

# Indexación y *hashing*

Dr. José Luis Zechinelli Martini

[jose Luis.zechinelli@udlap.mx](mailto:jose Luis.zechinelli@udlap.mx)

LIS – 2061

*Equipo Tecnologías de Bases de Datos  
LAFMIA – UDLAP*

## Plan

- Conceptos básicos
- Indexación
- Árboles B+ y B
- Funciones de asociación (*hash*)
- Comparación entre indexación y asociación
- Acceso por claves múltiples

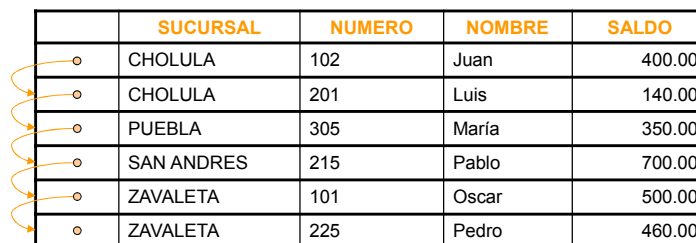
## Conceptos básicos (1)

Recuperar una pequeña parte de los registros de un archivo: **consultar**, **insertar**, **eliminar**.

- **Índices ordenados**. Estos índices están basados en una disposición ordenada de los valores.
- **Índices asociativos (*hash*)**. Estos índices están basados en una distribución uniforme de los valores a través de una serie de cubetas (*buckets*). Cada cubeta es determinada por una función.
- Los índices son **estructuras adicionales** que permiten acceder de manera rápida a la información almacenada en archivos.

## Conceptos básicos (2)

- **Clave de búsqueda**: atributo o conjunto de atributos para buscar los registros en un archivo.
- Se pueden tener más de un índice o función de asociación por archivo: **uno por cada atributo**.



	SUCURSAL	NUMERO	NOMBRE	SALDO
○	CHOLULA	102	Juan	400.00
○	CHOLULA	201	Luis	140.00
○	PUEBLA	305	María	350.00
○	SAN ANDRES	215	Pablo	700.00
○	ZAVALETA	101	Oscar	500.00
○	ZAVALETA	225	Pedro	460.00

## Evaluación de estructuras

- **Tiempo de acceso:** encontrar un dato (valor o rango) determinado.
- **Tiempo de inserción:** insertar el nuevo dato y actualizar la estructura de indexación.
- **Tiempo de borrado:** eliminar el dato y actualizar la estructura de indexación.
- **Espacio adicional requerido:** sacrificar generalmente espacio para lograr una mejora en el rendimiento.

## Plan

- ✓ **Conceptos básicos**
- Indexación
- Árboles B+ y B
- Funciones de asociación (*hash*)
- Comparación entre indexación y asociación
- Acceso por claves múltiples

## Indexación (1)

- **Índice primario:** Archivos ordenados secuencialmente según el valor de la clave de búsqueda.
- **Índice secundario:** Archivos ordenados aleatoriamente según el valor de la clave de búsqueda.
- Permiten el procesamiento secuencial de un archivo completo y el acceso aleatorio a registros.

## Indexación (2)

- Estructura de índice ordenado según una clave de búsqueda: índice primario + varios índices secundarios.
- Tipos de índices:
  - **Índice denso:** registros para cada valor de la clave de búsqueda (contiene el valor y un puntero al registro).
  - **Índice disperso:** registros solamente para algunos valores de la clave de búsqueda (encontrar la clave menor o igual al valor buscado, seguir los punteros del archivo).

## Índice primario denso

Clave de búsqueda: SUCURSAL.

		SUCURSAL	NUMERO	NOMBRE	SALDO
CHOLULA	○	CHOLULA	102	Juan	400.00
HUEJOTZINGO	○	CHOLULA	201	Luis	140.00
MOMOXPAN	○	HUEJOTZINGO	320	Esther	890.00
PUEBLA	○	MOMOXPAN	112	Ana	700.00
SAN ANDRES	○	PUEBLA	305	María	350.00
ZAVALETA	○	SAN ANDRES	215	Pablo	700.00
		ZAVALETA	101	Oscar	500.00
		ZAVALETA	225	Pedro	460.00

## Índice primario disperso

Clave de búsqueda: SUCURSAL.

		SUCURSAL	NUMERO	NOMBRE	SALDO
CHOLULA	○	CHOLULA	102	Juan	400.00
MOMOXPAN	○	CHOLULA	201	Luis	140.00
SAN ANDRES	○	HUEJOTZINGO	320	Esther	890.00
		MOMOXPAN	112	Ana	700.00
		PUEBLA	305	María	350.00
		SAN ANDRES	215	Pablo	700.00
		ZAVALETA	101	Oscar	500.00
		ZAVALETA	225	Pedro	460.00

## Índice denso vs. Índice disperso

- Índice denso generalmente **más rápido** que un índice disperso.
- Índice disperso requiere **menos espacio y mantenimiento** para inserciones y eliminaciones.
- Minimizar los accesos a bloques manteniendo el tamaño del índice disperso (espacio extra pequeño):
  - Una entrada de índice por bloque → solamente si los registros buscados no están en bloques de desbordamiento.
  - De manera general, hay que considerar uno o varios bloques por entrada de índice.

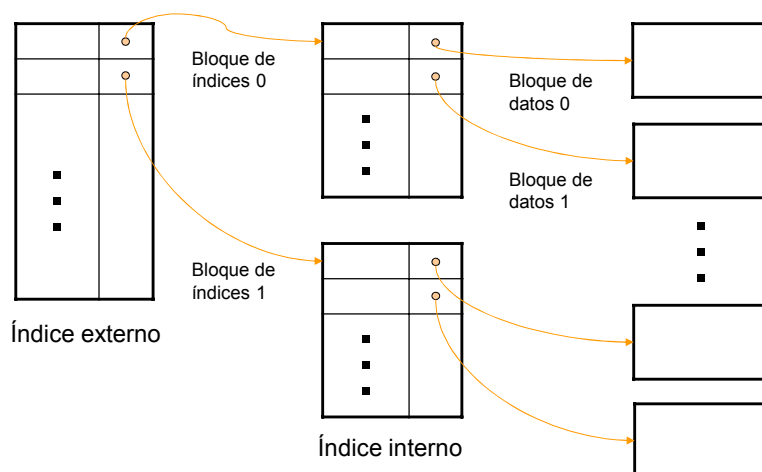
## Actualizar índices (algoritmos)

- Buscar y eliminar un registro en la BD:
  - En un índice denso, suprimir su clave del índice si es el último registro.
  - En un índice disperso, sustituir una clave por otra o eliminarla si ya existe en el índice o llegamos al final del archivo.
- Buscar posición e insertar un registro en la BD:
  - Usando un índice denso, si la clave no aparece, se inserta en el índice.
  - Usando un índice disperso, no se hace ningún cambio a menos que se cree un nuevo bloque.

## Ejemplo

- Archivo con 100 000 registros, con 10 registros almacenados en cada bloque.
- 10 000 registros en un índice disperso (1 registro índice por bloque).
- 100 registros índice en un bloque → índice de 100 bloques.
- Búsqueda binaria de un registro:  $1 + \log_2(b)$  con  $b = 100$  se tienen hasta 7 lecturas de bloque.

## Índices disperso multinivel



➤ Entre tres lecturas y una lectura !!!

## Índices secundarios

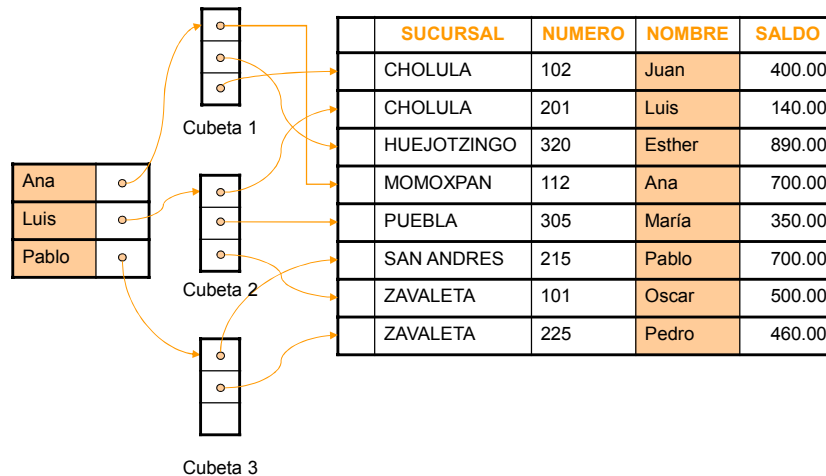
- Un índice secundario puede ser **denso o disperso**.
- Los punteros en un índice secundario señalan a una **lista de punteros (cubeta)** a registros de un archivo.
- Las consultas pueden ser procesadas utilizando **únicamente los punteros**.
- **Costo de administración**: estimar la frecuencia relativa del tipo de operaciones que se pueden realizar.
- En un índice disperso, se puede **asociar la clave** de búsqueda al puntero almacenado en la cubeta.

## Índice secundario denso

		SUCURSAL	NUMERO	NOMBRE	SALDO
Ana	○	CHOLULA	102	Juan	400.00
Esther	○	CHOLULA	201	Luis	140.00
Juan	○	HUEJOTZINGO	320	Esther	890.00
Luis	○	MOMOXPAN	112	Ana	700.00
María	○	PUEBLA	305	María	350.00
Oscar	○	SAN ANDRES	215	Pablo	700.00
Pablo	○	ZVALETA	101	Oscar	500.00
Pedro	○	ZVALETA	225	Pedro	460.00



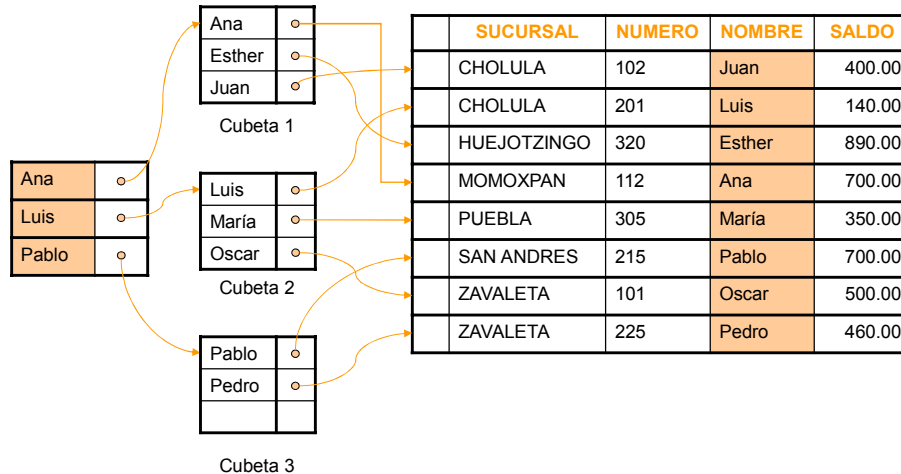
## Índice secundario disperso (1)



## Índices secundarios

- Un índice secundario puede ser **denso o disperso**.
- Los punteros en un índice secundario señalan a una **lista de punteros (cubeta)** a registros de un archivo.
- Las consultas pueden ser procesadas utilizando **únicamente los punteros**.
- **Costo de administración**: estimar la frecuencia relativa del tipo de operaciones que se pueden realizar.
- En un índice disperso, se puede **asociar la clave** de búsqueda al puntero almacenado en la cubeta.

## Índice secundario disperso (2)



## Plan

- ✓ Conceptos básicos
- ✓ Indexación
  - Árboles B+ y B
  - Funciones de asociación (*hash*)
  - Comparación entre indexación y asociación
  - Acceso por claves múltiples

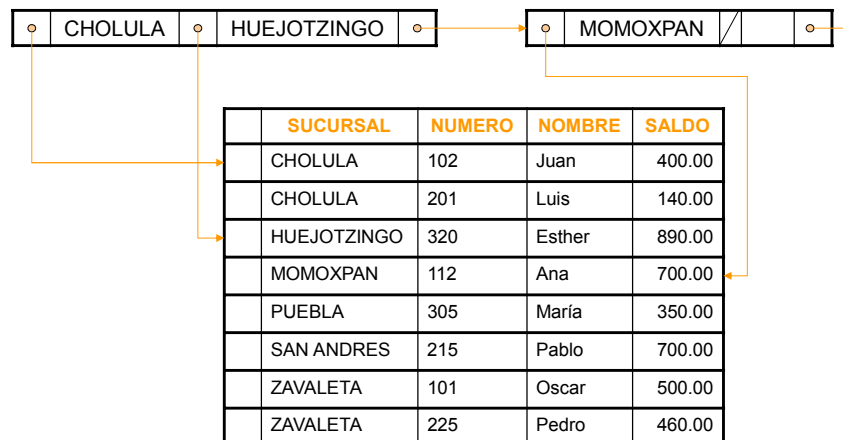
## Árboles B+

- Árbol equilibrado donde todo nodo debe tener entre  $\lceil n/2 \rceil$  y  $n$  hijos, excepto para la raíz. Nodo típico de un árbol B+:

$P_1$	$K_1$	$P_2$	...	$P_{n-1}$	$K_{n-1}$	$P_n$
-------	-------	-------	-----	-----------	-----------	-------

- Contiene hasta  $n-1$  valores de clave de búsqueda y  $n$  punteros. Si  $i < j$ , entonces  $K_i < K_j$ .
- Para  $1 \leq i < n$  en un nodo hoja,  $P_i$  apunta a un registro (índice primario) o a una cubeta (índice secundario).

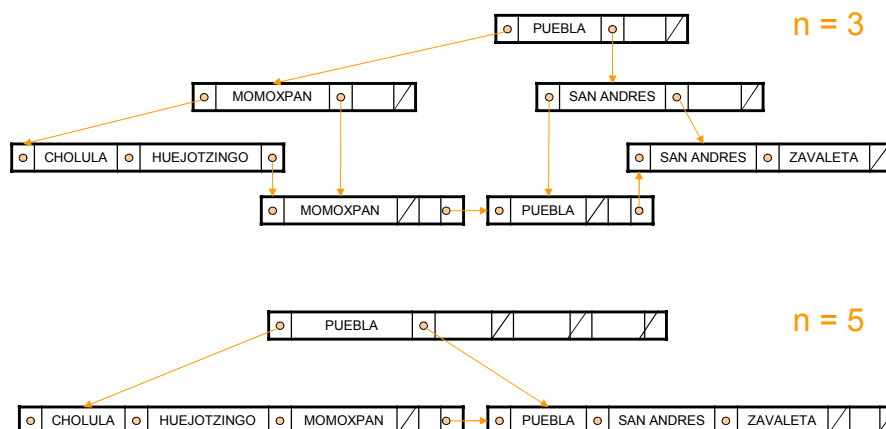
## Árboles B+ (nodos hoja)



## Árboles B+ (continuación)

- El puntero  $P_n$  permite un procesamiento secuencial del archivo (nodos hojas).
- Los nodos hojas forman un **índice denso**.
- Los nodos no hojas son **índices dispersos**, donde todos sus punteros apuntan a otros nodos.
- Siempre es posible construir un árbol B+ en el que todos los nodos distintos del nodo raíz contienen por lo menos  $\lceil n / 2 \rceil$  punteros.
- El equilibrio de los árboles asegura un buen rendimiento en las consultas, inserciones y eliminaciones.

## Árboles B+: ejemplos

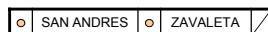


## Árboles B+: consultas

- Tamaño de nodo (valor de  $n$ ) igual al tamaño de un bloque (o múltiplo).
- Para  $k$  valores diferentes de clave, el camino no es más largo que  $\log_{\lceil n/2 \rceil} (k)$ .
  - Con  $k$  igual a un millón y  $n$  entre 11 y 101, una búsqueda requeriría entre seis y tres accesos respectivamente.
- **Buscar un registro con clave  $V$ :**
  - Clave más grande en el nodo que contiene a  $V$ .
  - Mayor o igual al valor buscado  $V$ : seguir el apuntador derecho.
  - En caso contrario, seguir el apuntador izquierdo ↔ **Hasta antes de llegar a un nodo hoja.**

## Árboles B+: inserciones (1)

- Partir nodos con  $n-1$  claves.
- Asegurar que el equilibrio se conserva.
- En general, poner los primeros nodos  $\lceil (n-1) / 2 \rceil$  en el nodo existente y los valores restantes en un nuevo nodo.

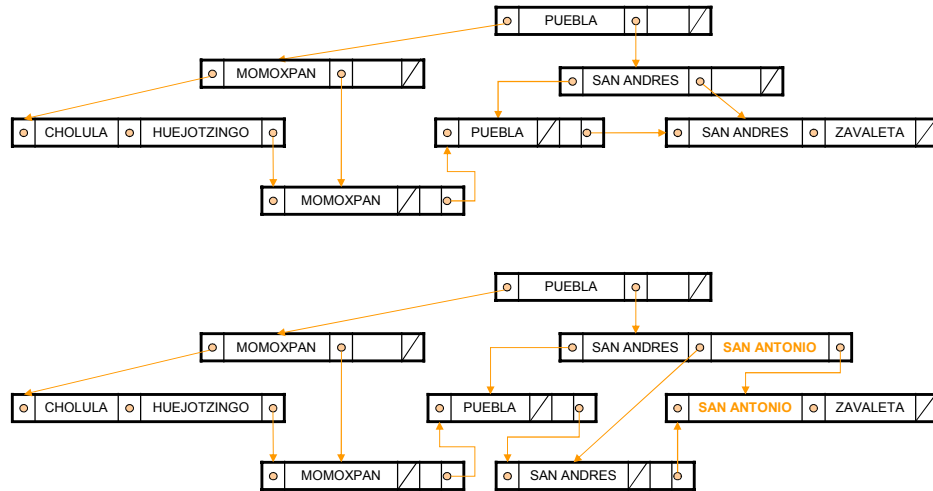


$n = 3$



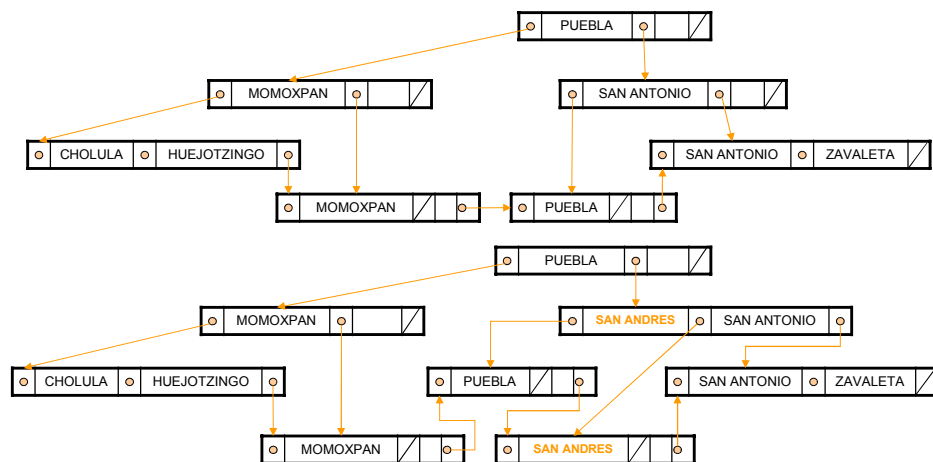
Insertar: un registro con el valor 'SAN ANTONIO' en SUCURSAL

## Árboles B+: inserciones (2)



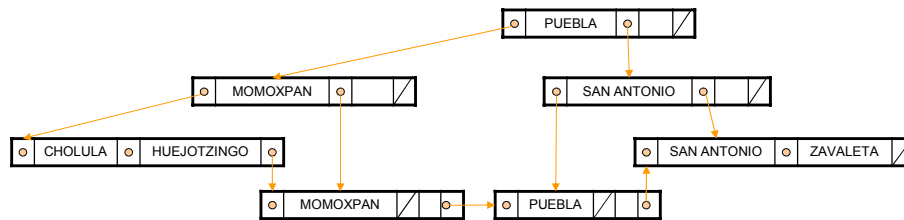
## Árboles B+: eliminaciones (1)

Eliminar: los registro con el valor 'SAN ANDRES' en SUCURSAL



## Árboles B+: eliminaciones (1)

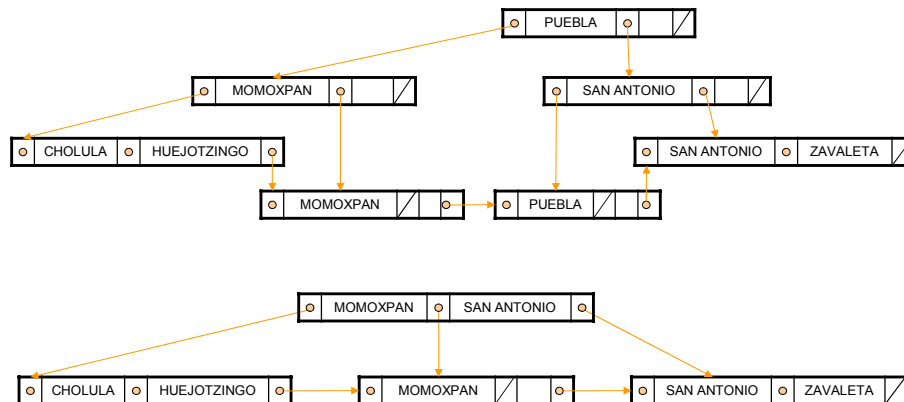
Eliminar: los registro con el valor 'PUEBLA' en SUCURSAL



- Nodos con **información útil**.
- No siempre es posible fusionar nodos → **redistribuir los punteros**.

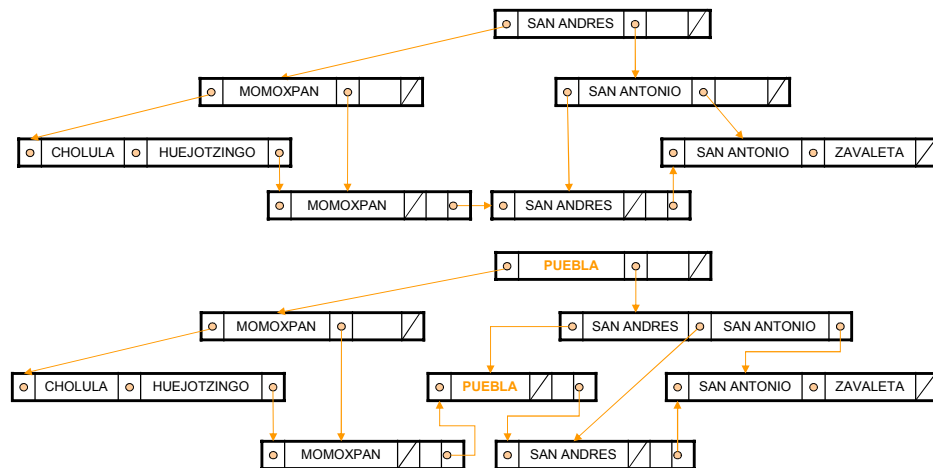
## Eliminaciones: información útil

Eliminar: el registro con el valor 'PUEBLA' en SUCURSAL



## Eliminaciones: redistribuir punteros

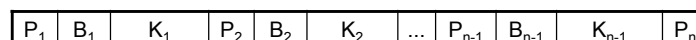
Eliminar: los registro con el valor 'PUEBLA' en SUCURSAL



**UDLAP**  
UNIVERSIDAD DE LAS  
AMÉRICAS PUEBLA

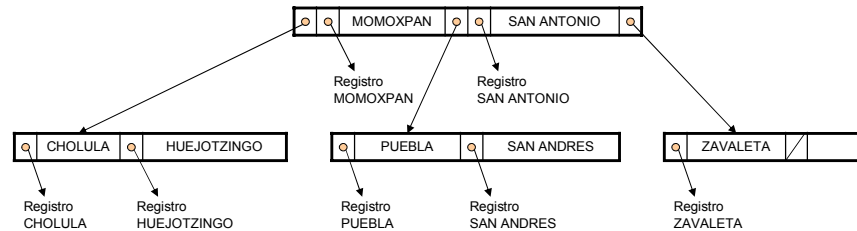
## Árboles B

- **Eliminar el almacenamiento redundante** de valores en la clave de búsqueda.
- Incluir un campo de **puntero adicional** en los nodos no hojas (encontrar la clave antes de leer un nodo hoja).





## Árboles B



- Nodos no hojas más grandes: complica la gestión.
- Eliminación más complicada: elegir la clave apropiada para sustituir una clave en un nodo no hoja.

## Plan

- ✓ Conceptos básicos
- ✓ Indexación
- ✓ Árboles B+ y B
  - Funciones de asociación (*hash*)
  - Comparación entre indexación y asociación
  - Acceso por claves múltiples

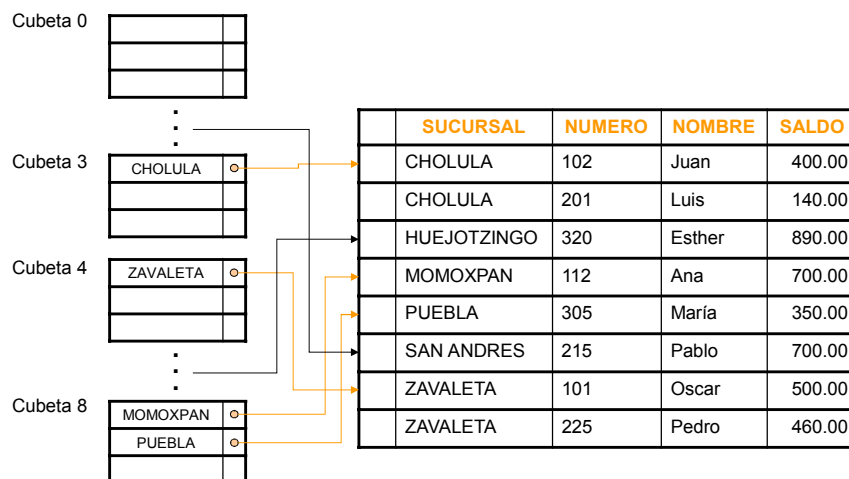
## Asociación estática (1)

- **Mejorar el rendimiento**: evitar el acceso a una estructura multinivel de índices.
- **Tabla de asociación** (*hashing table*): índice denso dividido entre un número de diferentes cubetas  $B_i$ .
- **Función de asociación** (*hashing function*): función sobre el valor de búsqueda del registro deseado  $K$ :

$$h(K) \rightarrow B_i$$

- **Conjunto finito de claves**  $\{K_1, K_2, \dots, K_n\}$ : se puede estimar cuántos valores de clave de búsqueda van a ser almacenados.

## Asociación estática: ejemplo



## Asociación estática (2)

- **Función deficiente:** asignar todas las claves a la misma cubeta.
- **Función ideal:** asignar cada clave de búsqueda a una cubeta distinta.
  - **Distribución uniforme:** cada cubeta tiene asignado el mismo número de claves de búsqueda posibles.
  - **Distribución al azar:** cada cubeta en promedio tendrá casi el mismo número de claves asignadas.

## Asociación estática: ejemplo

- Asignar nombres que empiezan con la letra i del alfabeto a la cubeta i (27 cubetas):

Función sencilla, pero falla en la distribución al azar. Más sucursales que empiezan con letras "A", "C" y "S".

- Aprovechar la representación binaria de los caracteres:

Calcular el módulo de la suma de la representación binaria de caracteres entre el número de cubetas.

## Asociación estática (3)

- Inserción y eliminación casi tan simples como la búsqueda.
- Función de *hashing* elegida al implementar el sistema y no se puede cambiar fácilmente después.
- Dado un conjunto fijo de direcciones de cubetas  $B_i$  con  $0 \leq i \leq n$ :
  - Si  $n$  es excesivamente grande  $\rightarrow$  Se desperdicia espacio.
  - Si  $n$  es demasiado pequeña  $\rightarrow$  Se tiene un bajo rendimiento.
- Si el tamaño de tabla de asociaciones es igual al doble del número de claves, se tiene un buen equilibrio entre espacio y rendimiento.

## Asociación estática (4)

Un problema serio con esta técnica es fijar el conjunto de cubetas  $B_i$ .  
Técnicas usadas para definir una función de asociación:

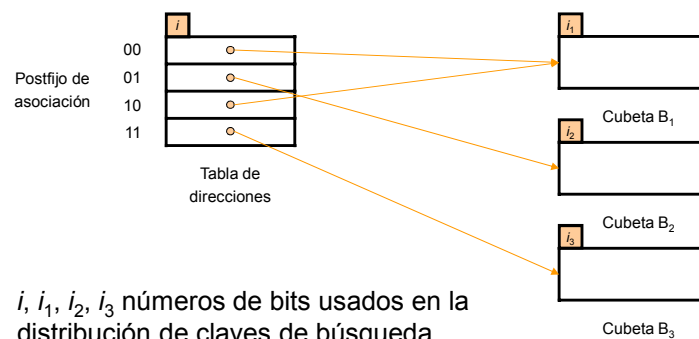
- **Tamaño actual del archivo** (degradación del rendimiento conforme crezca la base de datos).
- **Tamaño anticipado** (al principio desperdicia una cantidad importante de espacio).
- **Reorganización periódica** de la estructura (elevado consumo de tiempo, prohibir el acceso).

## Asociación dinámica (1)

Modificar de manera dinámica la función de asociación.

- Asociación extensible: dividir y fusionar cubetas.
  - Eficiencia de espacio.
  - Poco tiempo extra requerido (reorganizar una sola cubeta).
- Función de *hashing* uniforme y aleatoria con rango relativamente grande.
  - Enteros binarios de 32 bits (entre 0 y  $2^{32}$ ).
  - Cubetas según la demanda (posición relativa de  $i$  bits,  $0 \leq i \leq 32$ ).

## Asociación dinámica (2)



Ejemplo:  $i = 2, i_1 = 1, i_2 = 2, i_3 = 2$ .

## Asociación dinámica (3)

- Buscar:  $h(K) \rightarrow B_j$ .
- Insertar un registro (partir cubetas y redistribuir registros):
  - $i = i_j$  : aumentar el tamaño de la tabla.
  - $i > i_j$  : incrementar y reorganizar los registros en dos cubetas.
- Eliminar un registro (unir cubetas):
  - Eliminar cubetas sin registros (apuntadores).
  - Reducir el tamaño de la tabla si para toda  $i_j$  tenemos  $i > i_j$ .

## Asociación dinámica: ejemplo



SUCURSAL	NUMERO	NOMBRE	SALDO
CHOLULA	102	Juan	400.00
HUEJOTZINGO	320	Esther	890.00
MOMOXPAN	112	Ana	700.00
PUEBLA	305	María	350.00
SAN ANDRES	215	Pablo	700.00
SAN ANTONIO	222	Marta	295.00
ZAVALETA	101	Oscar	500.00

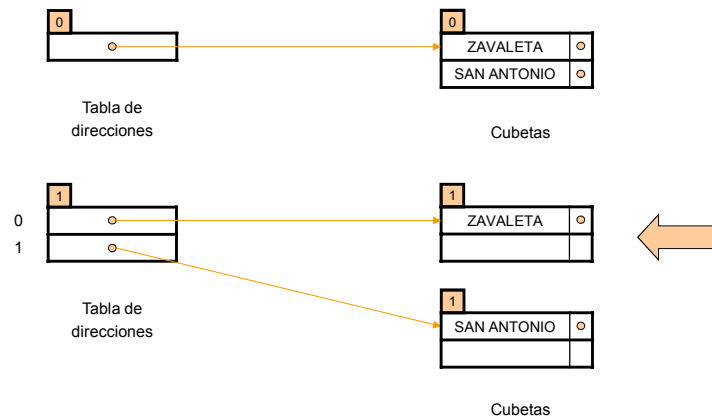
$h(\text{SUCURSAL}) \rightarrow B_i$
0010 1101 1111 1011 0010 1100 0000 <b>0101</b>
1101 0101 1101 1110 0100 0110 1001 <b>0011</b>
1010 0011 1010 0000 1100 0110 1001 <b>1000</b>
1000 0111 1110 1101 1011 1111 0011 <b>0110</b>
1111 0001 0010 0100 1001 0011 0110 <b>1011</b>
1011 0101 1010 0110 1100 1001 1110 <b>0111</b>
0101 1000 0011 1111 1001 1100 0011 <b>0000</b>

## Asociación dinámica: ejemplo



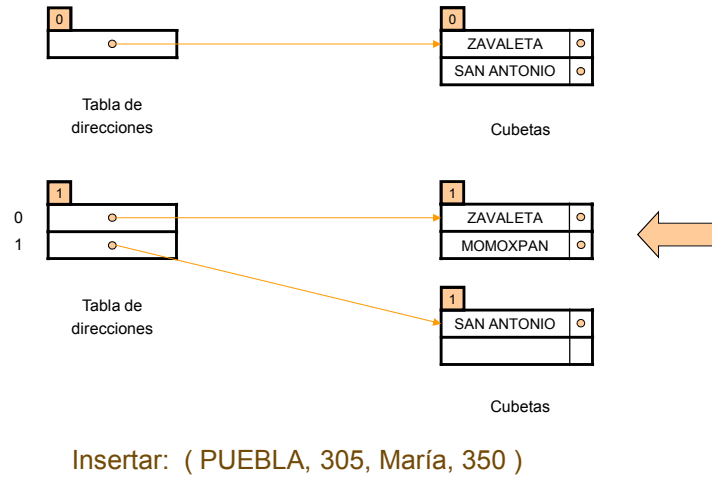
Insertar: ( MOMOXPAN, 112, Ana, 700 )

## Asociación dinámica: ejemplo

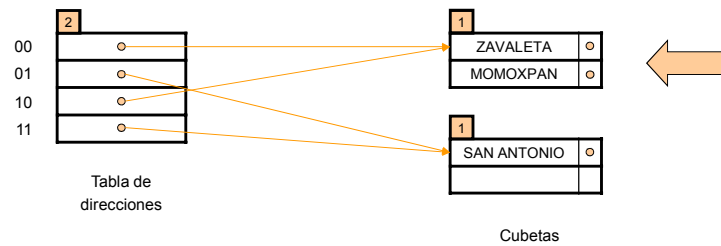


Insertar: ( MOMOXPAN, 112, Ana, 700 )

## Asociación dinámica: ejemplo

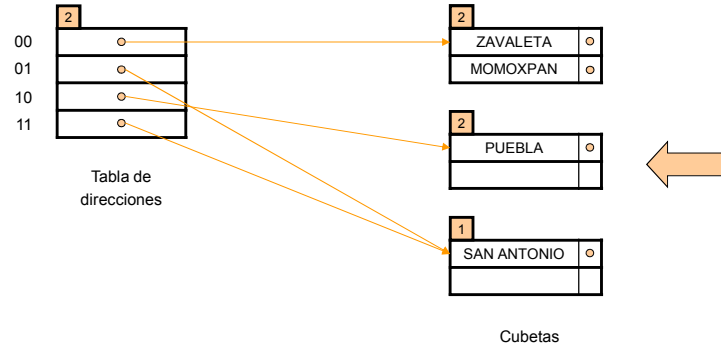


## Asociación dinámica: ejemplo



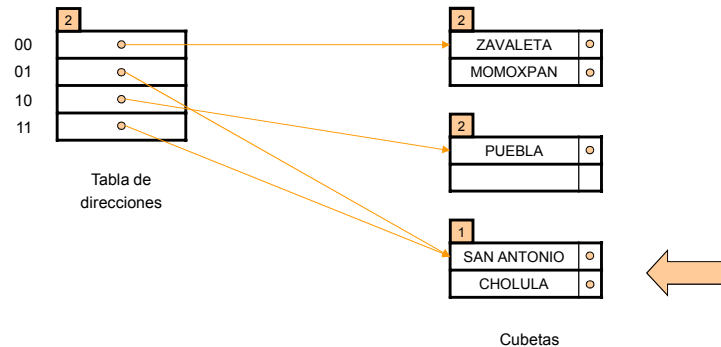


## Asociación dinámica: ejemplo



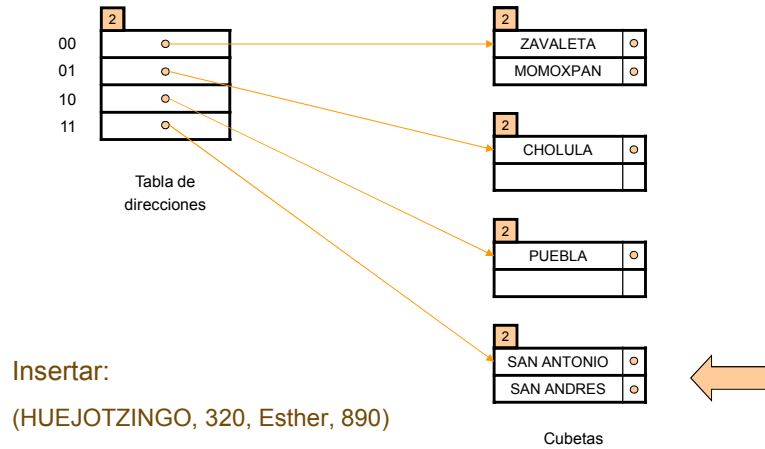
Insertar: (CHOLULA, 102, Juan, 400 )

## Asociación dinámica: ejemplo

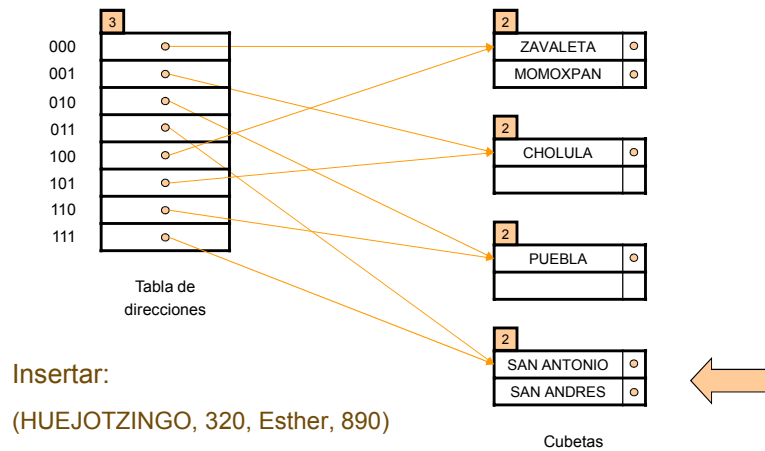


Insertar: (SAN ANDRES, 215, Pablo, 700 )

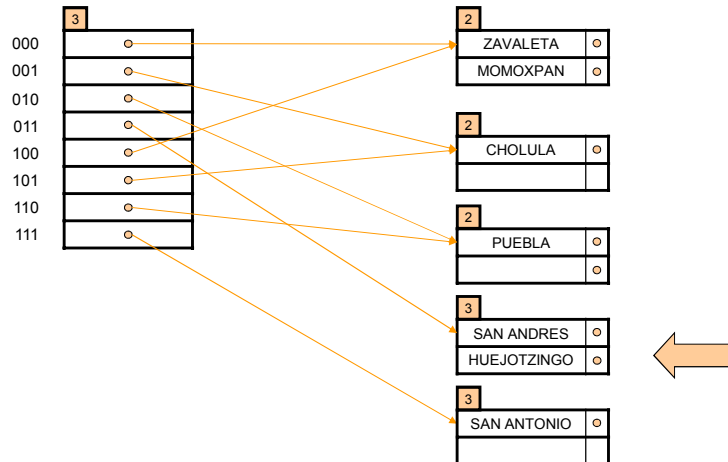
## Asociación dinámica: ejemplo



## Asociación dinámica: ejemplo



## Asociación dinámica: ejemplo



## Plan

- ✓ Conceptos básicos
- ✓ Indexación
- ✓ Árboles B+ y B
- ✓ Funciones de asociación (*hash*)
  - Comparación entre indexación y asociación
  - Acceso por claves múltiples

## Indexación / asociación (*hash*)

- Muchos esquemas de indexación y de asociación:
  - Mucho código y espacio adicional.
  - Decisión final del diseñador de la base de datos.
- Una sola forma de indexación o de asociación.
- Factores a considerar:
  - Reorganización periódica del índice o de la estructura de asociación.
  - Frecuencia relativa de inserciones y eliminaciones.
  - Tiempo de acceso promedio vs. tiempo de acceso en el peor de los casos.
  - Tipos de consultas que harán normalmente los usuarios.

## Tipos de consultas (1)

- Funciones de asociación (*hash*): tiempo constante.

**Recuperar los registros del archivo A  
cuyo campo  $C_i$  sea igual a V**

En el peor de los casos, el tiempo de búsqueda es proporcional al número de valores del atributo  $C_i$  en A.

- Árboles B+ o árboles B: tiempo de búsqueda proporcional al logaritmo del número de valores de  $C_i$  en A.

## Tipos de consultas (2)

- Árboles B+ o árboles B: búsqueda de un rango de valores.

Recuperar los registros del archivo A

cuyo valor del campo  $C_i$  se encuentre entre  $V_1$  y  $V_2$

- Funciones de asociación (*hash*): secuencia de cubetas.
- Función de asociación que conserve el orden, si  $K_1 < K_2$  entonces  $h(K_1) < h(K_2)$ . Difícil de definir  $h$ .

## Plan

- ✓ Conceptos básicos
- ✓ Indexación
- ✓ Árboles B+ y B
- ✓ Funciones de asociación (*hash*)
- ✓ Comparación entre indexación y asociación
- Acceso por claves múltiples

## Archivo DEPOSITO

SUCURSAL	NUMERO	NOMBRE	SALDO
CHOLULA	102	Juan	400.00
CHOLULA	201	Luis	140.00
PUEBLA	305	María	350.00
SAN ANDRES	215	Pablo	700.00
ZAVALETA	101	Oscar	500.00
ZAVALETA	225	Pedro	460.00

## Acceso por claves múltiples

Saldo de todos los depósitos de Juan en la sucursal de Cholula:

**Buscar todos los registros en DEPOSITO**

**donde SUCURSAL = "Cholula" y NOMBRE = "Juan"**

- Utilizar el índice por SUCURSAL.
- Utilizar el índice por NOMBRE.
- Tomar la intersección de los índices por SUCURSAL y por NOMBRE:  
gran cantidad de registros con una intersección pequeña.

## Estructura de rejilla

- Arreglo n-dimensional ordenado según los valores de las claves, v.g., SUCURSAL y NOMBRE.
- Estructura apropiada también para consultas con una sola clave.
- Cantidad de espacio y tiempo extra en las inserciones y eliminaciones de registros.

## Función de asociación dividida

- Función de asociación (*hash*) n-dimensional para la clave: (SUCURSAL, NOMBRE).
- Restricción adicional sobre la función de asociación  $h$ : valor de asociación dividido.

Clave de búsqueda	$h(\text{SUCURSAL, NOMBRE})$
( CHOLULA, Juan )	( 0101, 0111 )
( CHOLULA, Luis )	( 0101, 0110 )
( PUEBLA, María )	( 0110, 1101 )
( SAN ANDRES, Pablo )	( 1011, 0010 )
( ZAVALETA, Oscar )	( 0000, 1001 )
( ZAVALETA, Pedro )	( 0000, 1011 )