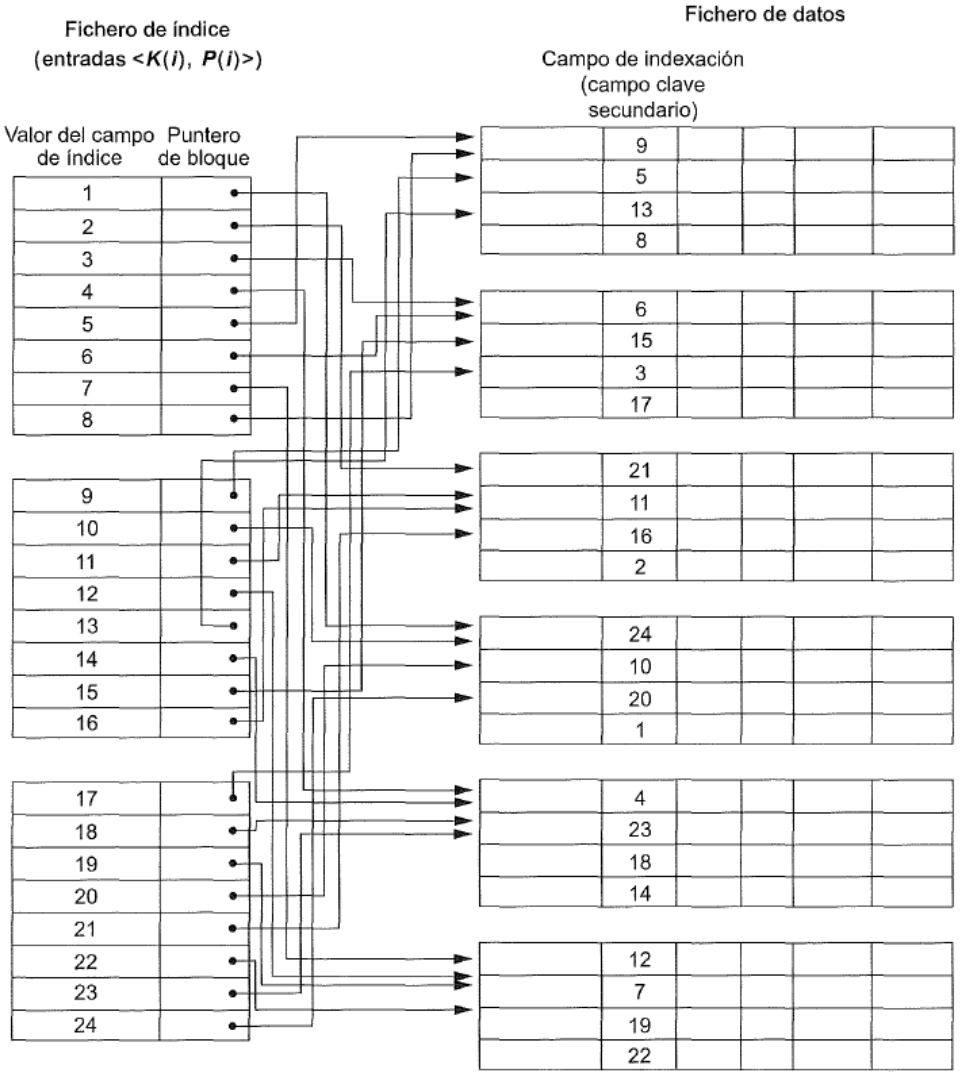
- Reciben el nombre de ficheros invertidos.
- A partir de un dato (clave secundaria), se obtiene una clave primaria que lleva a más datos. Éste recibe el nombre de campo de indexación y puede ser cualquiera que no sea el campo de ordenación
- Existen dos tipos de ficheros invertidos: Los ficheros totalmente invertidos y los ficheros parcialmente invertidos
- En los ficheros totalmente invertidos de una clave secundaria se obtienen todas las primarias relacionadas.

k^2	k^1
Albacete	1
	...
	...
	97
Almería	98
	...
	...
	156

k^1 = Clave primaria
k^2 = Clave secundaria

Es importante recalcar que pueden definirse varios índices secundarios sobre un mismo fichero a manera de reducir la posibilidad de colisiones en la clave primaria.

Ejemplos



Índice de Agrupamiento

Concepto

Son registros de longitud fija donde su campo de indexación es un campo no clave de ordenación del fichero de datos (campo de agrupamiento), es decir, éste no tiene un valor distinto para cada registro; para estos campos se puede crear el índice agrupado, con el fin de acelerar la recuperación de los registros que tienen el mismo valor en dicho campo.

Al ser similar al índice primario, en éste también podemos hacer una búsqueda binaria sobre el índice que nos facilitará la ubicación de un registro al visitar menos bloques de disco.

Sin embargo, también comparte sus desventajas, la inserción y el borrado de un registro todavía provoca problemas, porque los registros de datos están ordenados físicamente. Para aliviar el problema de la inserción, se suele reservar un bloque entero (o un grupo de bloques contiguos) para cada valor del campo agrupado; todos los registros con ese valor se colocan en el bloque (o grupo de bloques).

Un Índice agrupado es otro ejemplo de Índice no denso porque tiene una entrada por cada valor distinto del campo de indexación, que no es una clave por definición y, por tanto, tiene valores duplicados en lugar de un valor único por cada registro del fichero.

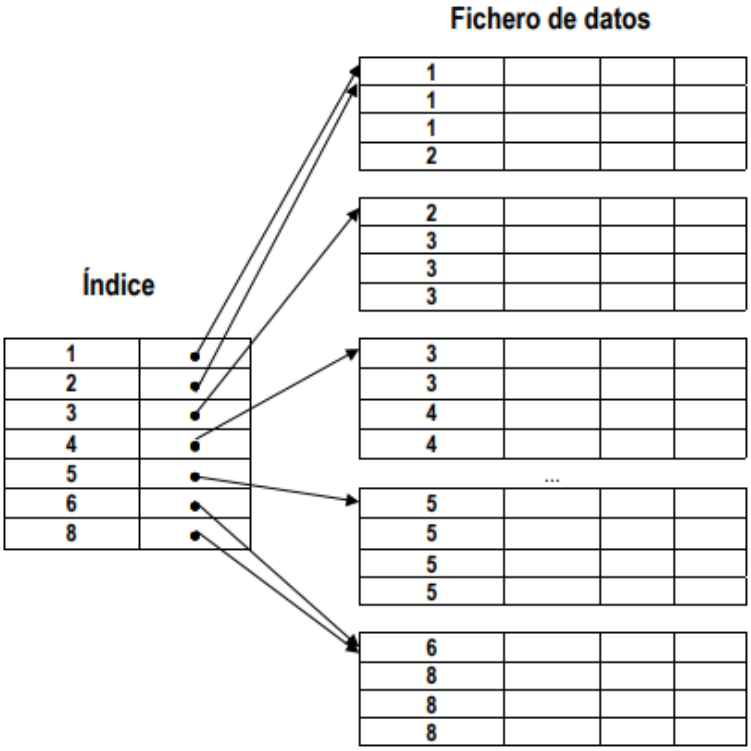
Estructura

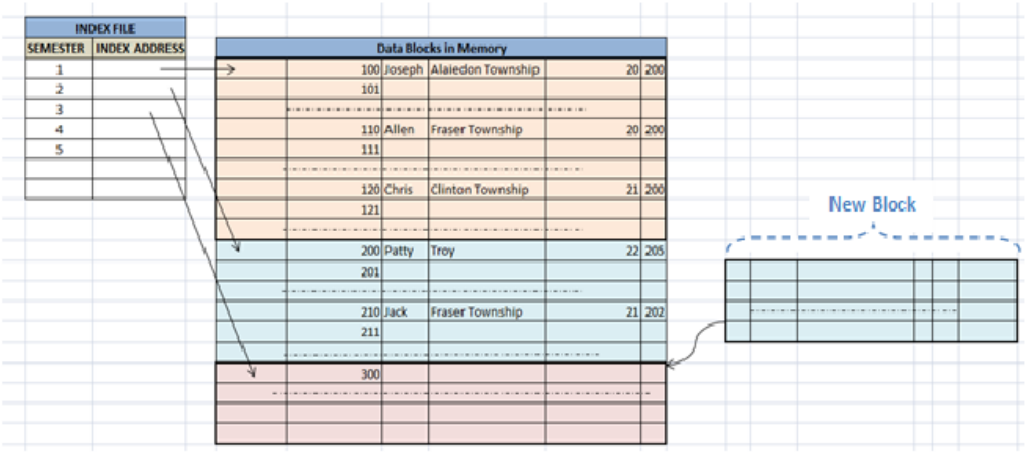
Un Índice agrupado también es un fichero ordenado con dos campos; el primero es del mismo tipo que el campo agrupado del fichero de datos, y el segundo es un puntero a un bloque. En el Índice agrupado hay una entrada por cada valor distinto del campo agrupado, que contiene el valor y un puntero al primer bloque del fichero de datos que tiene un registro con ese valor para su campo agrupado.

Así pues, la estructura de cada archivo indexado es similar a la del índice primario, mostrando la forma:

Clave de ordenación	Puntero

Ejemplos


Fichero de datos
Índice
1
2
3
4
5
6
8
1 1 1 2
2 3 3 3
3 3 4 4
...
5 5 5 5
6 8 8 8


INDEX FILE
SEMESTER
INDEX ADDRESS
1
2
3
4
5
Data Blocks in Memory
100 Joseph Alaiedon Township 20 200
101
110 Allen Fraser Township 20 200
111
120 Chris Clinton Township 21 200
121
200 Patty Troy 22 205
201
210 Jack Fraser Township 21 202
211
300
New Block

Índice de Múltiples Niveles

Concepto

Una búsqueda binaria requiere aproximadamente ($\log_2 b_i$) accesos a bloques para un índice con b_i bloques, porque cada paso del algoritmo reduce por un factor de 2 la parte del fichero de Índice que continuamos explorando. Por eso, tomamos la función log en base 2. La idea que hay detrás de un índice multinivel es reducir la palie del Índice que continuamos explorando por bfr_i, el factor de bloqueo para el Índice, que es mayor que 2. Por tanto, el espacio de búsqueda se reduce mucho más rápidamente, El valor bfr_i se denomina fan-out del Índice multinivel, al que nos referiremos mediante el símbolo f_0. La búsqueda en un índice multinivel requiere aproximadamente ($\log_{f0} b_i$) accesos a bloques, que es un número más pequeño que para la búsqueda binaria si el fan-out es mayor que 2.

El esquema multinivel aquí descrito puede utilizarse en cualquier tipo de Índice, sea principal, agrupado o secundario (siempre y cuando el índice de primer nivel tenga valores distintos para K(i) y entradas de longitud fija).

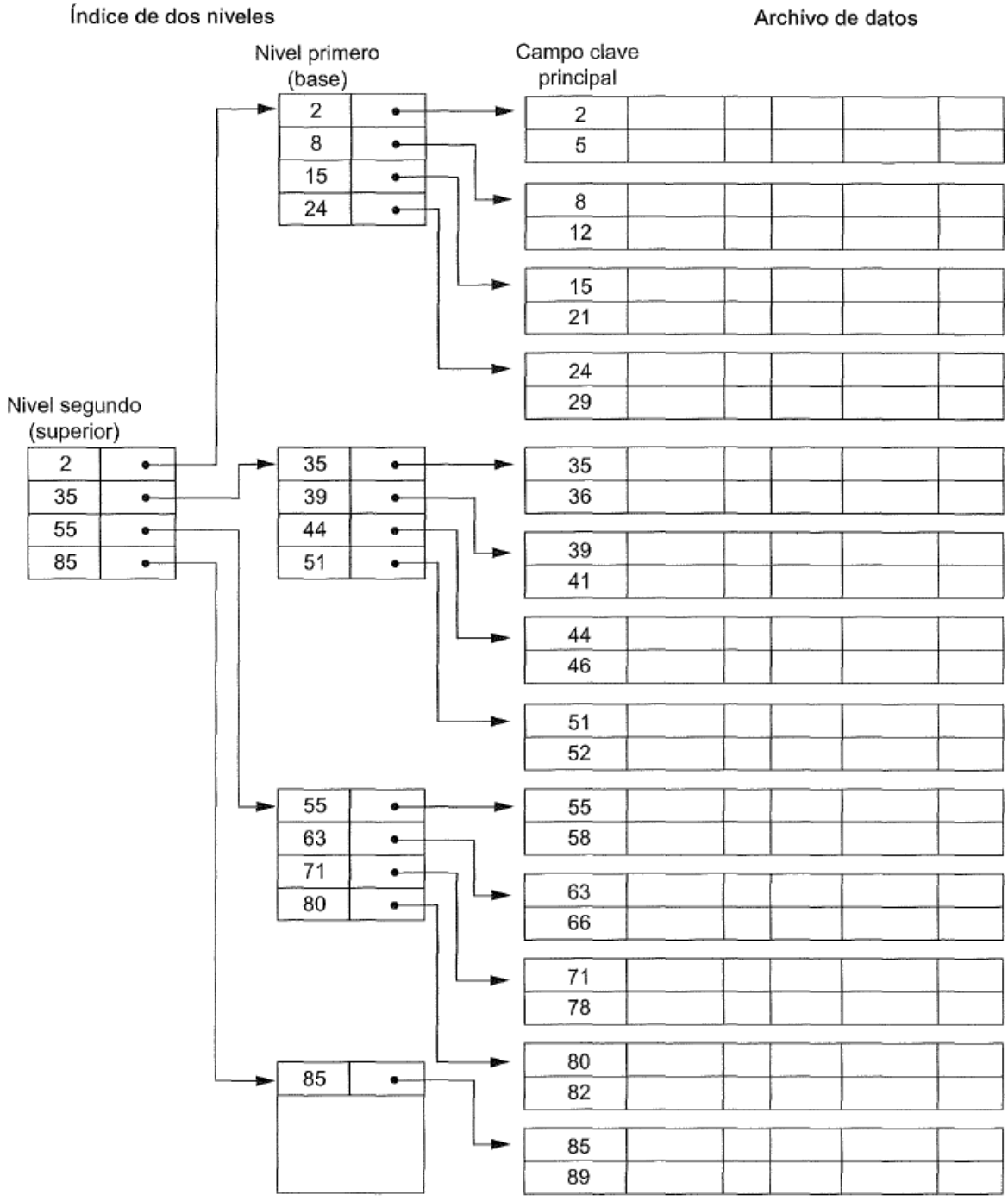
Este tipo de índice presente problemas de rendimiento, ya que al ser ficheros ordenados tiende a dificultar la inserción y eliminación, pero en este caso, se ve amplificado ya que hay múltiples capas de índices ordenados, lo que vuelve exponencial la reordenación en una situación no óptima. A su vez, requiere un mayor espacio en memoria al requerir un mayor número de índices para las referencias.

Estructura

Un índice multinivel considera el fichero de índice, al que ahora nos referiremos como primer nivel (o base) de un índice multinivel, como un fichero ordenado con un valor distinto por cada K(i). Por consiguiente, podemos crear un índice principal para el primer nivel; este índice al primer nivel se denomina segundo nivel del índice multinivel. Como el segundo nivel es un índice principal, podemos utilizar anclas de bloque de modo que el segundo nivel tenga una sola entrada por cada bloque del primer nivel. El factor de bloqueo bfr_i para el segundo nivel (y para todos los niveles subsiguientes) es el mismo que para el índice de primer nivel porque todas las entradas del índice tienen el mismo tamaño; cada una con un valor de campo y una dirección de bloque. Si el primer nivel tiene 1'1 entradas, y el factor de bloqueo (que también es el fan-out) para el Índice es $bfr_i = f_0$, entonces el primer nivel necesita $\frac{r_1}{f_0}$ bloques, que es por consiguiente el número de entradas r_2 necesarias en el segundo nivel del índice.

Este proceso se puede repetir subsecuentemente siguiendo la misma lógica del segundo nivel.

Ejemplos


Índice de dos niveles
Archivo de datos
Nivel primero
(base)
Campo clave
principal
Nivel segundo
(superior)
2
8
15
24
35
39
44
51
55
63
71
80
85
2
5
8
12
15
21
24
29
35
36
39
41
44
46
51
52
55
58
63
66
71
78
80
82
85
89

Índice Hash Tables

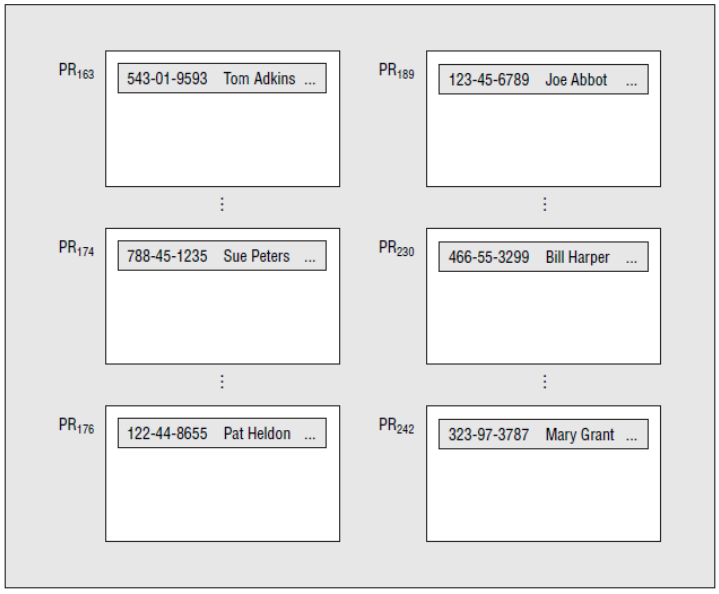
Concepto

Si el campo usa número no consecutivo o no es numérico es necesario convertir su valor en alguna forma. El esquema de conversión para valores numéricos se llama esquema de dispersión (hashing) y el campo sobre el que se realiza es el campo de claves hashing. Los valores no numéricos se convierten fácilmente en numéricos al usar algún tipo de código. Un gran problema, llamado colisión, ocurre cuando dos diferentes valores clave producen la misma dirección disponible. Entonces las claves se llaman sinónimas, y puede conllevar a un retraso debido a que a cada colisión le corresponderá un debido reordenamiento en la base, por ello se debe elegir cuidadosamente la función hash.

Hash se refiere a una función o método para generar claves o llaves que representen de manera casi unívoca a un documento, registro, etc. Una función hash es una función para resumir o identificar probabilísticamente un gran conjunto de información, dando como resultado un conjunto imagen finito generalmente menor. Esta función es determinista, ya que 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.

Otro problema que se deriva es la búsqueda secuencial, debido a la naturaleza del esquema de dispersión, podremos buscar registros durante un tiempo antes de encontrar la primer incidencia, esto debido a que la tabla Hash asigna el índice con base en la función, y no con base a una secuencia en el indexado.

StdSSN	StdSSN Mod 97	Número PR
122448655	26	176
123456789	39	189
323973787	92	242
466553299	80	230
788451235	24	174
543019593	13	163

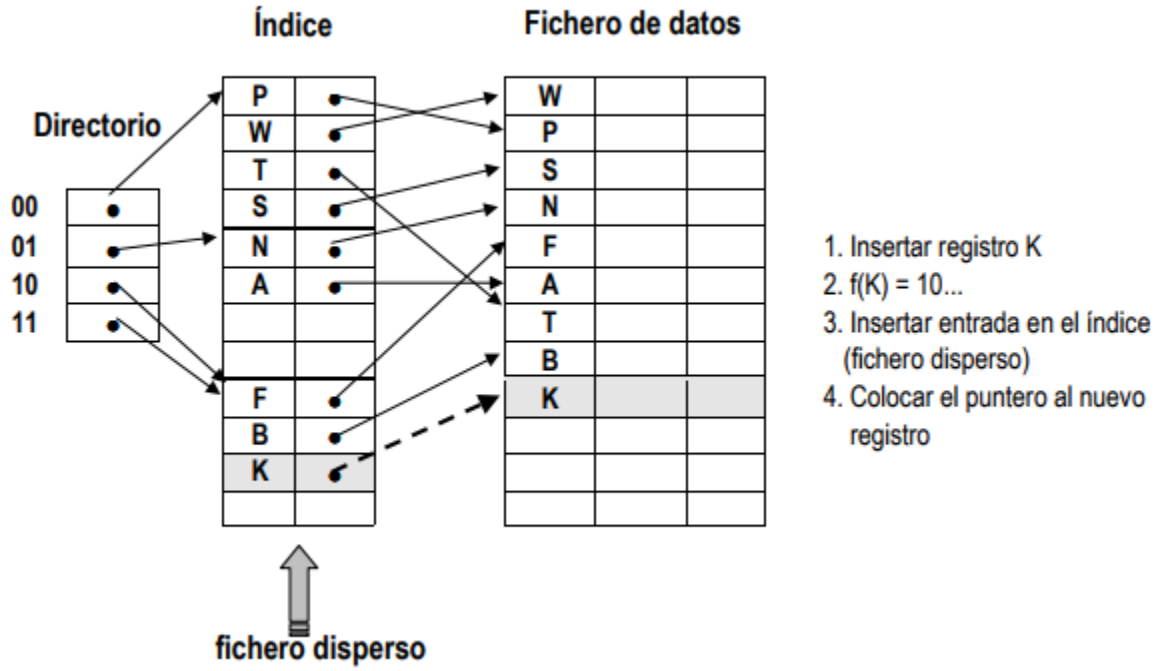


Estructura

Un índice hash consta de una colección de cubos organizados en una matriz. Una función hash asigna las claves de índice a los cubos correspondientes en el índice hash, y esta se organiza en memoria de tal forma que 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.

Los archivos indexados contienen un tupla de dos elementos, donde el primer elemento contiene el elemento a dispersar, o bien, la clave hash una vez aplicada la función; y por otra parte el puntero al directorio que contiene la información.

Ejemplos



Índice Árbol B+

Concepto

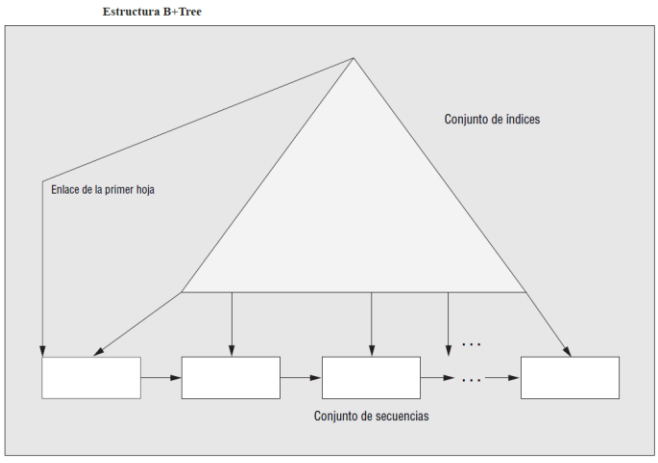
Los árboles se usan para contener y procesar varias estructuras de base de datos, pero se usan ampliamente para indexar archivos. Una estructura llamada árbol B+ se puede usar para almacenar un índice jerárquico eficiente y flexible que proporcione tanto acceso secuencial como directo a los registros. La mayoría de las implementaciones de un índice multinivel dinámico utilizan una variante de la estructura de datos en árbol B denominada árbol B+. En un árbol B, cada valor del campo de búsqueda aparece una vez en algún nivel del árbol, junto con un puntero de datos. En un árbol B + los punteros a datos se almacenan sólo en los nodos hoja del árbol, por lo cual, la estructura de los nodos hoja difiere de la de los nodos internos. Los nodos hoja tienen una entrada por cada valor del campo de indexación, junto con un puntero al registro (o al bloque que contiene ese registro) si el campo de búsqueda es un campo clave. En el caso de un campo de búsqueda que no es clave, el puntero apunta a un bloque que contiene punteros a los registros del fichero de datos, creándose un nivel extra de indirección.

Estructura

El índice consiste en dos partes, llamadas conjunto índice y conjunto secuencia. El conjunto secuencia está en el nivel inferior del índice (los nodos hoja) y consiste en todos los valores clave ordenados en secuencia con un puntero desde cada valor clave hasta su registro correspondiente en el archivo de datos.

No se muestran los registros de datos, que se pueden ordenar de manera aleatoria o en cualquier secuencia física deseada. Se supone que los registros de datos no están bloqueados y que cada puntero conduce hacia un solo registro. Sin embargo, los punteros pueden conducir a repositorios, espacios que suelen contener muchos registros, si se desea. También notará que el puntero de la extrema derecha de cada nodo hoja, el puntero horizontal, se usa para vincular el nodo con el siguiente en el conjunto secuencia. Esto permite usar el conjunto secuencia para acceso secuencial al archivo. Todo lo que necesita es comenzar en el nodo hoja de la extrema izquierda y ubicar cada registro desde dicha hoja a su vez, luego seguir los punteros horizontales para llegar al siguiente nodo del conjunto secuencia, y así por el estilo.

El acceso directo a los registros se logra con el uso del conjunto índice, desde el nodo raíz y siguiendo una ruta jerárquica estricta hacia el nodo apropiado en el conjunto secuencia.



Ejemplos

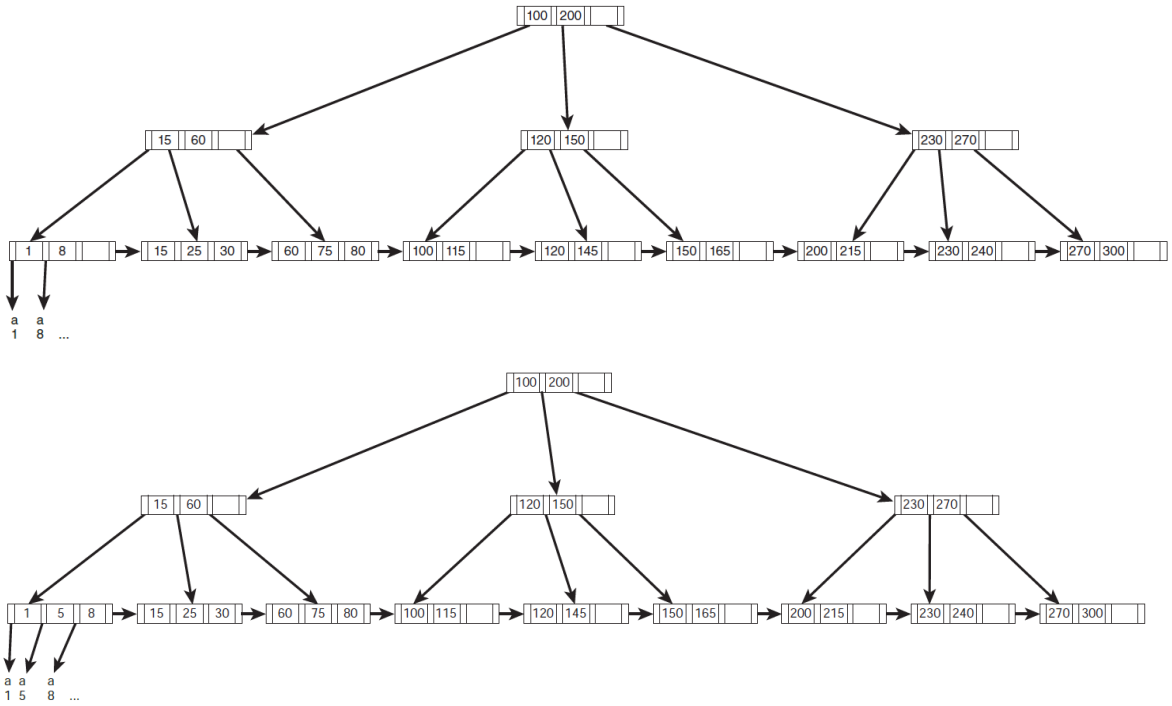


FIGURA A.14(a)

Inserción del valor clave 5

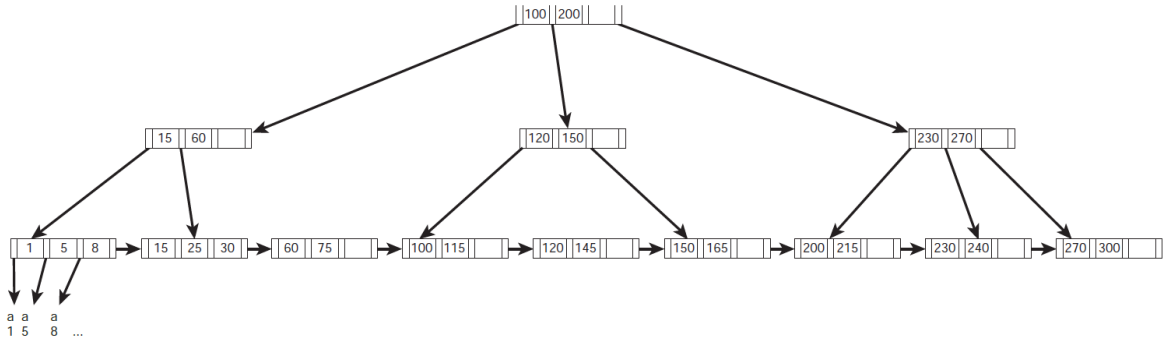


FIGURA A.14(b)

Índice después de insertar 5 y borrar 80

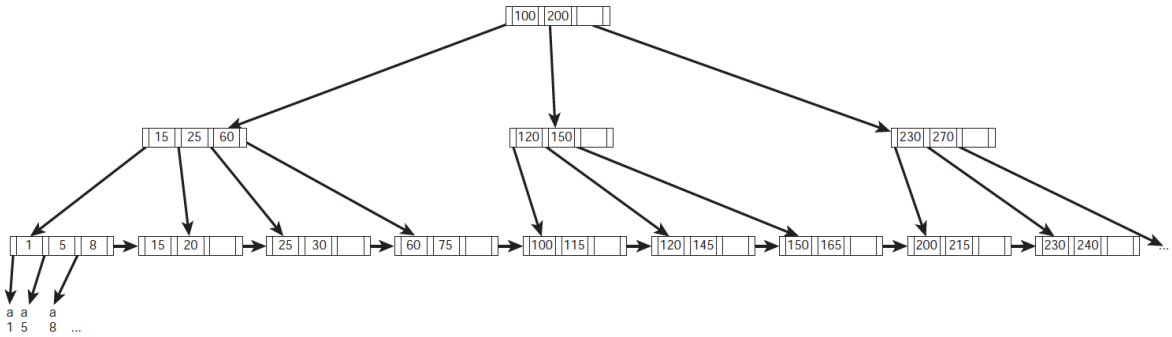


FIGURA A.14(c)

Índice después de insertar 5, borrar 80 e insertar 20

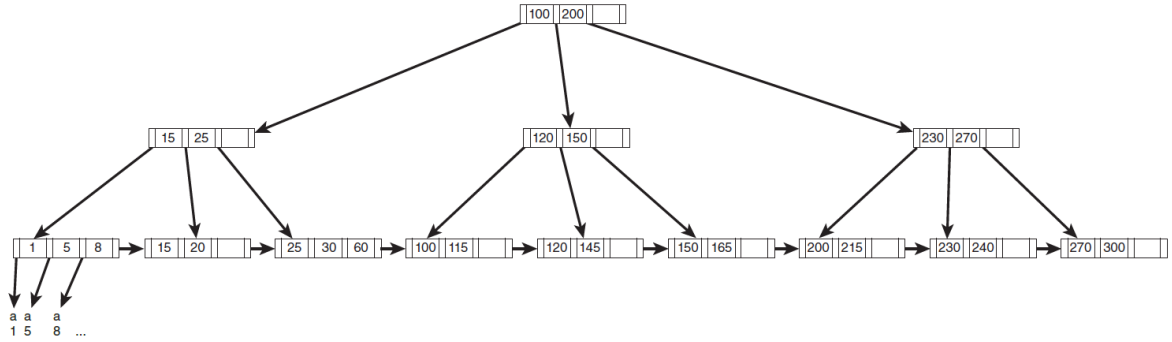


FIGURA A.14(d)

Índice después de insertar 5, borrar 80, e insertar 20 y borrar 75

SGBD y su implementación

	MySQL	Oracle	IBM DBZ	Postgres	SQL Server	Informix
Primario	×	×	×		×	
Secundario	×		×	×		×
Agrupamiento	×	×	×		×	×
Múltiples Niveles	×		×		×	
Hash Tables	×	×		×		×
Árboles B+	×	×		×		×

Referencias

Libros

Ramez, E., & Navathe, S. (2000). *Sistemas de Bases de Datos: Conceptos Fundamentales* (1st ed.). México: Pearson Educación.

Ricardo, C., Campos Olguín, V., & Enríquez Brito, J. (2010). *Bases de datos*. México: Mcgraw Hill/Interamericana Editores.

Mannino, M. (2007). *Administración de Bases de Datos* (3rd ed.). México: McGraw-Hill.

Páginas Web

5.5.1. Indexación | Manual Dataprix TI. (2017). *Dataprix.com.* Retrieved 12 October 2017, from http://www.dataprix.com/551-indexacion

Barzanallana, R. (2014). *Apuntes. Organizacion de ficheros. Sistemas de bases de datos. 2005/06-7 Informatica Aplicada a la Gestion Publica. Rafael Barzanallana.* Um.es. Retrieved 12 October 2017, from http://www.um.es/docencia/barzana/IAGP/Iagp7.html

Database index. (2017). *Wikipedia*. Retrieved 12 October 2017, from https://en.wikipedia.org/wiki/Database_index

Indexing in Databases | Set 1 - GeeksforGeeks. (2008). *GeeksforGeeks*. Retrieved 12 October 2017, from http://www.geeksforgeeks.org/indexing-in-databases-set-1/

Indexing in DBMS (2010). *Tutorialcup.com*. Retrieved 12 October 2017, from https://www.tutorialcup.com/dbms/indexing.htm

Access Index. (2012). *Universitat Jaume I*. Retrieved 14 October 2017, from http://www3.uji.es/~mmarques/f47/teoria/tema2b.pdf

Base de datos Indexada. (2015). *prezi.com*. Retrieved 14 October 2017, from https://prezi.com/jbwprnft0so7/base-de-datos-indexada/?webgl=0

Bases de Datos 2 - Teórico. (2016*). Facultad de Ingeniería - Universidad de Uruguay.* Retrieved 14 October 2017, from https://www.fing.edu.uy/tecnoinf/mvd/cursos/bd2/material/teo/bd2-teorico07.pdf

IBM Knowledge Center - home. (2017). *IBM*. Retrieved 14 October 2017, from https://www.ibm.com/support/knowledgecenter/