

CONCEPTOS DE BASES DE DATOS

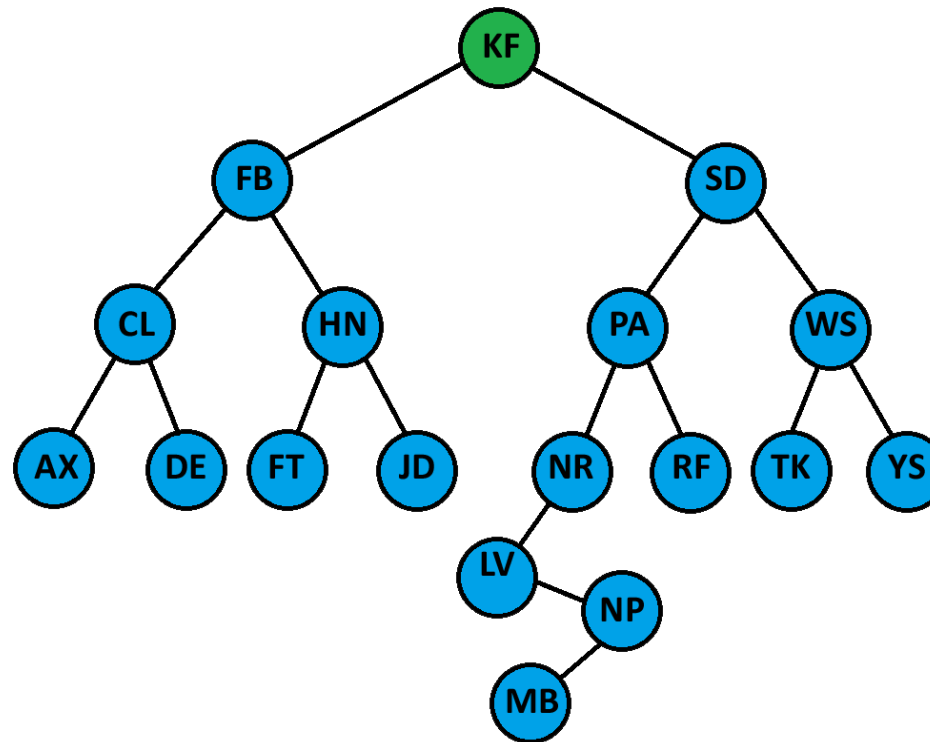
CLASE 6



Árboles

Resumen

- Un árbol **binario** se desbalancea fácilmente



- Los árboles **balanceados en altura** proporcionan una solución aceptable al problema de **mantener un índice ordenado**
 - Los cambios son locales (no masivos)
- Pero como son árboles binarios, son muy profundos al usarlos con muchos elementos
 - Para 1.000.000 claves, un árbol **AVL** puede tener 28 niveles → **demasiado**

- Los árboles **binarios paginados** introducen la idea de **agrupar claves en páginas**, pero no se ha encontrado la forma de hacerlo eficientemente
- Problemas
 - ¿Cómo construirlo?
 - ¿Cómo elegir la raíz?
 - ¿Cómo mantenerlo balanceado?

Árboles

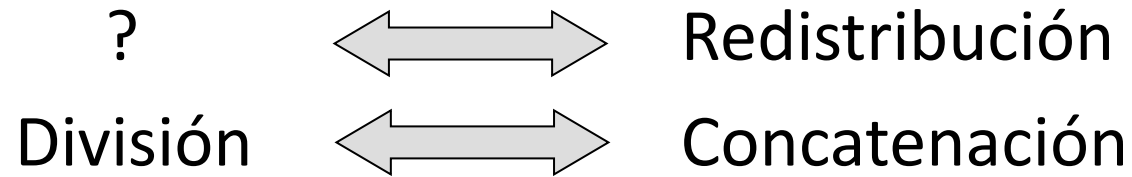
Resumen

- Los árboles **multicamino** son una buena opción que resuelve uno de los problemas que tienen los árboles binarios → **cantidad de niveles**
- Para asegurar que las búsquedas realizadas tengan un costo logarítmico → se debe asegurar que el árbol esta **balanceado**

- Árboles B (orden M)
 - Son árboles **multicamino**
 - Se construyen en **forma ascendente** → se mantiene **balanceado a bajo costo**
 - Cantidad de hijos por nodo:
 - Mínimo: $\lceil M/2 \rceil$ hijos (menos raíz y terminales)
 - Máximo: M hijos
 - Nodos terminales al mismo nivel
 - Mínimo: $\lceil M/2 \rceil - 1$ claves
 - Máximo: $M - 1$ claves

- Árboles B
 - **Inserción:** si el nuevo registro produce **overflow**
 - **División del nodo.** Promoción. puede propagarse hasta **generar una nueva raíz.**
 - **Eliminación:** si el nodo queda en **underflow**
 - **Redistribución.** Alguno de sus **nodos adyacentes hermanos** tiene elementos suficientes → se realiza la redistribución entre dicho nodo y el que quedó en underflow.
 - **Concatenación.** Ninguno de sus **nodos adyacentes hermanos** tiene elementos suficientes → se realiza la concatenación entre uno de ellos y el que quedó en underflow. Es posible **perder un nivel del árbol.**

- Inserción **vs** Eliminación



- La redistribución podría **posponer la creación** de nuevos nodos
- Se pueden generar **árboles B más eficientes** en términos de utilización de espacio

Árboles

Balanceados (B*)

- **Árbol B***: Es un **árbol B especial** en el que cada nodo está lleno por lo menos en $\frac{2}{3}$ partes (excepto la raíz)
- Propiedades de un **árbol B*** de orden **M**
 - **Cantidad de hijos por nodo:**
 - Máximo: **M** hijos
 - Mínimo: $\lceil \frac{2M - 1}{3} \rceil$ hijos (menos raíz y terminales)
 - Mínimo raíz: **2** hijos (o sino **ninguno**)
 - **Nodos no terminales:** **K** hijos \rightarrow **K-1** claves
 - **Nodos terminales (hoja):** todos están al mismo nivel
 - Máximo: **M-1** claves
 - Mínimo: $\lceil \frac{2M - 1}{3} \rceil - 1$ claves

- **Búsqueda:** similar a árbol B
- **Inserción:** redistribución / división. Tres políticas:
 - **Derecha** (de un lado):
 - Redistribuir con nodo adyacente hermano derecho (o izq. si es el último). Si esta lleno se realiza la división usando ambos nodos.
 - **Derecha o Izquierda** (de un lado u otro)
 - Si el nodo derecho está lleno, se intenta hacerlo con el izquierdo. Si ambos están llenos se usa el nodo en overflow y su hermano derecho para realizar la división.
 - **Derecha e Izquierda** (de un lado y otro)
 - Similar anterior, pero al realizar la división se usan los tres nodos, generando cuatro nodos que tendrán 3/4 parte llena.
- **Eliminación:** similar a árbol B (redist./concat.)

- Manejo de nodos (páginas) en buffers
 - **Objetivo:** minimizar cantidad de accesos a disco
 - El SO administra gran cantidad de buffers
 - Buffers más usados → se mantienen en RAM
 - Estrategias de reemplazo: **LRU** (Least Recently Used)
 - Tamaño buffer → capacidad del nodo
 - Si se almacenan nodos en un **buffer en RAM**
 - Sólo el nodo raíz → se ahorra 1 acceso a mem. secundaria
 - Nodo raíz y otros principales → **se ahorran varios accesos** a memoria secundaria

Árboles

Balanceados

- Manejo de nodos (páginas) en buffers → Ej.
 - Cantidad de claves = 2400
 - Cantidad de nodos = 140
 - Altura del árbol = 3

Cantidad de páginas en buffers en RAM	1	5	10	20
Cantidad de accesos promedio por búsqueda	3	1.71	1.42	0.97

Árboles

Balanceados

- Conclusiones
 - Los árboles balanceados B y B* permiten el **acceso aleatorio por clave** de forma eficiente
 - Además, los árboles B* exigen que sus nodos se mantengan más llenos que los árboles B → **menor altura**
 - ¿Cómo se realiza un procesamiento secuencial ordenado por clave?

- Archivos **secuenciales indizados**
 - Permiten dos formas de acceso:
 - **Indizado**: acceso aleatorio por clave
 - **Secuencial**: acceso secuencial ordenado por clave
 - Alternativas de implementación con los métodos vistos hasta ahora:
 - **Orden físico**: **ineficiente** al sufrir modificaciones
 - **Árbol B**: rápida recuperación para acceso aleatorio pero **ineficiente** para acceso secuencial
 - Es necesaria una solución que permita **ambos accesos de forma eficiente** → **Árboles B+**

Árboles

Balanceados (B+)

- Los **árboles B+** constituyen una mejora sobre los árboles B
 - Conservan la propiedad de **acceso aleatorio rápido** de los árboles B (acceso indizado)
 - Permiten además un **recorrido secuencial ordenado rápido**

Árboles

Balanceados (B+)

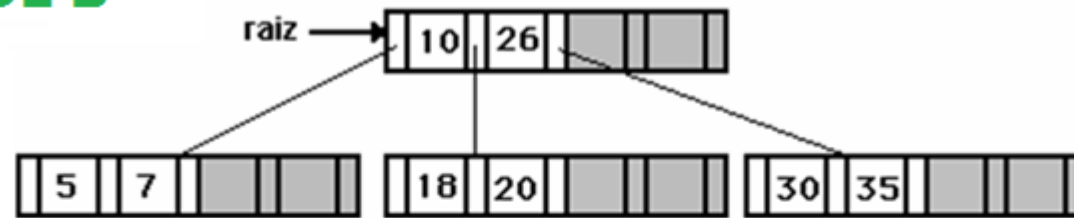
- Un árbol B+ consta de dos partes
 - Índice → nodos interiores
 - En la raíz y nodos interiores **se duplican las claves** necesarias para definir los caminos de búsqueda
 - Secuencia → nodos hojas enlazados
 - Todas las claves se encuentran en los nodos hoja
 - Los nodos hoja están vinculados → se obtiene una **trayectoria secuencial** para recorrer las claves del árbol

Árboles

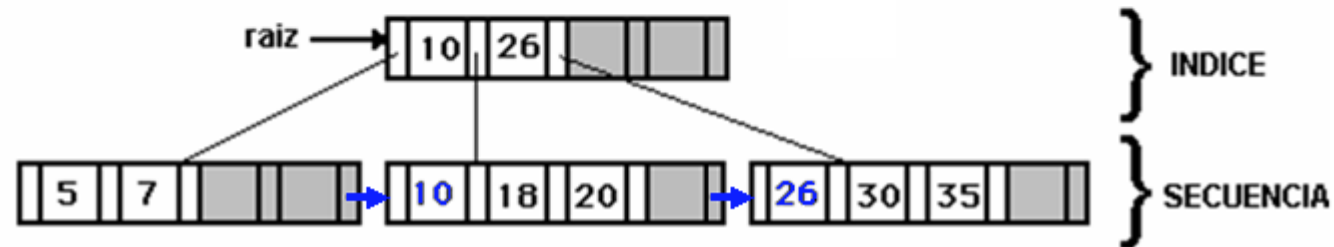
Balanceados (B+)

- Comparación

ÁRBOL B



ÁRBOL B+



Árboles

Balanceados (B+)

- Un árbol B+ **ocupa más espacio** que un árbol B equivalente, ya que algunas de sus claves se encuentran más de una vez en el árbol
- Las claves del nodo raíz y nodos interiores se utilizan **únicamente como índice** para las búsquedas

Árboles

Balanceados (B+)

- Propiedades de un **árbol B+** de orden **M**
 - **Cantidad de hijos por nodo:**
 - **Máximo:** **M** hijos
 - **Mínimo:** **[M/2]** hijos (menos raíz y terminales)
 - **Mínimo raíz:** **2** hijos (o sino **ninguno**)
 - **Nodos no terminales (interiores)**
 - Son punteros a los datos → **no contienen datos**
 - **Nodos terminales (hoja):** todos están al mismo nivel
 - Representan un conjunto de datos y **son linkados juntos**

- **Búsqueda**
 - **No debe detenerse** cuando se encuentre la clave en la **raíz** o en un **nodo interior**
 - La búsqueda continúa por el nodo apuntado por la **rama derecha** de dicha clave **hasta llegar a un nodo hoja**
 - Si se llega a un nodo hoja sin éxito → la clave **no se encuentra** en el archivo

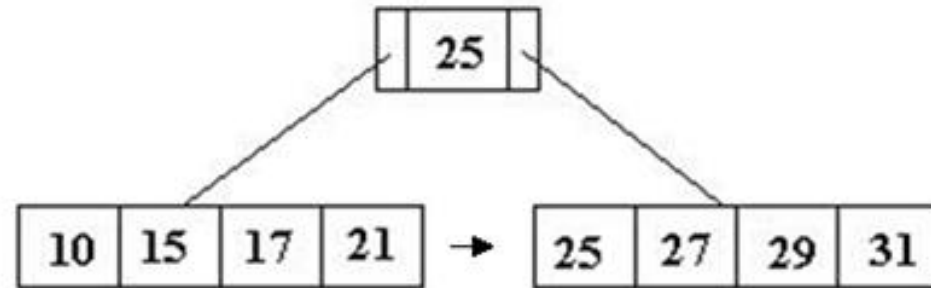
- **Insertión**

- Es **similar a la de los árboles B**, excepto cuando se desea insertar una clave en donde el nodo se encuentra lleno
- En ese caso, el **nodo lleno se divide** en otros dos, pero ahora:
 - El primero contendrá $M/2$ claves
 - El segundo $1+M/2$ claves, ya que lo que subirá al padre será una **copia de la clave central**

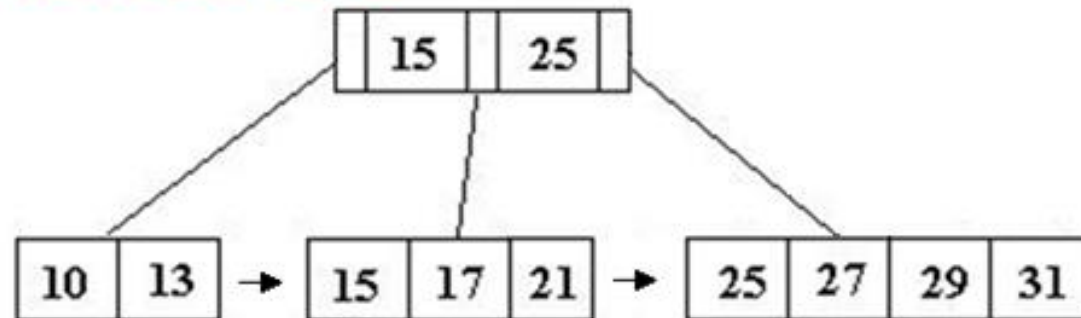
Árboles

Operaciones (B+)

- Inserción (orden $M = 5$)



AL INSERTAR LA CLAVE 13:



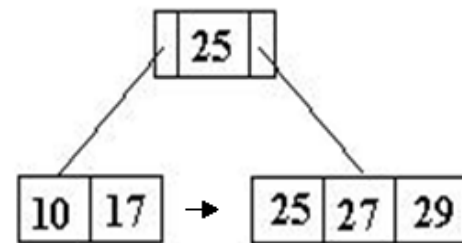
- **Eliminación**

- Todas las claves se encuentran en los nodos hoja → **más simple que en los árboles B**
- Si al eliminar **no se produce underflow**
 - Las claves de la raíz o nodos internos **no se modifican** ya que siguen siendo un **separador válido** entre las claves de los nodos descendientes
- Si al eliminar **se produce underflow**
 - Es necesario una **redistribución** o **fusión** de las claves, que involucra a las hojas y el índice

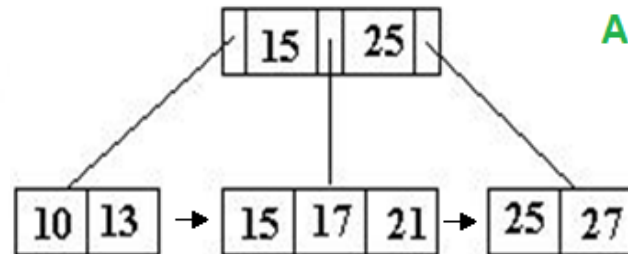
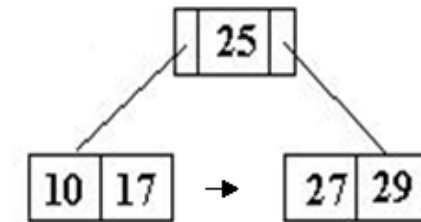
Árboles

Operaciones (B+)

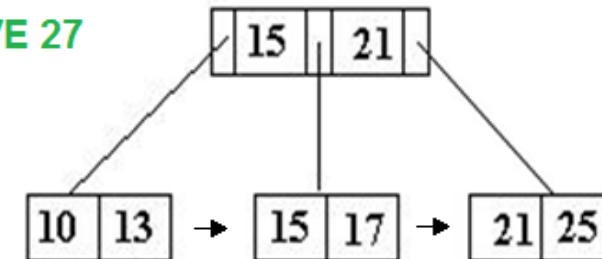
- **Eliminación** (orden $M = 6$)



AL ELIMINAR LA CLAVE 25



AL ELIMINAR LA CLAVE 27

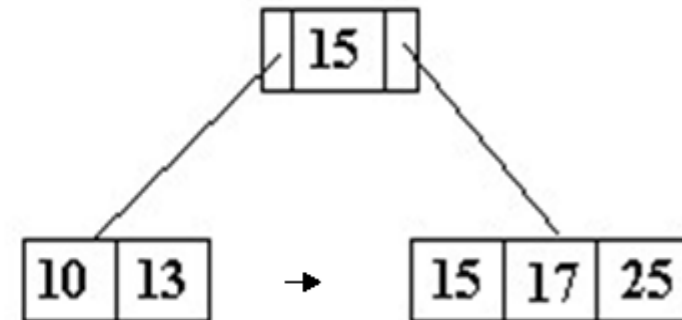
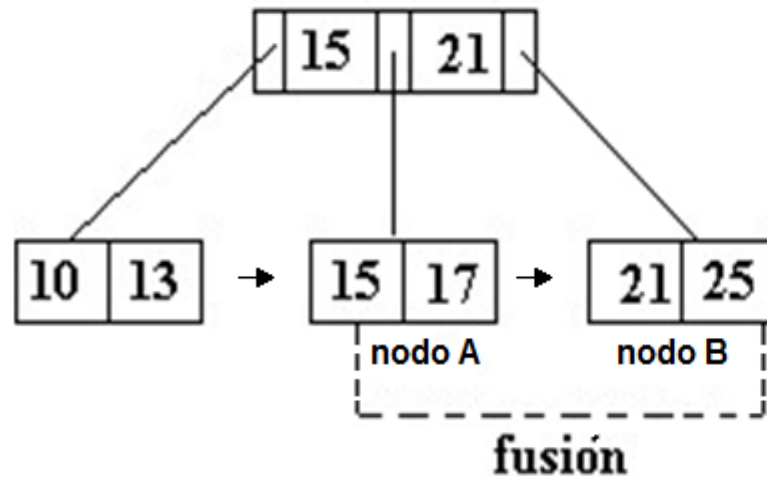


Árboles

Operaciones (B+)

- **Eliminación** (orden $M = 6$)

AL ELIMINAR LA CLAVE 21



- **Separadores**

- Son **derivados de las claves** de los registros que limitan un bloque en el conjunto de secuencia
- Separadores más cortos → **ocupan espacio mínimo**

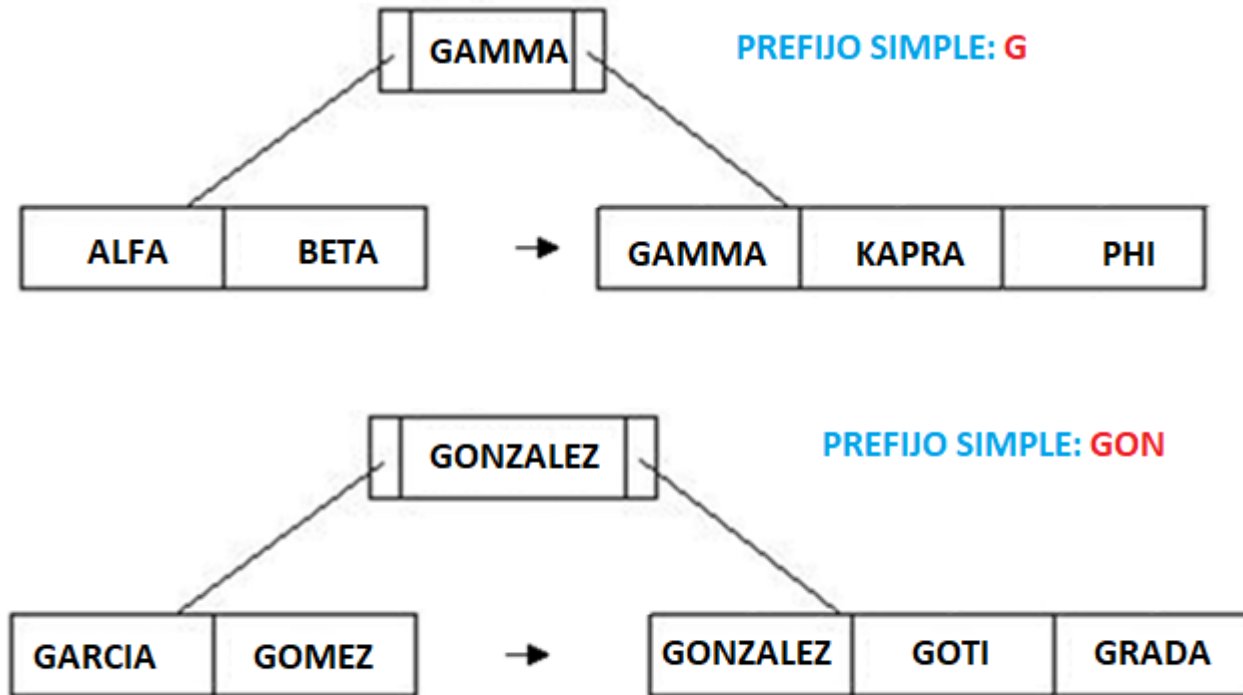
- **Árbol B+ de prefijos simples**

- Es un árbol B+ en el cual el conjunto índice está constituido por **separadores más cortos**

Árboles

Balanceados (B+)

- **Árbol B+ de prefijos simples**



Árboles

Balanceados

- **Comparación**

	Árbol B	Árbol B+
Ubicación de datos	Nodos (cualquiera)	Nodos terminales
Tiempo de búsqueda	Equivalente (puede ser algo mejor)	Equivalente
Procesamiento secuencial ordenado	Lento (complejo)	Rápido (con punteros)
Inserción/Eliminación	Equivalente	Equivalente