Árboles B y B+

Unidad 8.

Almacenamiento y Estructuras de Archivos

Índices de Acceso Secuencial

Índices de acceso secuencial

- Las estructuras de archivos con archivos indexados pueden verse de dos maneras:
  - Indexado: El archivo puede ser visto como un conjunto de registros indexado por llave
  - Secuencia: El archivo puede ser accesado secuencialmente (sin búsqueda), regresando registros ordenados por llave

Acceso secuencial indexado

- La idea es tener un método de organización que contenga ambos tipos de enfoque

- Un archivo de índices con estructura de árbol B proporciona un excelente acceso indexado a cualquier registro por medio de una llave, incluso si los registros son añadidos o eliminados

Acceso secuencial indexado

- Es deseable utilizar este archivo como una parte de una mezcla cosecuencial

- En un procesamiento cosecuencial se quieren recuperar todos los registros ordenados por llave

Acceso secuencial indexado

- Los registros en este sistema de archivos son introducidos de manera secuencia pero no ordenada

- La única manera de recuperarlos sin tener que ordenarlos es a través del índice

Acceso secuencial indexado

- Sin embargo, dado un archivo de N registros, se requieren N accesos para este trabajo

- Esto es mucho menos eficiente que una lectura secuencial

Conjuntos en secuencia

Conjunto en secuencia

- Conjunto de registros en un orden físico ordenados por el valor de su llave conforme son eliminados o añadidos

Uso de bloques

- Es muy costoso ordenar un archivo completo después de insertar o eliminar un registro

- Una de las herramientas para minimizar esto costos son los bloques

Uso de bloques

- Se utilizan los bloques para realizar un ordenamiento de un conjunto en secuencia
- Cuando se colocan los registros en bloques, los bloques se convierten en la unidad básica de la entrada y la salida
- Se lee y se escribe el bloque entero como uno solo

Manteniendo el orden con ayuda de bloques

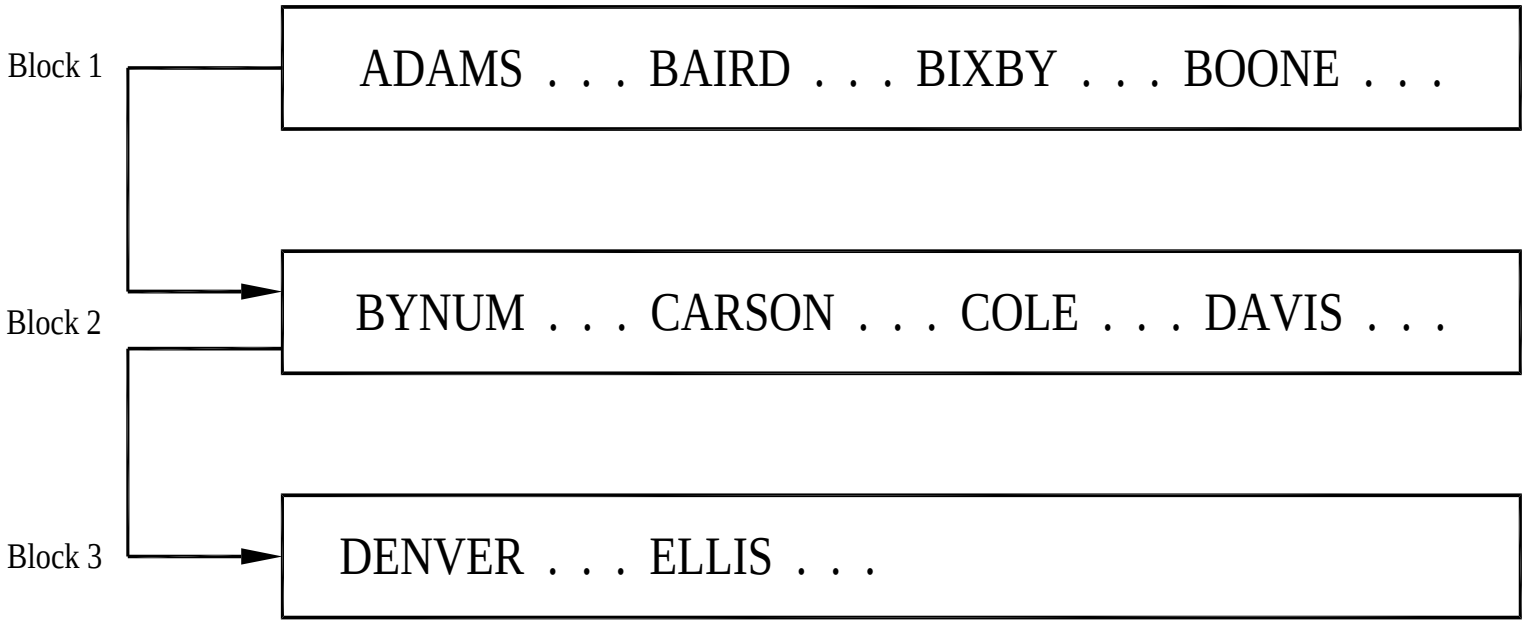
- Suponer que se tienen registros que están manejados por una llave basada en el último nombre y recolectados juntos de tal manera que se encuentran cuatro en un bloque

- También se incluyen campos en cada bloque que apuntan a los bloques siguiente y al anterior

- Al igual que con lo árboles B, la inserción de nuevos registros en bloques pueden causar un sobre flujo

- La condición de sobre flujo puede ser manejada por un proceso de separación de bloques que es análogo al utilizado en un árbol B

Ejemplo


Block 1
ADAMS . . . BAIRD . . . BIXBY . . . BOONE . . .
Block 2
BYNUM . . . CARSON . . . COLE . . . DAVIS . . .
Block 3
DENVER . . . ELLIS . . .

Ejemplo, cont...

- Se inserta un nuevo registro con la llave CARTER

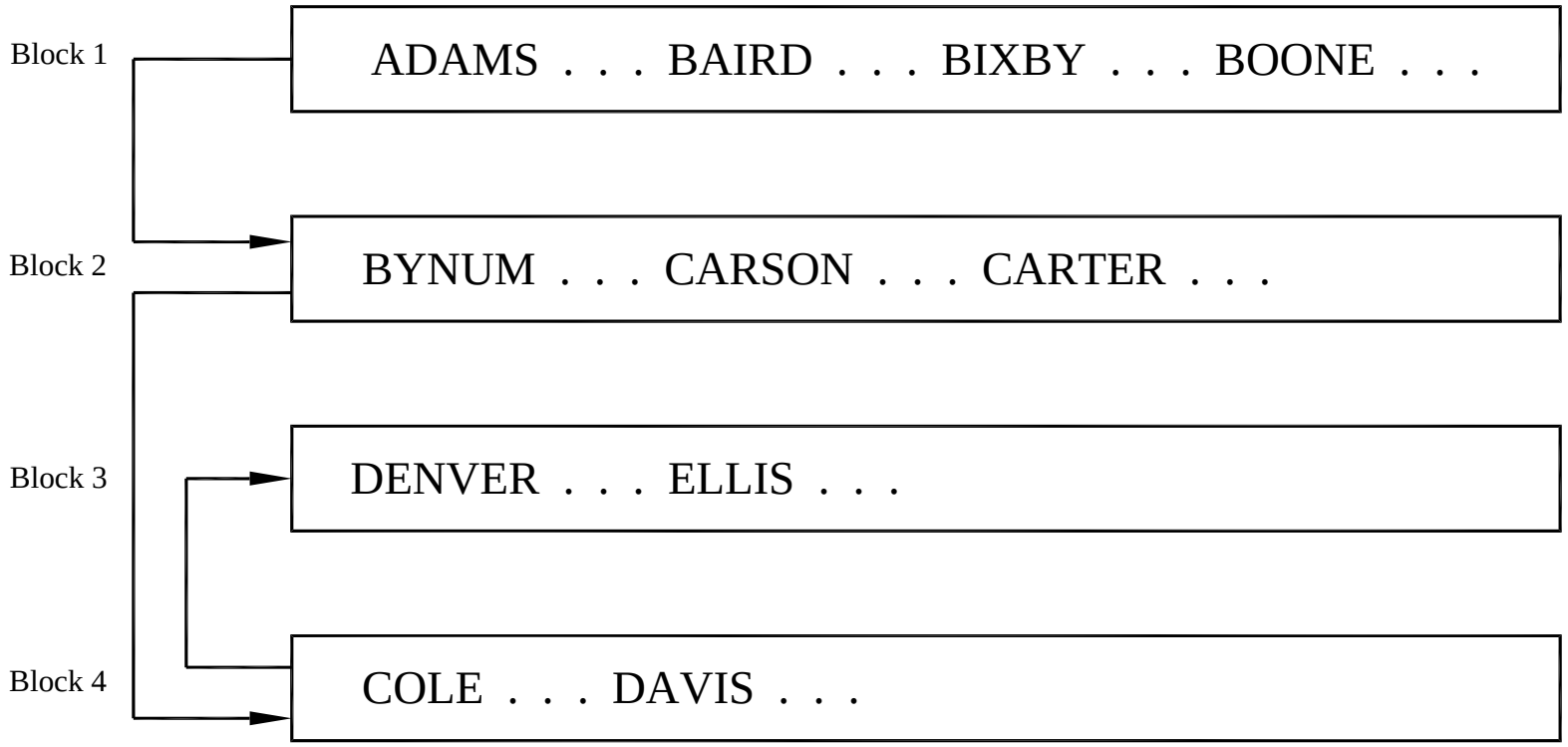
- Esta inserción causa que el bloque dos se divida, la segunda mitad de lo que se encontraba originalmente en el bloque 2, es colocado en el bloque 4 después de la partición

Ejemplo, cont...

- Contrario a los árboles B, aquí no existe una promoción

- Solamente se dividen los registros entre dos bloques y reorganizan las ligas de tal manera que se pueda mover a través del archivo ordenado por llaves bloque tras bloque

Ejemplo, cont...



Eliminación

- La eliminación de registro puede causar que un bloque tenga menos de la mitad de elementos y se ocasione un sub-flujo

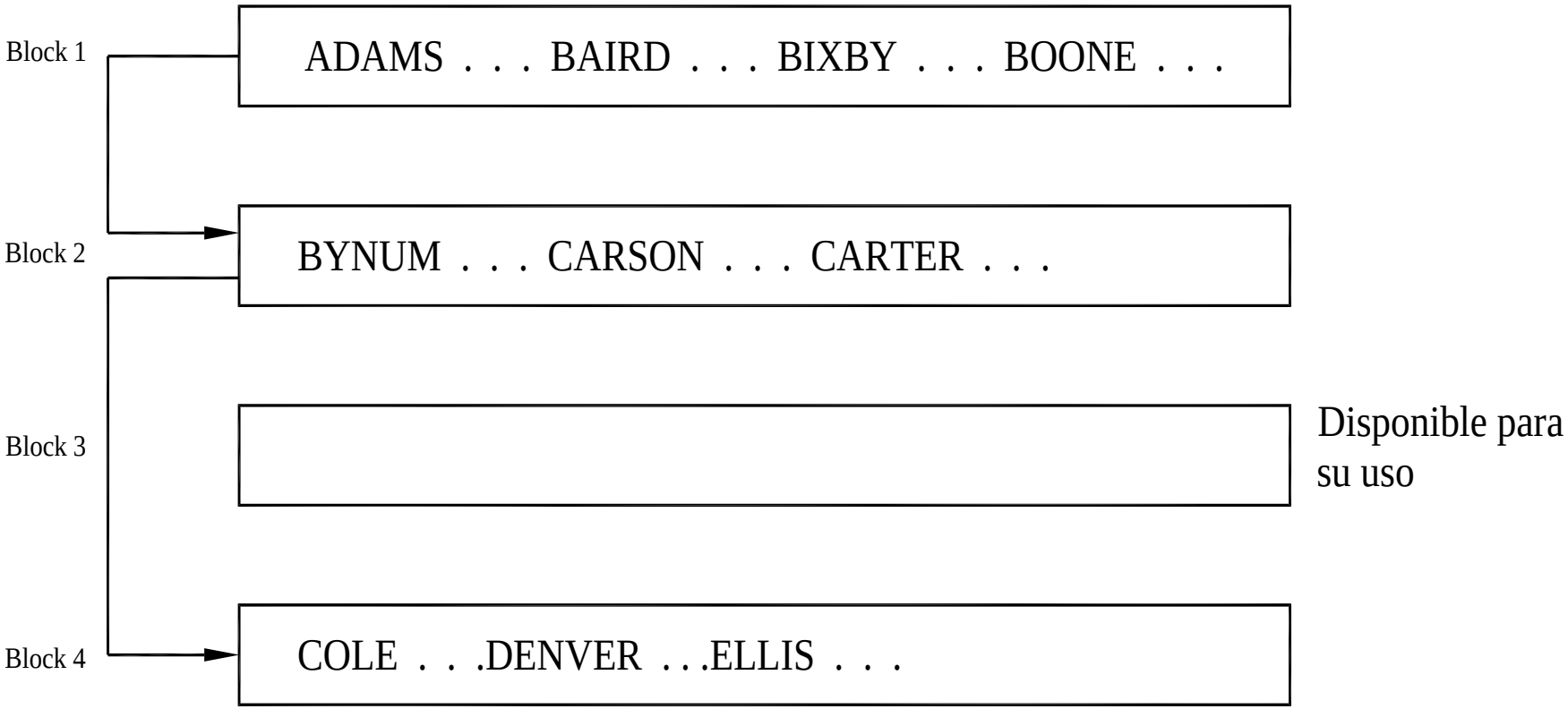
Manejo del sub-flujo

- El manejo es similar al de un árbol B:
  - Si un nodo vecino no está lleno, se pueden unir dos nodos liberando uno para su uso posterior
  - Si los nodos vecinos tienen más de la mitad de los elementos, se pueden redistribuir los registros entre los nodos para hacer la redistribución

Ejemplo

- Eliminar la llave DAVIS
  - El bloque 4 tiene un sub-flujo y después es mezclado con su sucesor en una secuencia lógica que es el bloque 3
  - El proceso de mezclado libera el bloque 3 para su uso posterior

Ejemplo, cont...



Costos

- Los costos asociados con este tipo de ordenamiento son:
  - Una vez que la inserción es realizada, se requiere más espacio que con archivo no manejado por bloque debido a la fragmentación dentro del bloque
  - El orden del registro no es necesariamente físicamente secuencial a través del archivo

Tamaños de bloque

Introducción

- Un bloque es la unidad básica para las operaciones de E/S
- Cuando se leen datos del disco, nunca se lee menos que un bloque
- Cuando se escriben datos, siempre se escribe menos de un bloque
- Un bloque es la máxima extensión física para una secuencia

Introducción

- Se debe pensar en términos de bloques grandes en donde cada uno contiene muchos registros

- La elección a realizar es elegir el límite que tendrá ese bloque

Consideración 1

- El tamaño del bloque debe ser tal que pueda contener una gran cantidad de bloques en memoria

- Cuando se parte o se mezcla un bloque, se desea mantener al menos dos bloques en memoria en cualquier momento

Consideración 2

- Leer o escribir un bloque no debe tomar mucho tiempo
- Incluso si se tiene una ilimitada cantidad de memoria, se desea poner un límite al tamaño del bloque
- De esta manera no se acaba leyendo el archivo entero para obtener un solo registro

Elegir el tamaño de un bloque

- Una sugerencia razonable para elegir el tamaño de un bloque es hacerlo del tamaño de un cluster

- La decisión final probablemente dependerá de diversos factores

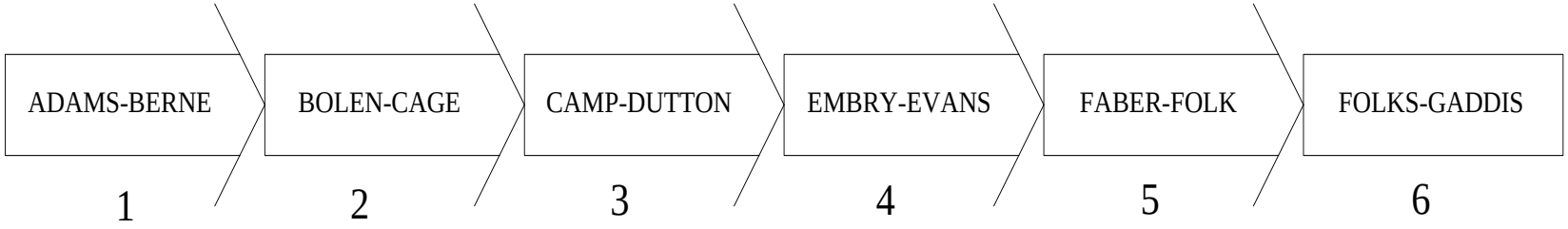
Añadir índices al conjunto secuencia

Añadir un índice al conjunto secuencia

- Se cuenta con un mecanismo para mantener un conjunto de registros de tal manera que se pueda acceder a ellos secuencialmente basados en el ordenamiento de sus llaves

Añadiendo índices

- Se puede ver cada uno de los bloques como un rango de registros como:



Representación

- Con esta representación se puede elegir el bloque que contendría la información requerida

- Por ejemplo: si se busca la llave BURNS se encontraría en el segundo bloque

Representación

- Es fácil construir un índice simple para este tipo de bloques
- Se debe elegir construir un índice de tamaño fijo que contenga la llave del último registro en cada bloque

Representación

- La combinación de este tipo de índices con el conjunto secuencia de bloques provee un acceso secuencial completo:
  - Si se requiere recuperar un registro específico se consulta el índice y posteriormente se recupera el bloque correspondiente
  - Si se quiere un acceso secuencial, se comienza con el primer bloque y se lee a través de las listas ligadas de los bloque hasta que se hayan leído todas ellas

Índice en memoria

- El índice debe estar en memoria por dos razones:
  - Se encuentra el registro específico por medio de una búsqueda binaria en el índice
  - Como los bloques en el conjunto cambian debido a la separación, mezcla y redistribución, el índice tiene que ser actualizado, actualizar un índice simple de este tipo funciona bien si el índice es relativamente pequeño y está almacenado en memoria

Índice muy grande

- Si el índice es muy grande para almacenarse en memoria:
  - Los árboles B son una estructura adecuada para el manejo de índices que no pueden almacenarse en memoria
  - El uso de un árbol B para la secuencia de bloques, forma una estructura híbrida conocida como árbol B+

Contenido de índice: separadores en lugar de llaves

Contenido del índice

- El propósito de un índice es auxiliar cuando se estén buscando registros con una llave específica

- El índice debe conducir al bloque en el conjunto secuencia que contiene al registro

Uso de separadores

- El contenido del índice solo si conduce al bloque correcto en el conjunto secuencia

- El índice no contiene esta información, solo contiene el lugar en donde se puede obtener la información

Uso de separadores

- Dado esta vista del índice como un mapa, es fácil reconocer que no se necesitan las llaves en el índice, lo que se necesitan son separadores

- Existen muchos potenciales separadores capaces de distinguir entre dos bloques

Separadores

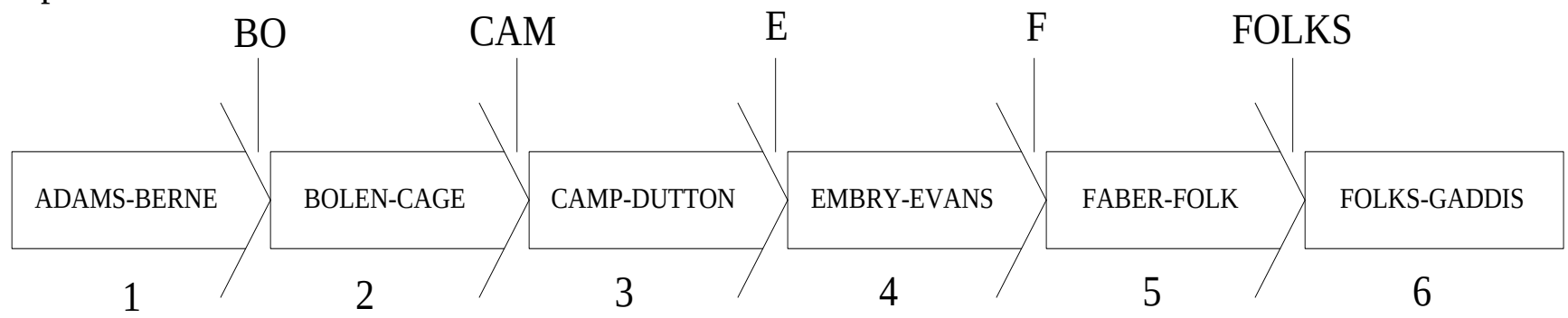
- Si una comparación entre la llave que se está buscando y cualquiera de los separadores muestra que la llave precede al separador, se busca en el bloque izquierdo, si la llave es mayor que el separador se continúan realizando comparaciones en los bloques a la derecha

Elección de separadores

- Si se desean manejar los separadores como entidades de longitud variable se puede ahorrar espacio utilizando el *separador más corto*

Ejemplo

Separadores



- Por lo tanto, si se utilizan los separadores como un mapa para el conjunto secuencia, se debe decidir si se recupera el bloque de la derecha del separador o el de la izquierda de acuerdo a la siguiente regla:

Relación de llave y separador	Decisión
Llave < separador	Izquierda
Llave = separador	Derecha
Llave > separador	Derecha

Árboles B

Introducción

- Actualmente es difícil pensar en un sistema de archivos que no esté formado por árboles

- El objetivo del desarrollo de este tipo de estructuras fue diseñar un método general para almacenar y recuperar datos en grandes archivos que proveyeran acceso rápido al dato con el costo mínimo asociado al movimiento de la cabeza

Árboles con índices

- El problema fundamental de mantener un índice en un medio de almacenamiento secundario es que acceder a este es muy lento

Problemas con los índices

- La búsqueda en el índice debe ser más rápida que una búsqueda binaria
- Buscar una llave en un disco en ocasiones involucra buscar en diferentes pistas
- Una búsqueda en ocasiones debe examinar en más de tres o cuatro localidades antes de encontrar la llave

Problemas con los índices

- La inserción y eliminación deben ser tan rápidas como la búsqueda
- Si el insertar una llave en un índice involucra mover una gran cantidad de otras llaves en el mismo, el mantenimiento es considerado muy impráctico en un medio de almacenamiento secundario que consista en solo unos cientos de llaves
- Se debe encontrar una manera de realizar inserciones y eliminaciones que tengan efectos solamente locales en el índice en lugar de que se requiera una organización masiva

Indexado con ABB y AVL

Índices con ABB

- Dado una lista ordenada, se puede expresar una búsqueda binaria en la lista utilizando un ABB
- La búsqueda binaria no es lo suficientemente rápida para un índice que se encuentre en un disco
- El principal problema es la ausencia de una estrategia efectiva de balancear el árbol

Índices con árboles AVL

- Los árboles AVL garantizan que se tendrá un costo mínimo en las operaciones de búsqueda

- Mantener un árbol en su forma de AVL mientras se insertan nuevos nodos, involucra el uso de uno de cuatro conjuntos de rotaciones posibles

Índices con árboles AVL

- Los árboles AVL no son por si mismos directamente aplicables a la mayoría de las estructuras de archivos debido a que como todos los árboles binarios pueden llegar a tener muchos niveles

Índice con árboles AVL

- Cuando se utilizan medios de almacenamiento secundario, un procedimiento requiera más de cinco o seis búsquedas para encontrar una llave es poco menos que deseable y veinte o más es inaceptable
- Esto regresa los dos problemas que previamente se habían identificado:
  - La búsqueda binaria requiere muchas búsquedas
  - Mantener un índice ordenado es costoso

Árboles con páginas binarias

Introducción

- El uso de ABB es muy ineficiente, leer un nodo binario gasta mucho del tiempo de lectura de datos en un disco

- Dado que esta lectura es el factor crítico en el costo de la búsqueda, no se puede permitir este desperdicio de tiempo

Propiedades

- Un árbol de páginas binarias intenta solucionar este problema localizando nodos binarios múltiples en la misma página del disco

- No se utiliza el tiempo en localizar solo una cantidad de bytes, sino que se lee una página completa del archivo

Propiedades

- Esta página consiste en una gran cantidad de registros individuales

- Si el siguiente bit de información que se necesita del disco se encuentra en la página que se está leyendo, se ha ahorrado el costo de una nueva búsqueda en el disco

Propiedades

- Separar el árbol en páginas permite realizar una búsqueda más rápida en un medio de almacenamiento secundario

Ejemplo:

- Si se utiliza un tamaño de página de 8KBytes capaz de contener 511 pares de llave/referencia
- Se tienen 134,217,727 llaves

  - Utilizar un árbol binario balanceado
  - Utilizar un árbol de paginación

Ejemplo:

- Utilizando un árbol binario balanceado:

$$\log_2(N+1)$$

$$\log_2(134,217,727+1)$$

- Se necesitan 27 búsquedas

Ejemplo:

- Utilizando un árbol de páginas:

$$\log_{k+1}(N+1)$$

$$\log_{k+1}(134,217,727+1)$$

- k = número de pares de llave/referencia que puede contener el árbol
- Se necesitan 3 búsquedas

Problemas

- El principal problema con los árboles de páginas es la ineficiencia en el uso del espacio en disco

- Para que sea eficiente, las llaves deben estar ordenadas y la raíz deberá ser la llave que se encuentre a la mitad del conjunto ordenado

Multi indexado
Árboles B

Introducción

- Necesidad de un método que permitiera almacenar y recuperar datos en sistemas con archivos de gran tamaño

- R. Bayer y E. McCreight que trabajaban para la corporación Boeing

¿Árboles B?

- El nombre de árboles B aún ni ha sido bien establecido:
  - Balanceado
  - *Broad* (Amplio)
  - Boeing
  - Bayer

Propiedades

- Los árboles-B son índices multi-nivel que solucionan el problema del costo de la inserción y eliminación

- Cada nodo de un árbol B es un registro de índices, cada uno de estos registros tiene el mismo número de pares llave-referencia, llamado el orden del árbol B

Propiedades

- El registro también tiene un número mínimo de pares llave-referencia, típicamente la mitad del orden

- La única excepción es la raíz que puede tener un mínimo de dos llaves

Nomenclatura

- El orden de un árbol B se refiere al máximo número de descendientes que una página puede tener

- Cuando se parte la página de un árbol B, los descendientes son divididos entre la nueva y la antigua página, por consecuencia cada página excepto la raíz y las hojas tienen al menos $m/2$ descendientes

- Expresados en términos de una función techo, se puede decir que el mínimo número de descendentes es $[m/2]$

Definición formal

- Una definición más formal de un árbol B de orden *m* es :
  - Cada página tiene un máximo de *m* descendientes
  - Cada página, excepto por la raíz y las hojas, tiene al menos [*m*/*2*] descendientes
  - La raíz tiene al menos dos descendientes
  - Todas las hojas aparecen en el mismo nivel
  - El nivel hoja forma un completo y ordenado índice de los archivos de datos asociados

Inserción

- Insertar una nueva llave en un índice de registros que no este lleno es barato

- Actualizar el contenido del registro, si la nueva llave es el valor más grande en el índice de registros, se deben actualizar los nodos superiores

Inserción

- Cuando una inserción en un árbol causa que el registro tenga un sobre flujo, se divide en dos registros, cada uno con la mitad de las llaves

- Dado que un nuevo índice ha sido creado en este nivel, la nueva llave debe ser insertada en el siguiente nodo con un nivel más alto, a esto se la llama promoción de una llave

Inserción

- Esta promoción puede causar un desbordamiento en ese nivel
- Esto ocasiona una división de los nodos y una promoción de una llave al siguiente nivel
- Si el registro de índice en el nivel más alto sufre un sobre flujo debe dividirse, ocasiona que se añada otro nivel al índice multiniveles

Inserción

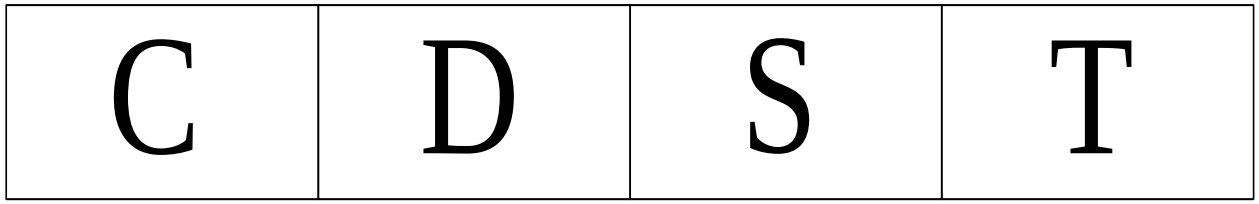
- Existen dos aspectos importantes que se pueden revisar acerca de la inserción, particionamiento y proceso de promoción:
  - Comienza con una búsqueda que se realiza hasta el nivel de hojas
  - Después de encontrar el lugar de la inserción en el nivel de hoja, el trabajo de inserción, detección de sobre flujo, y particionamiento procede hacia arriba partiendo del fondo

Inserción

- Se puede concebir como un procedimiento iterativo que consta de tres fases:
  - Búsqueda en el nivel de hoja antes de la iteración
  - Inserción, sobre flujo y partición hacia los niveles superiores
  - Creación de una nueva raíz, si la raíz fue dividida

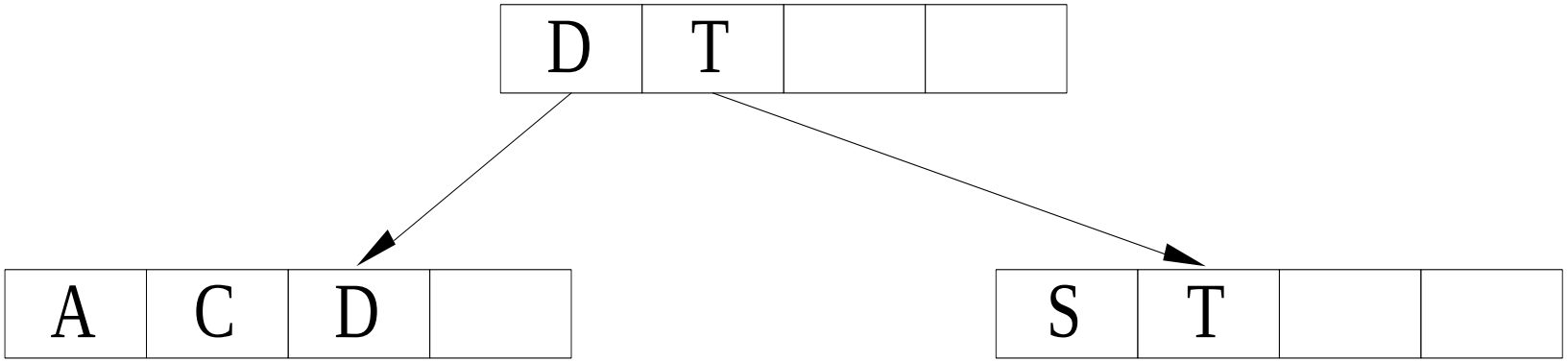
Creación de un árbol B.

- Insertar: C, S, D, T, A, M, P, I, B, W, N, G, U, R, K, E, H, O, L, J, Y, Q, Z, F, X, V
- Insertar C, D, S, T



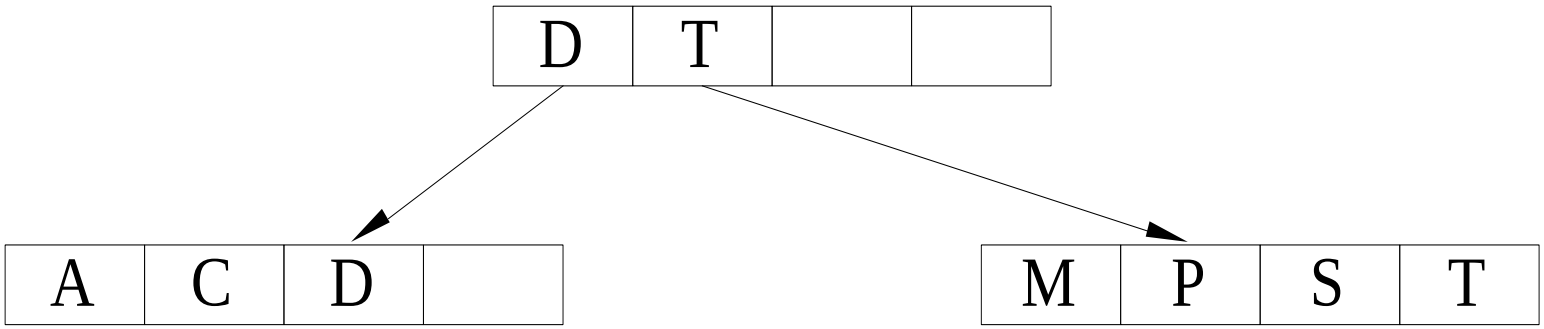
Ejemplo, cont...

- Insertar A



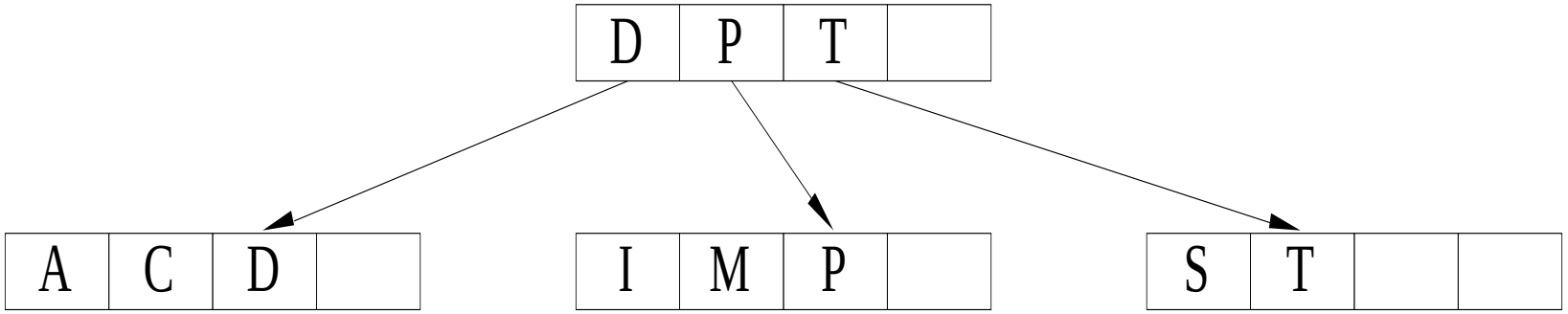
Ejemplo, cont...

- Insertar M y P



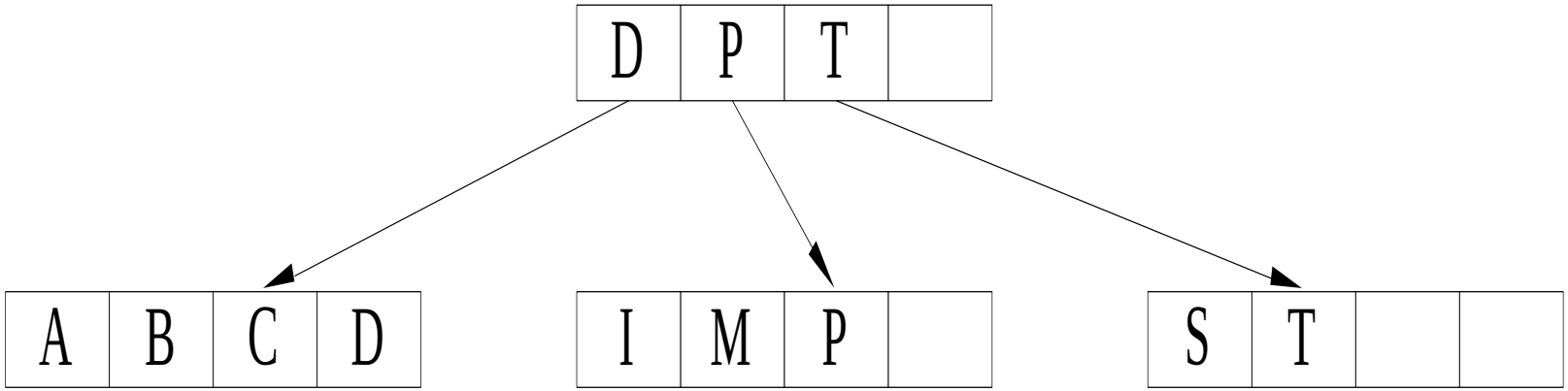
Ejemplo, cont...

- Insertar I



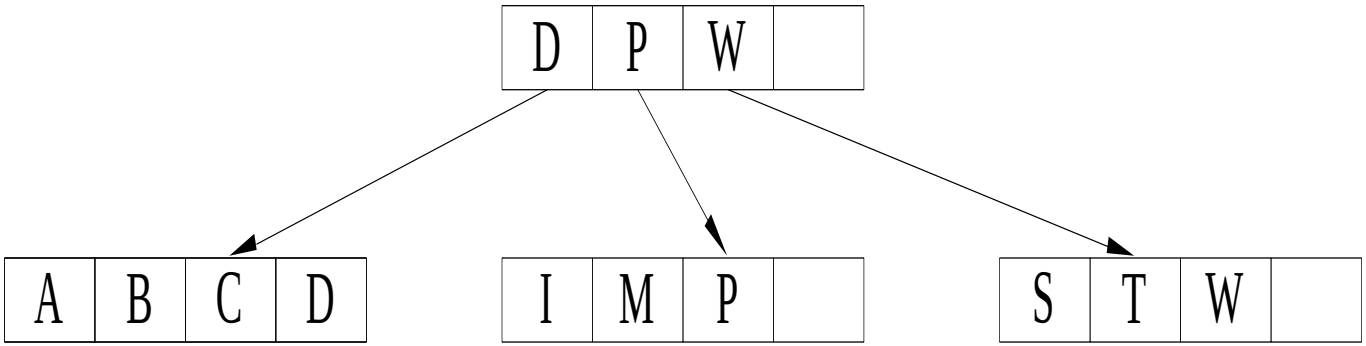
Ejemplo, cont...

- Insertar B



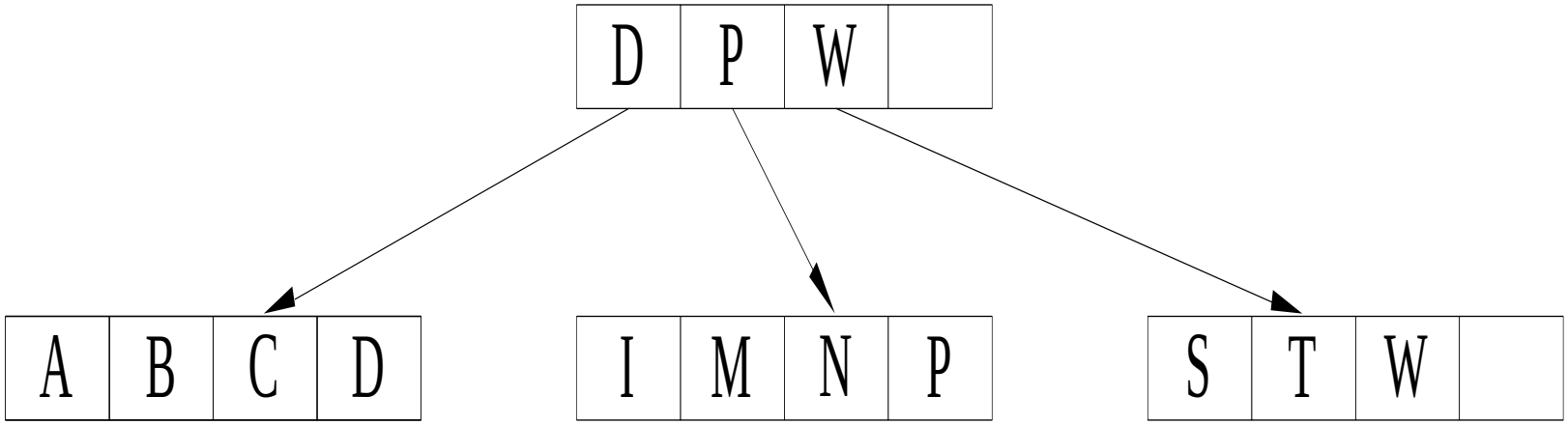
Ejemplo, cont...

- Insertar W



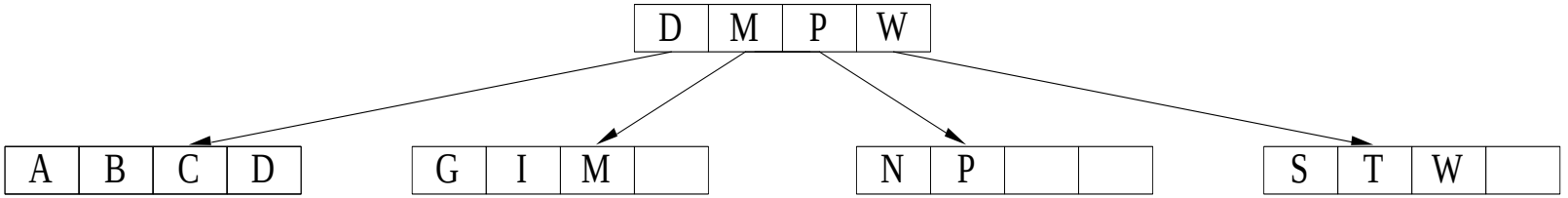
Ejemplo, cont...

- Insertar N



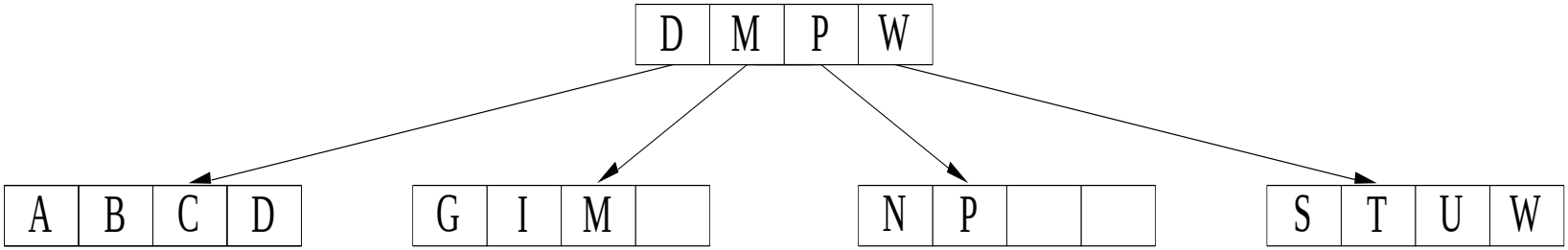
Ejemplo, cont...

- Insertar G



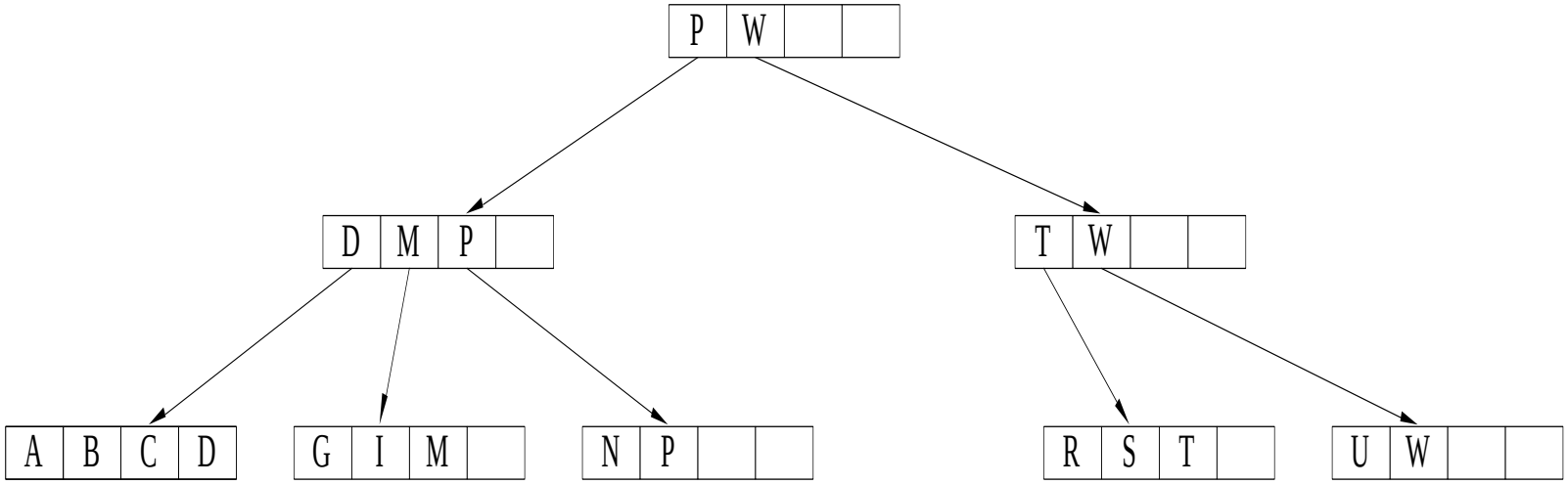
Ejemplo, cont...

- Insertar U



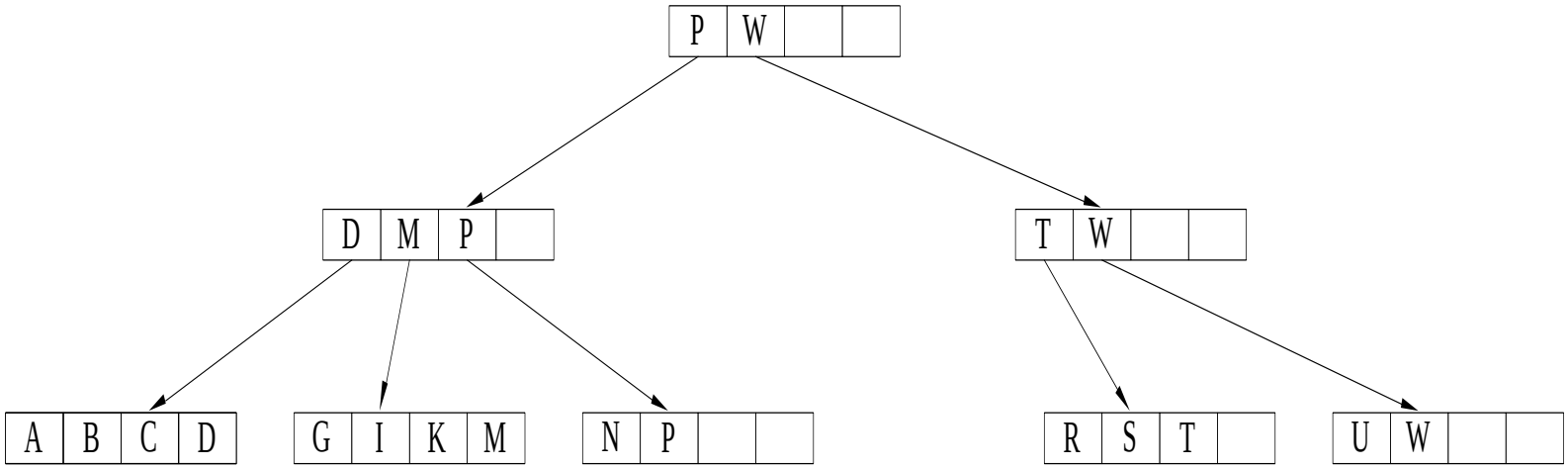
Ejemplo, cont...

- Insertar R



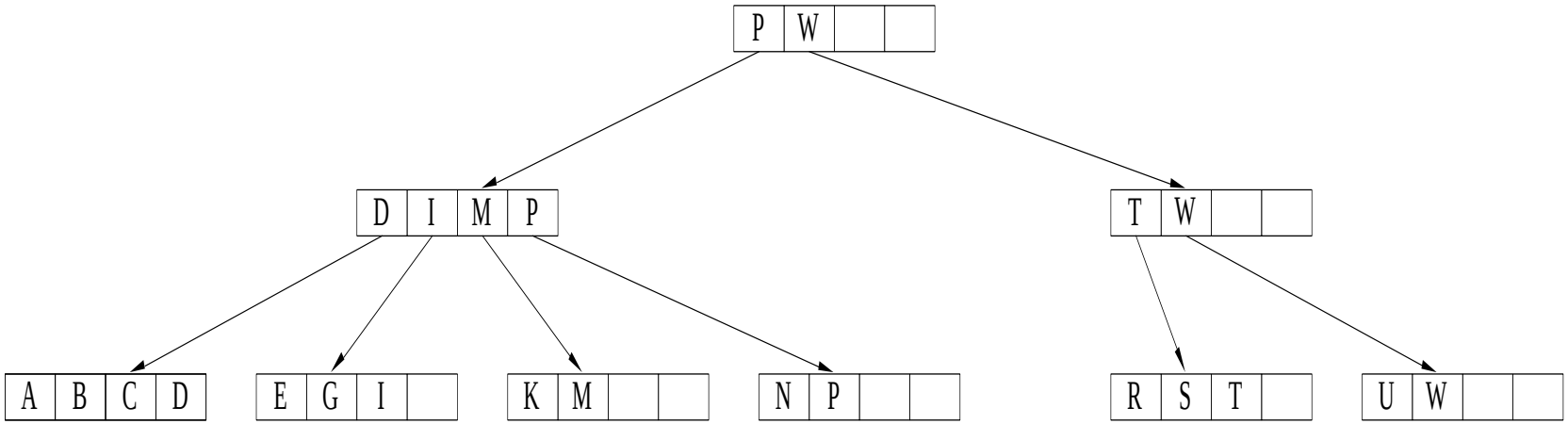
Ejemplo, cont...

- Insertar K



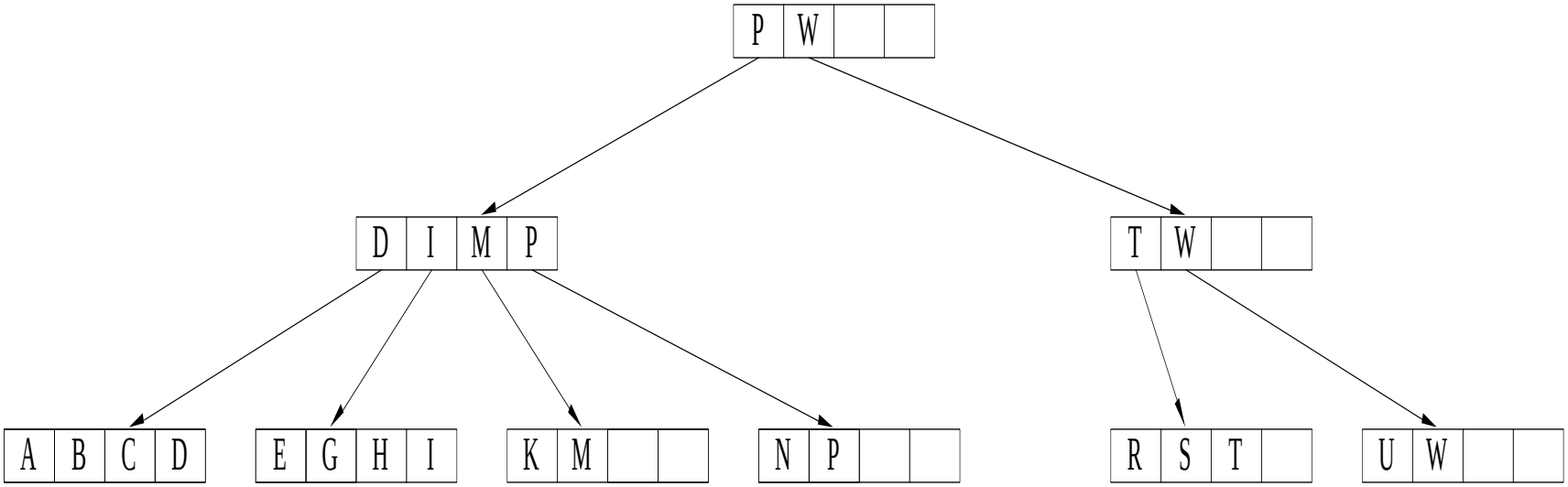
Ejemplo, cont...

- Insertar E



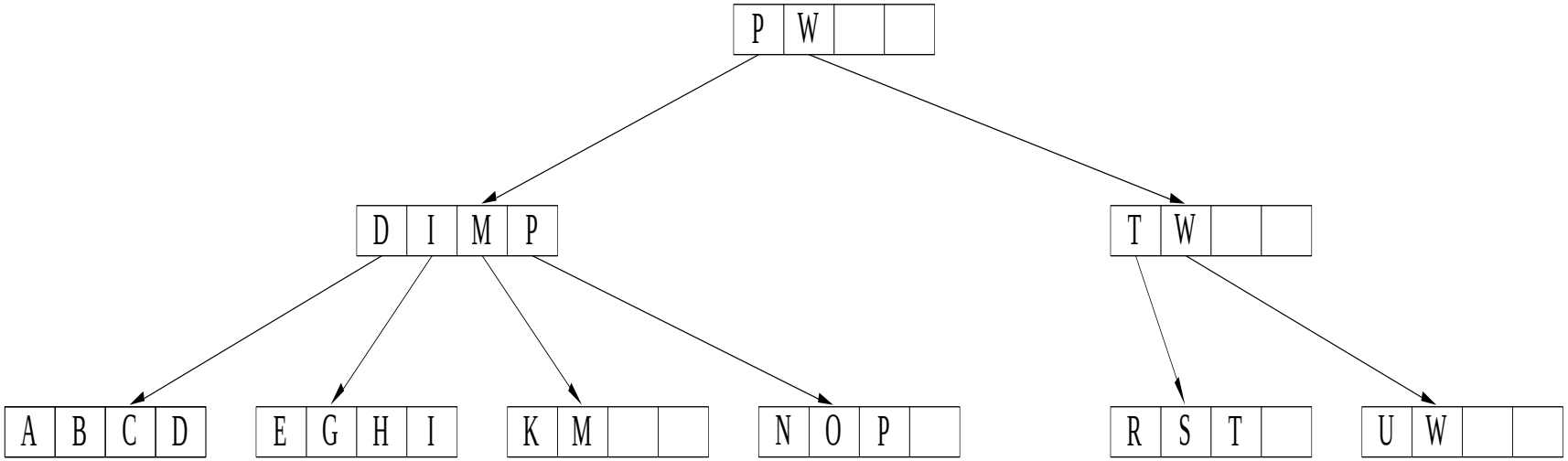
Ejemplo, cont...

- Insertar H



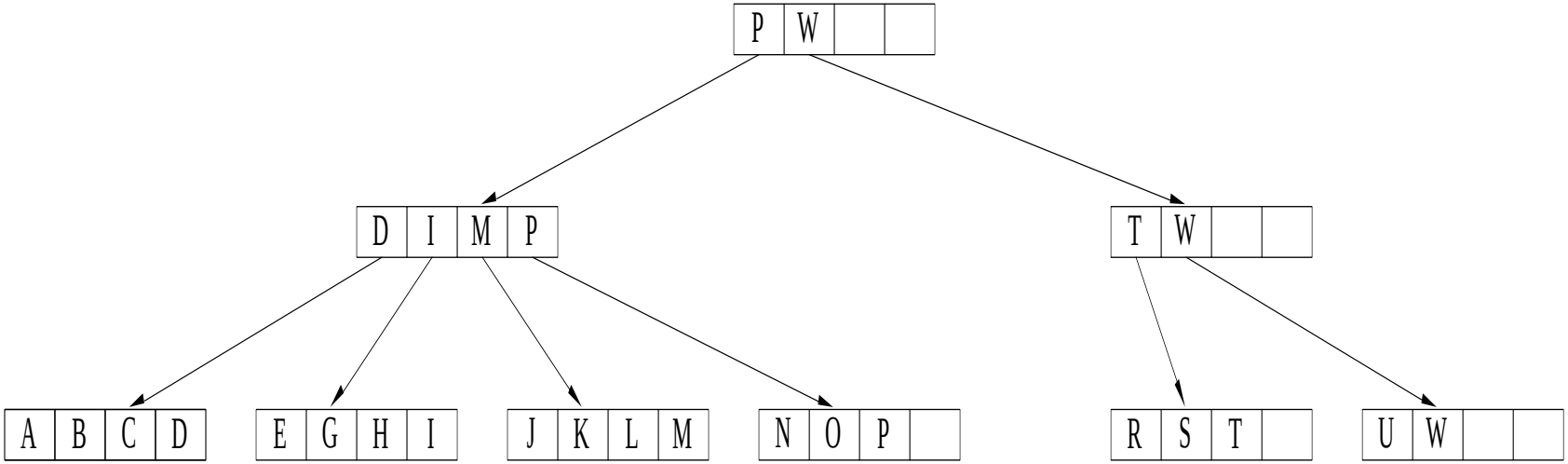
Ejemplo, cont...

- Insertar O



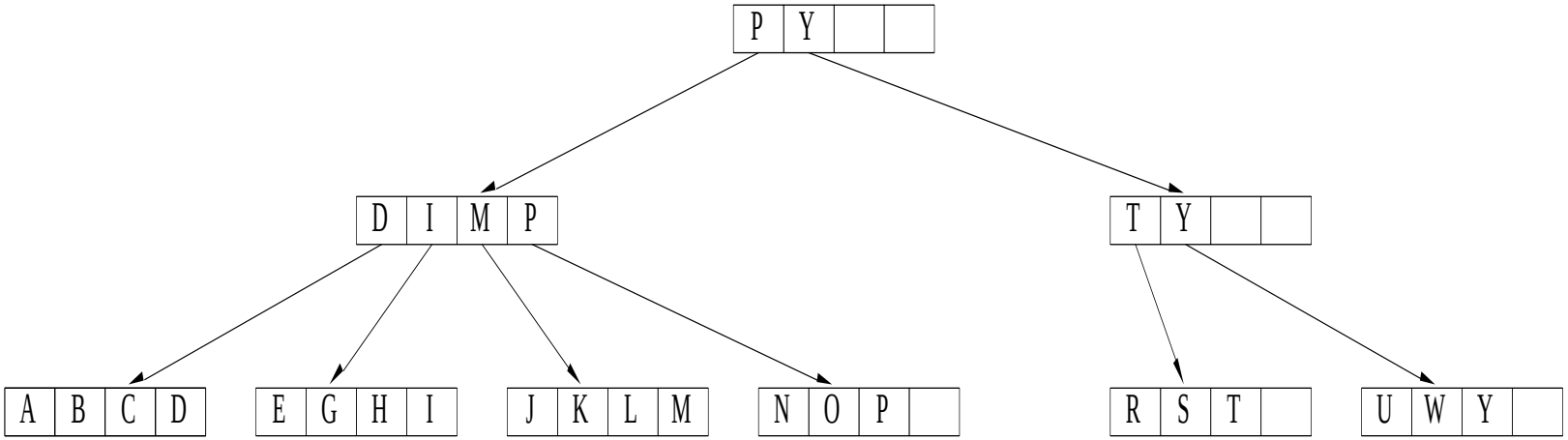
Ejemplo, cont...

- Insertar L y J



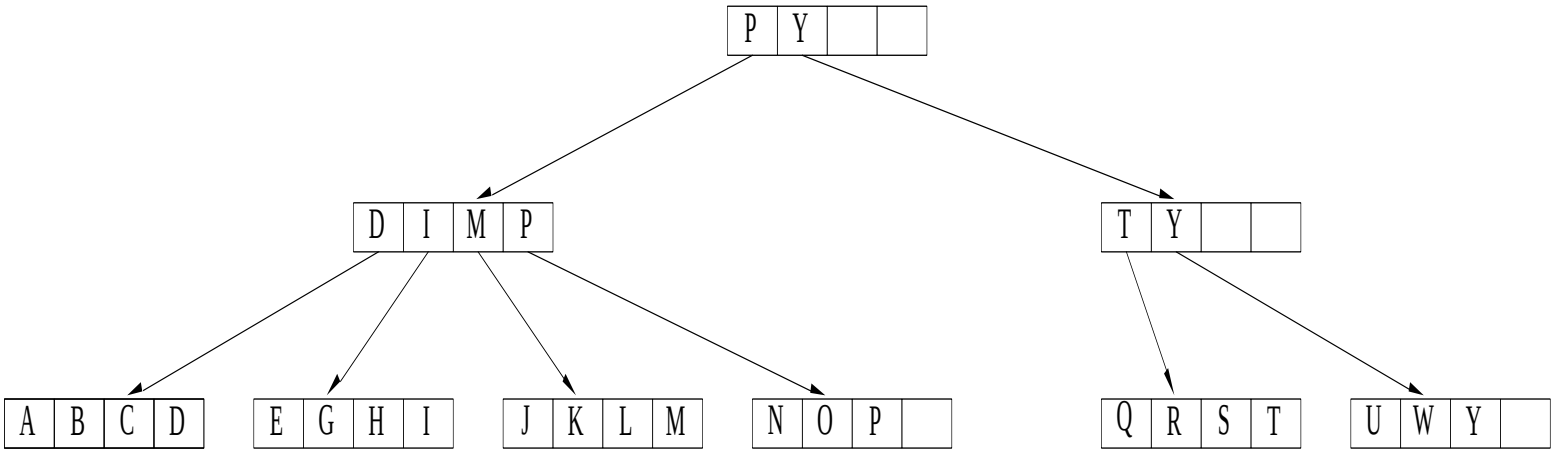
Ejemplo, cont...

- Insertar Y



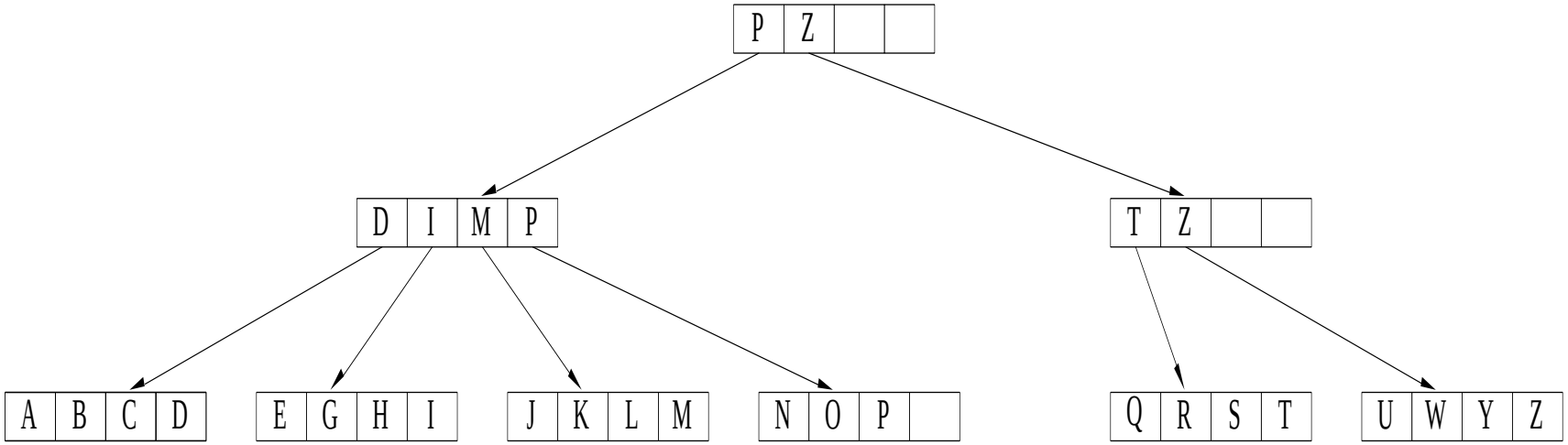
Ejemplo, cont...

- Insertar Q



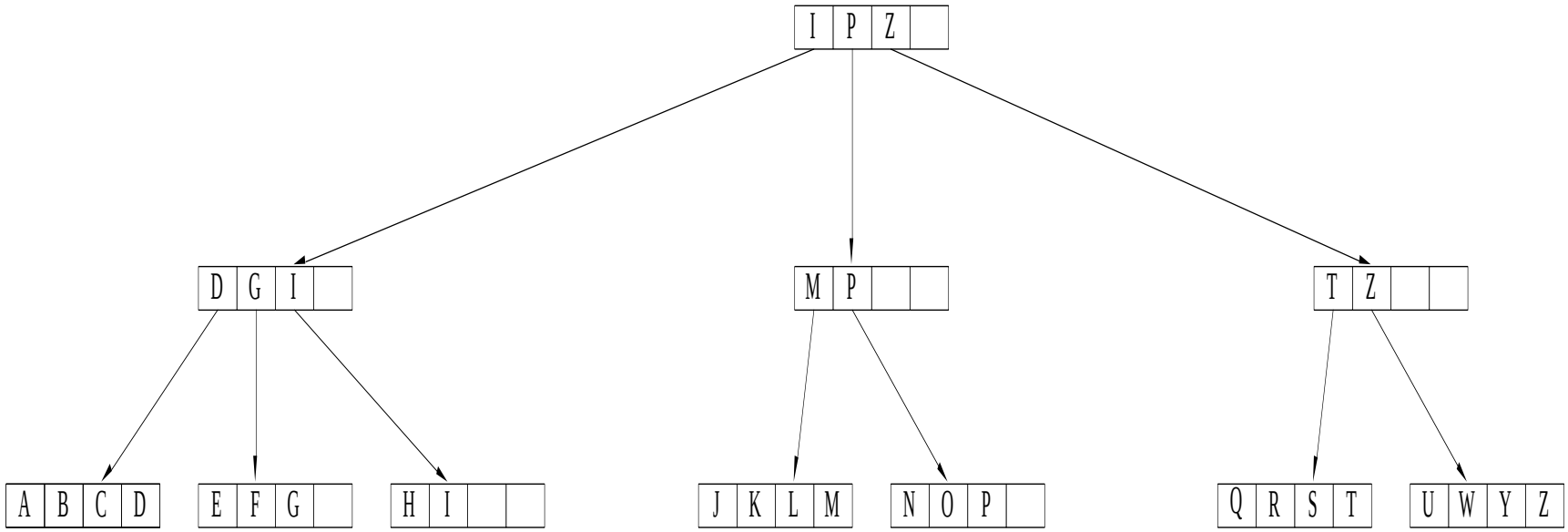
Ejemplo, cont...

- Insertar Z



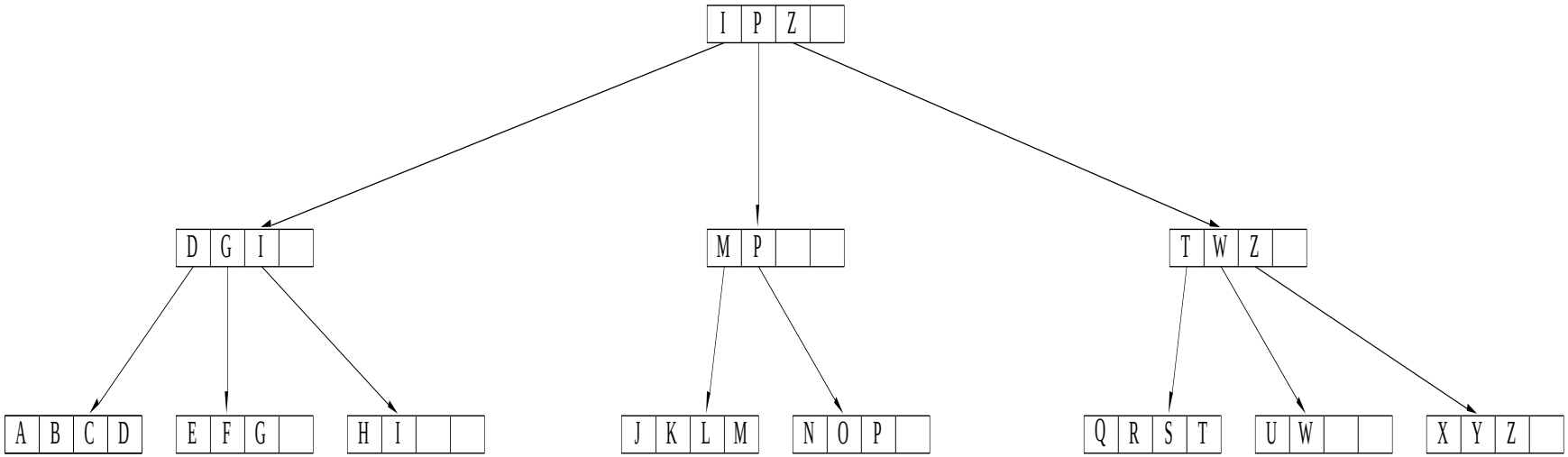
Ejemplo, cont...

- Insertar F



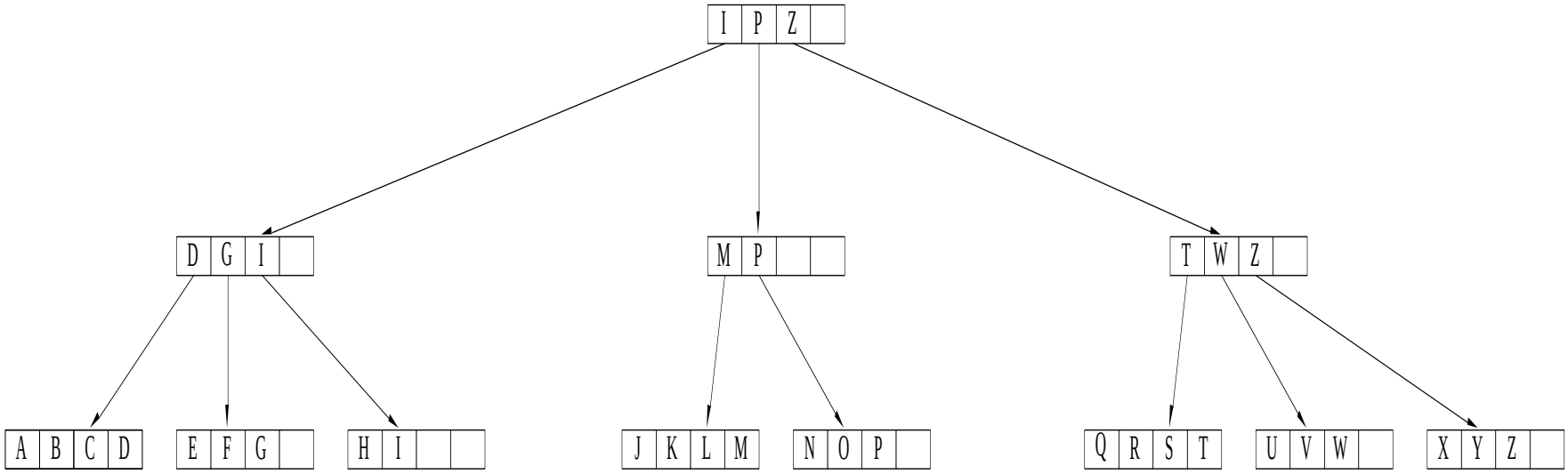
Ejemplo, cont...

- Insertar X



Ejemplo, cont...

- Insertar V



Archivo de índices

De esta manera, una posible representación del archivo de índices podría quedar de la siguiente manera:

30								
30	35	I	50	P	65	Z		
35	40	D	45	G	48	I		
50	52	M	57	P				
65	69	T	72	W	80	Z		
40	14	A	8	B	2	C	7	D
45	23	E	1	F	20	G		
48	15	H	9	I				
52	18	J	3	K	13	L	24	M
57	21	N	5	O	17	P		
69	25	Q	11	R	6	S	22	T
72	12	U	10	V	26	W		
80	16	X	4	Y	19	Z		

Donde:

	Representan referencias (*offset*) en el archivo de índices
	Representan referencias (*offset*) en el archivo de datos
	Representan posiciones en el archivo de índices

Búsqueda

- La búsqueda es relativamente simple pero ilustra algunos de los aspectos más importantes de un árbol B.
  - Es iterativa
  - Trabaja en dos etapas, opera alternativamente en páginas enteras y dentro de las páginas

Búsqueda

- El procedimiento de búsqueda es iterativo, cargando una página en memoria y después buscando a través de la página, buscando por una llave en los niveles más bajos hasta que encuentran el nivel de hojas

Archivo de índices

30								
30	35	I	50	P	65	Z		
35	40	D	45	G	48	I		
50	52	M	57	P				
65	69	T	72	W	80	Z		
40	14	A	8	B	2	C	7	D
45	23	E	1	F	20	G		
48	15	H	9	I				
52	18	J	3	K	13	L	24	M
57	21	N	5	O	17	P		
69	25	Q	11	R	6	S	22	T
72	12	U	10	V	26	W		
80	16	X	4	Y	19	Z		

Donde:

	Representan referencias (*offset*) en el archivo de índices
	Representan referencias (*offset*) en el archivo de datos
	Representan posiciones en el archivo de índices

Ejemplo

- Búsqueda de la información con la llave “L”:
  - Se abre el archivo de índices.
  - Se busca la posición de la raíz (IPZ)
  - Se compara con I, como L es mayor se compara con P, como es menor se va hacia su referencia (50)
  - Se carga el registro con las llaves (MP)
  - Se compara con M, como es menor se va hacia la referencia que indica el valor de llave M (52)
  - Se carga el registro, como es el último nivel (de hojas), se busca el elemento L, al encontrarlo se obtiene la referencia (13) que indica en donde está la información en el archivo de dato que corresponde a la llave L

Eliminación, mezcla y redistribución.

Reglas para la eliminación

- Las reglas para eliminar una llave *k* de un registro *n* del archivo de índices son:
  - Si *n* tiene más del mínimo de llaves y *k* no es la mayor en *n*, simplemente se elimina.
  - Si *n* tiene más del mínimo número de llaves y *k* es la mayor en *n*, eliminar *k* y modificar los niveles de índices más elevados para reflejar el nuevo valor de la llave más grande en *n*.

Reglas para la eliminación

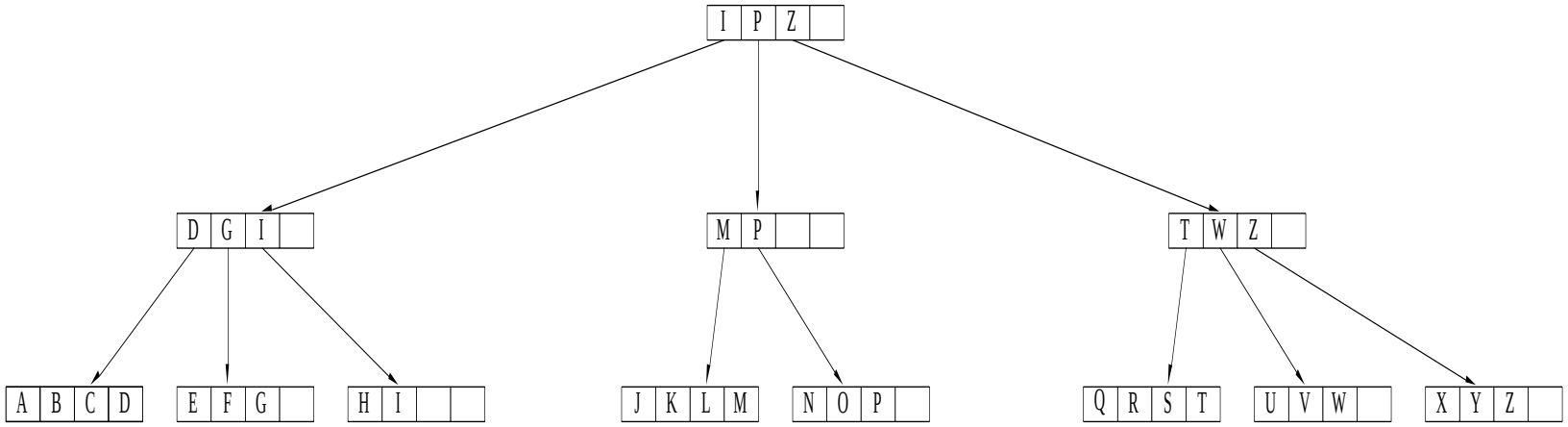
- Si *n* tiene exactamente el mínimo número de llaves y uno de los registros que representan a sus hermanos tiene espacio para más llaves, mezclar *n* con sus hermanos y eliminar una llave del nodo padre

- Si *n* tiene exactamente el mínimo número de llaves y uno de los hermanos de *n* tiene llaves extra, redistribuirlas moviendo las llaves de un hermano a *n* y modificar los índices en los niveles más altos para que se reflejen los cambios en los nodos afectados

Eliminación

- Existen varios casos:
  - La eliminación no causa subflujo ni modificación en los nodos superiores
  - La eliminación no causa un subflujo pero si una modificación en los nodos superiores
  - La eliminación causa un subflujo

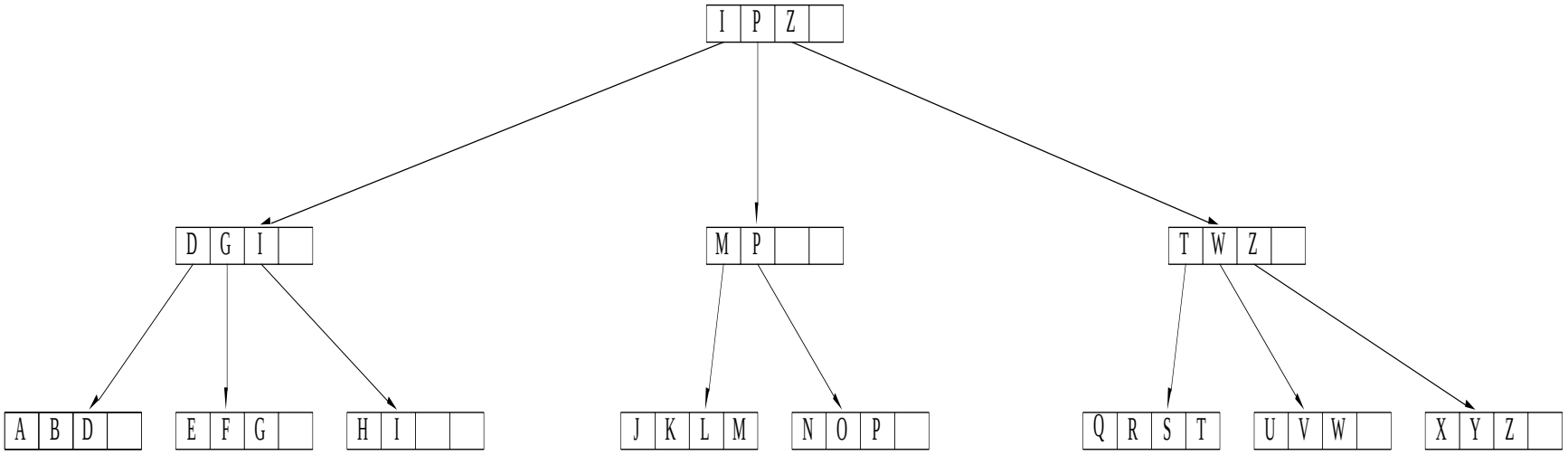
Ejemplo



Caso 1

- La eliminación más simple es ilustrada en la eliminación de la llave “C”
- Eliminar la llave de la primera hoja no causa un subflujo en el nodo y no cambia su valor más grande, por consecuencia la eliminación involucra solo el eliminar la llave del nodo

Eliminar C



Caso 2

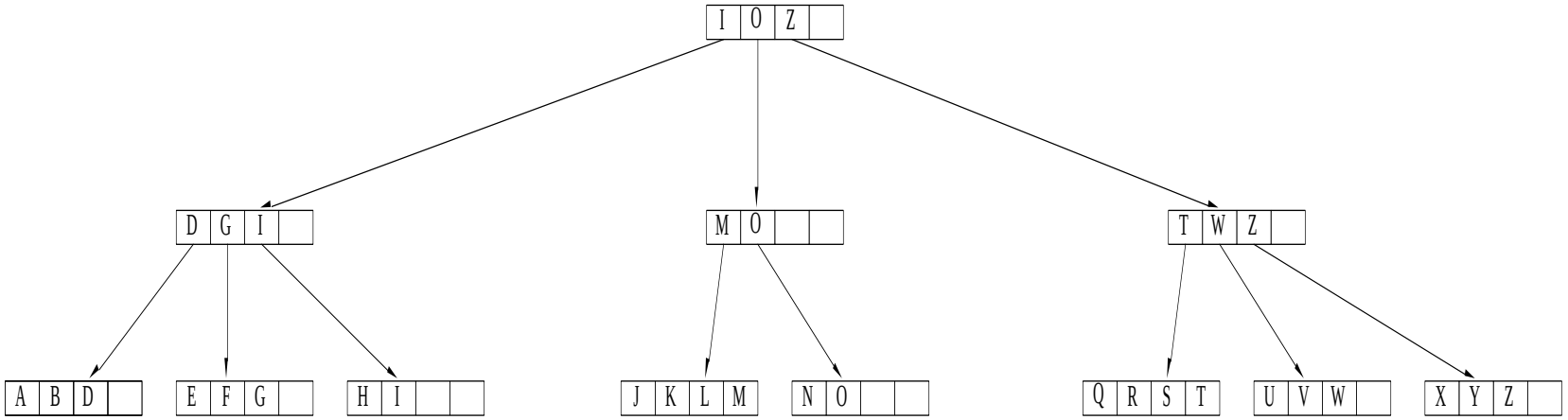
- Eliminar “P” es más complicado, remover “P” del nodo de la segunda hoja no causa un subflujo, pero cambia el valor de la llave más grande en el nodo, por lo tanto los nodos de los niveles superiores deben ser modificados para reflejar este cambio

Caso 2

- La mayor llave en el nodo de la hoja se convierte en “O” y el nodo del segundo nivel debe ser modificado para que contenga “O” en lugar de “P”

- Dado que “P” era la llave más larga en el segundo nodo en el segundo nivel del nodo, el nodo raíz también debe ser modificado para cambiar la “P” con el “O”

Eliminar P

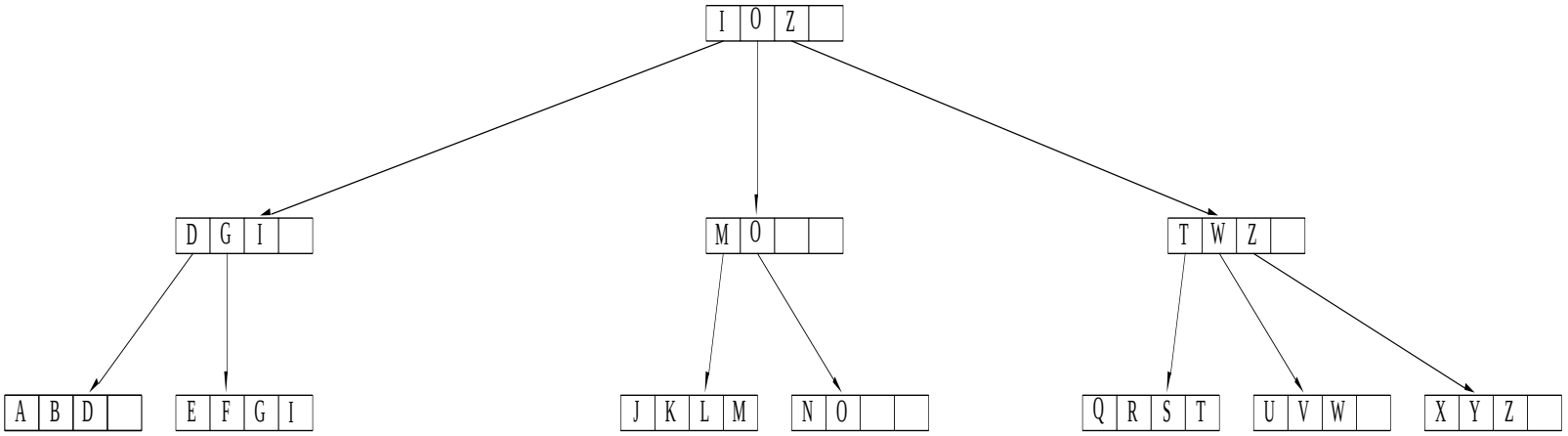


Caso 3

- Eliminar “H” causa un subflujo en el nodo de la tercera hoja, después de que H es eliminada, la única llave que permanece en el nodo, la llave “I” es insertada en el nodo vecino y la tercer hoja es eliminada.

- Dado que la segunda hoja tiene solo tres llaves, hay espacio para la llave “I” en el nodo, esto ilustra la operación de mezcla, después del mezclado el segundo nivel es modificado para reflejar el estado actual de los nodos hojas

Eliminar H



Redistribución

- Es una nueva idea en el manejo de la información en un árbol B

- La redistribución no ocasiona que la colección de nodos en el árbol sea modificada

- Con la redistribución no hay que tener en cuenta como se reorganizan las llaves

Redistribución

- La redistribución trabaja solo con nodos que son del mismo padre

- No realiza movimientos entre nodos que no sean del mismo padre aunque estos sean lógicamente adyacentes

Redistribución durante la inserción.

- La inserción no utiliza una operación como la redistribución
- La redistribución es una manera de evitar o al menos posponer la creación de nuevas páginas
- En lugar de dividir una página completa, la redistribución permite colocar las llaves que causan el sobre flujo en otras posiciones

Redistribución en la inserción

- El uso de la redistribución en lugar de la separación hace más eficiente al árbol B en términos de espacio utilizado

- Cuando un nodo se divide, la mitad de este queda vacío, dando en el peor de los casos un 50% de uso del árbol

Redistribución en la inserción

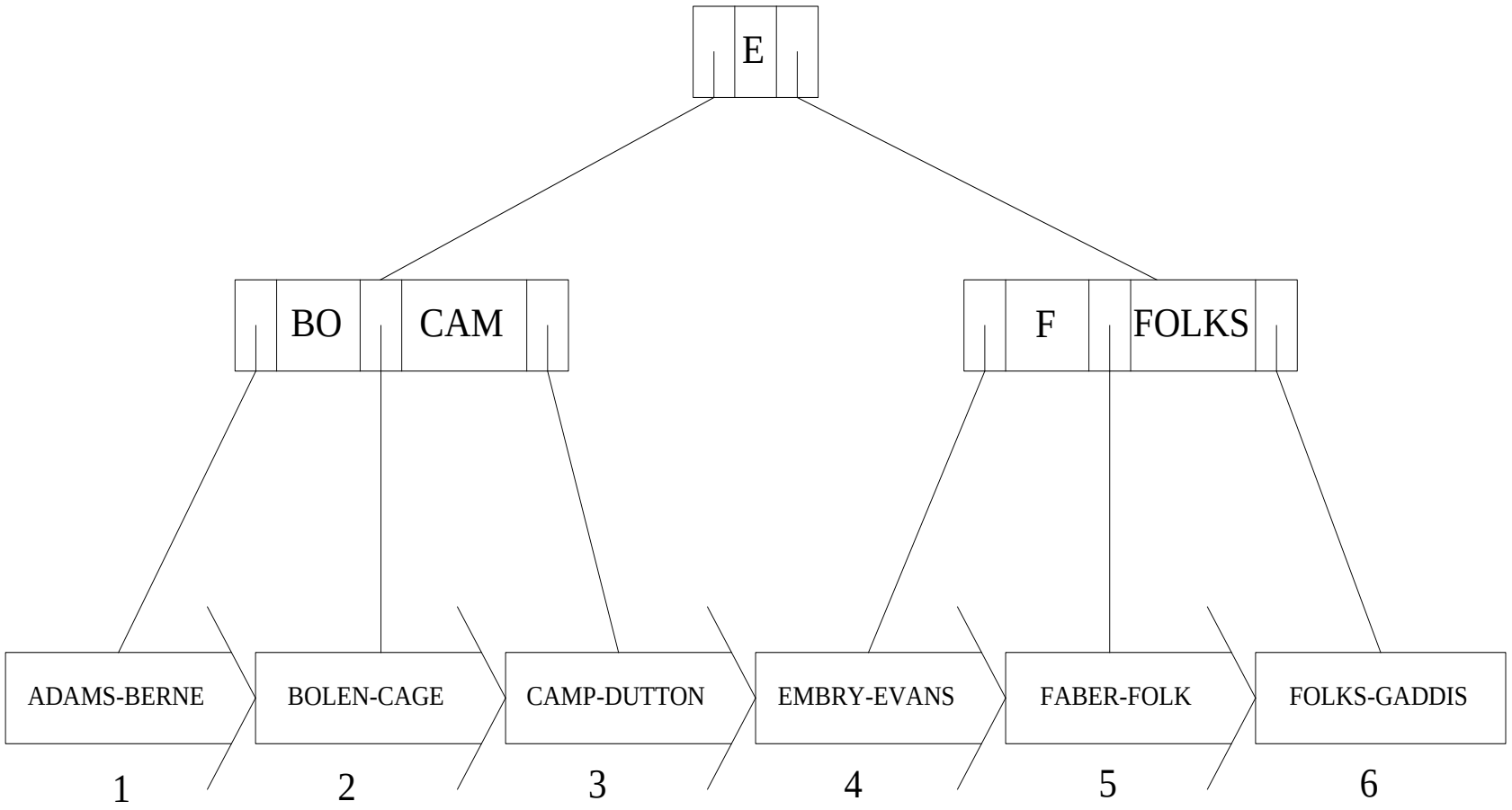
- Utilizar la redistribución cuando es posible en lugar de la división
- Dividir una página solo cuando los hermanos estén llenos
- Pruebas demostraron que esta técnica proporciona un 85% de uso del árbol

Árbol B+ de prefijos simples

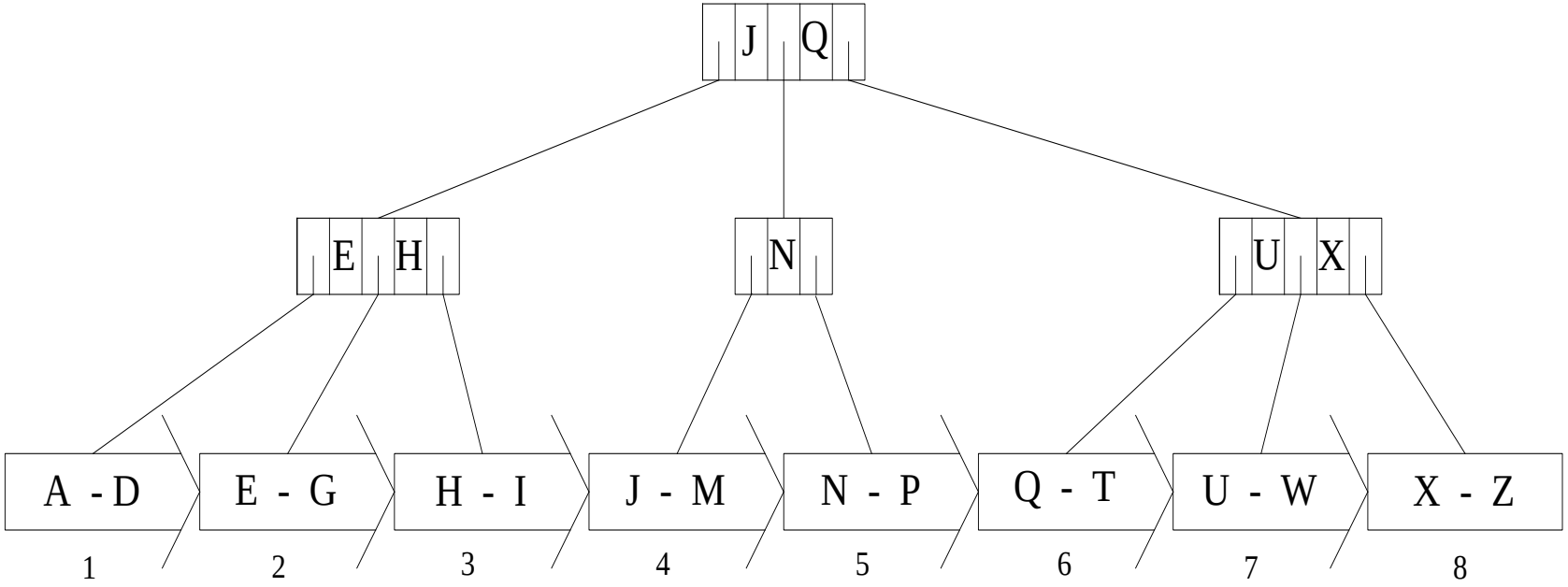
El caso más sencillo

- El árbol B de índices es llamado el conjunto índice
- Junto con el conjunto secuencia, forma una estructura de archivos llamada árbol B+
- Los separadores son simples debido a que son prefijos simples, son solo las letras iniciales dentro de las llaves

Árbol B+


E
BO
CAM
F
FOLKS
ADAMS-BERNE
BOLEN-CAGE
CAMP-DUTTON
EMBRY-EVANS
FABER-FOLK
FOLKS-GADDIS
1
2
3
4
5
6

Representando un árbol B como B+

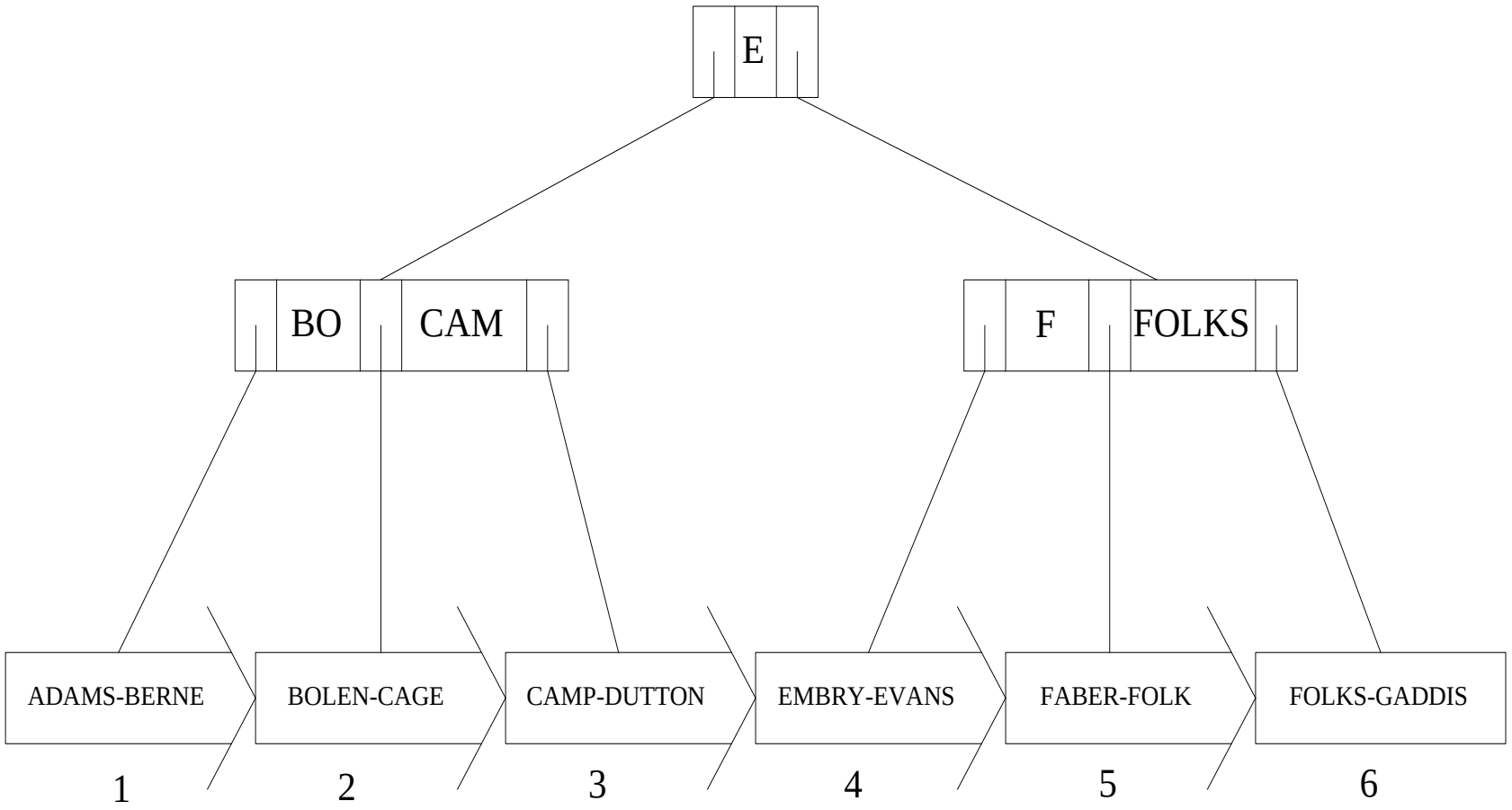


Mantenimiento de árboles B+ de prefijos simples

- Existen dos posibilidades:
  - Cambios localizados en un bloque simple
  - Cambios que involucran múltiples bloques

Cambios en un bloque simple

Árbol B+


E
BO
CAM
F
FOLKS
ADAMS-BERNE
BOLEN-CAGE
CAMP-DUTTON
EMBRY-EVANS
FABER-FOLK
FOLKS-GADDIS
1
2
3
4
5
6

Eliminación

- Suponga que se desean eliminar los registros de EMBRY y FOLKS y que ninguna de estas eliminaciones resultan en una mezcla o una redistribución

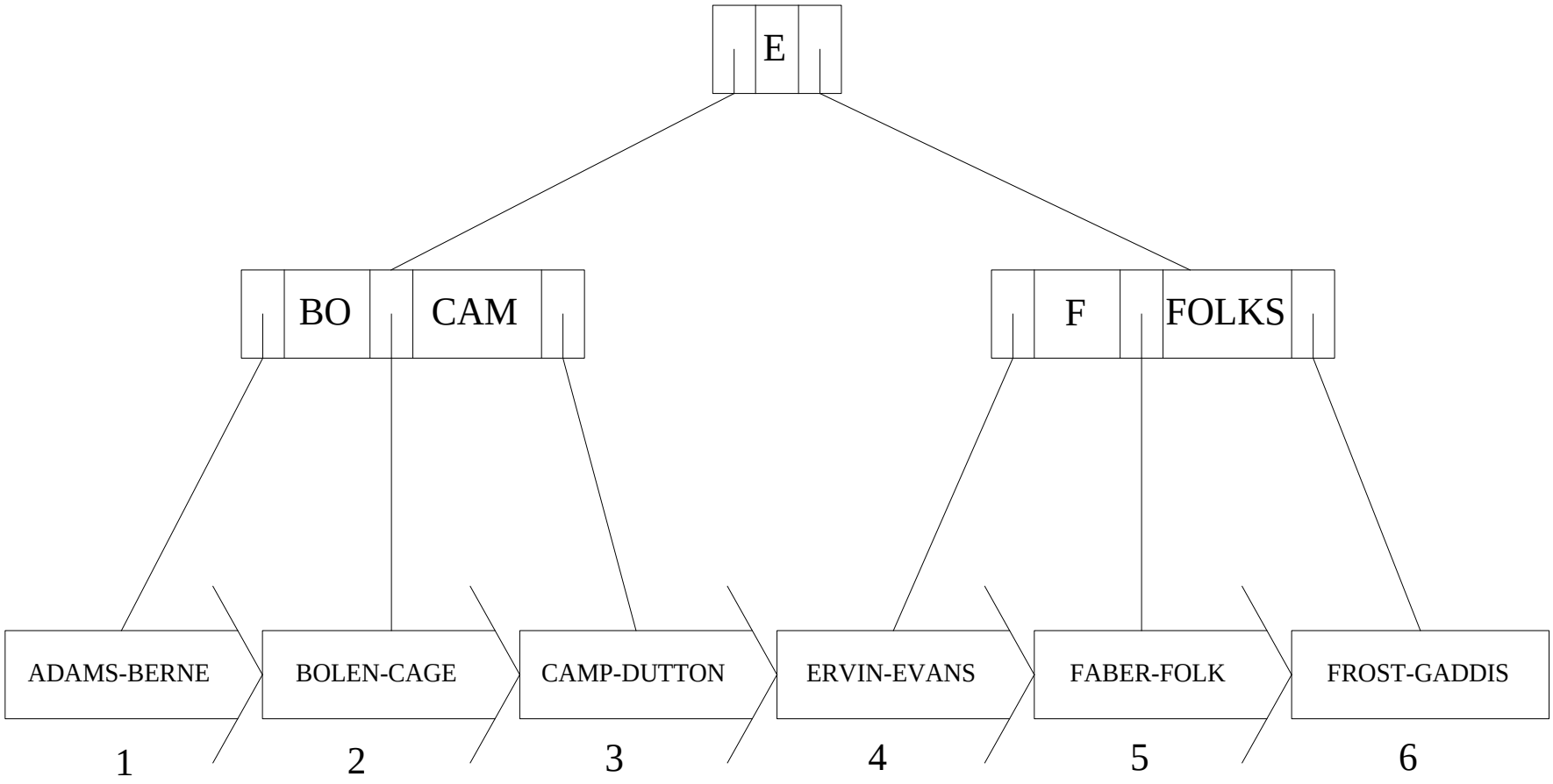
- El efecto de estas eliminaciones en el conjunto secuencia está limitado a los cambios dentro de los bloques 4 y 6

Eliminación

- El registro que era anteriormente el segundo en el bloque 4 ahora es el primero
- El segundo registro en el bloque 6 es ahora el de inicio en ese bloque
- Dado que el número de bloques de conjuntos de secuencia no ha sido modificado y que no se han movido registros entre los bloques, el índice puede permanecer sin ninguna modificación

Eliminación

- Esto se puede ver con la eliminación de EMBRY
- E aún permanece como un buen separador para el bloque de secuencias 3 y 4 por lo que no hay que realizar modificaciones en el índice
- El caso en donde se elimina FOLKS puede ser algo confuso por que aparece como una llave y como un separador
- A pesar de que la llave FOLKS ya no existe, el separador con ese nombre aún puede ser utilizado


E
BO
CAM
F
FOLKS
ADAMS-BERNE
BOLEN-CAGE
CAMP-DUTTON
ERVIN-EVANS
FABER-FOLK
FROST-GADDIS
1
2
3
4
5
6

Inserción

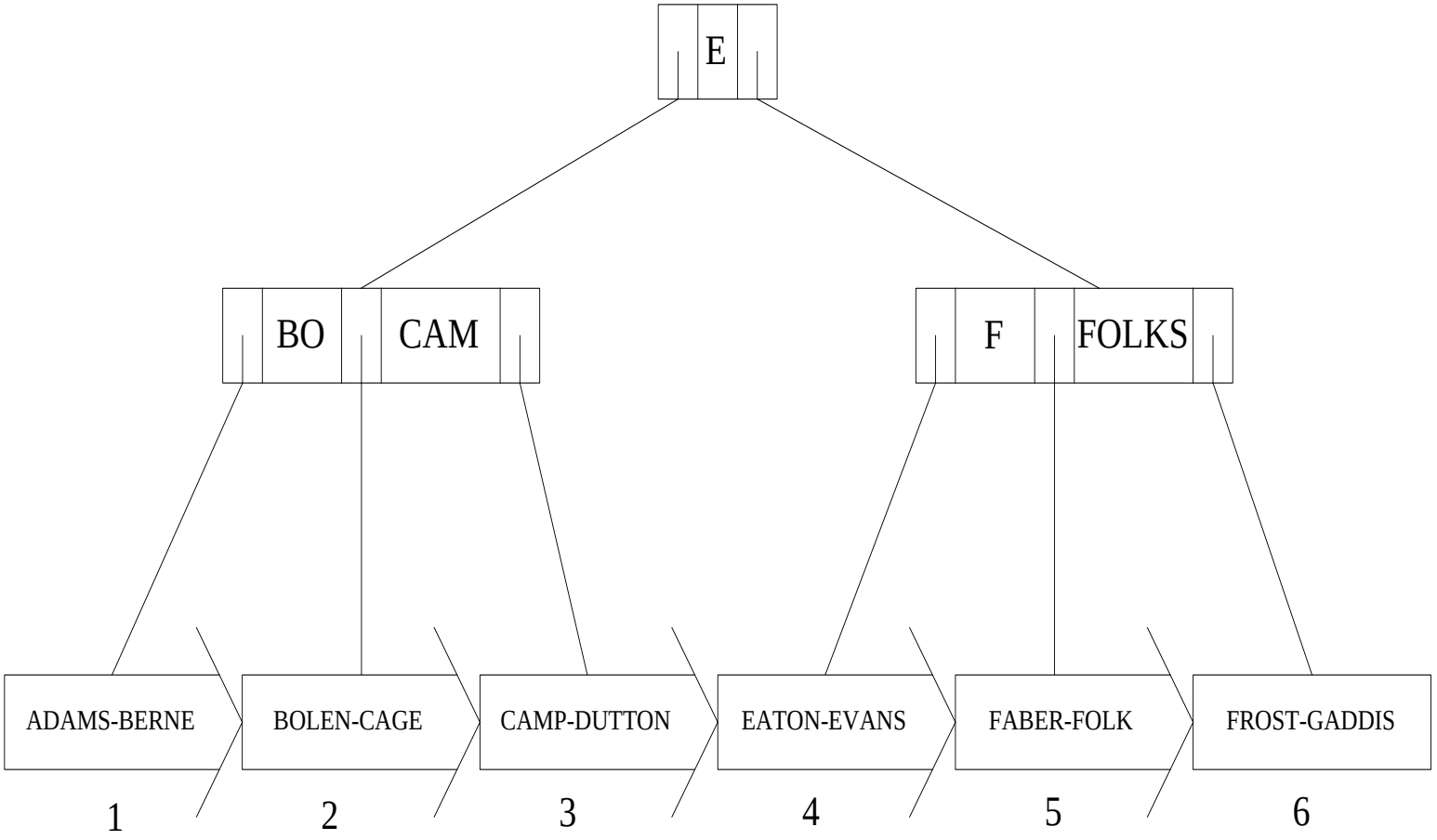
- El efecto de insertar en el conjunto secuencia nuevos registros que no causen una división de bloques es el mismo que cuando se eliminan registros que no causan partición:el índice permanece sin modificaciones

Ejemplo

- Si se inserta el registro con la llave EATON siguiendo el camino indicado por los separadores en el conjunto índice, se encuentra que se insertará en el bloque 4, por el momento se asume que hay espacio para insertarlo

- El nuevo registro se convierte en el primero en el bloque, pero no se requiere ninguna modificación en el índice

Después de la inserción



Cambios que involucran múltiples bloques

Introducción

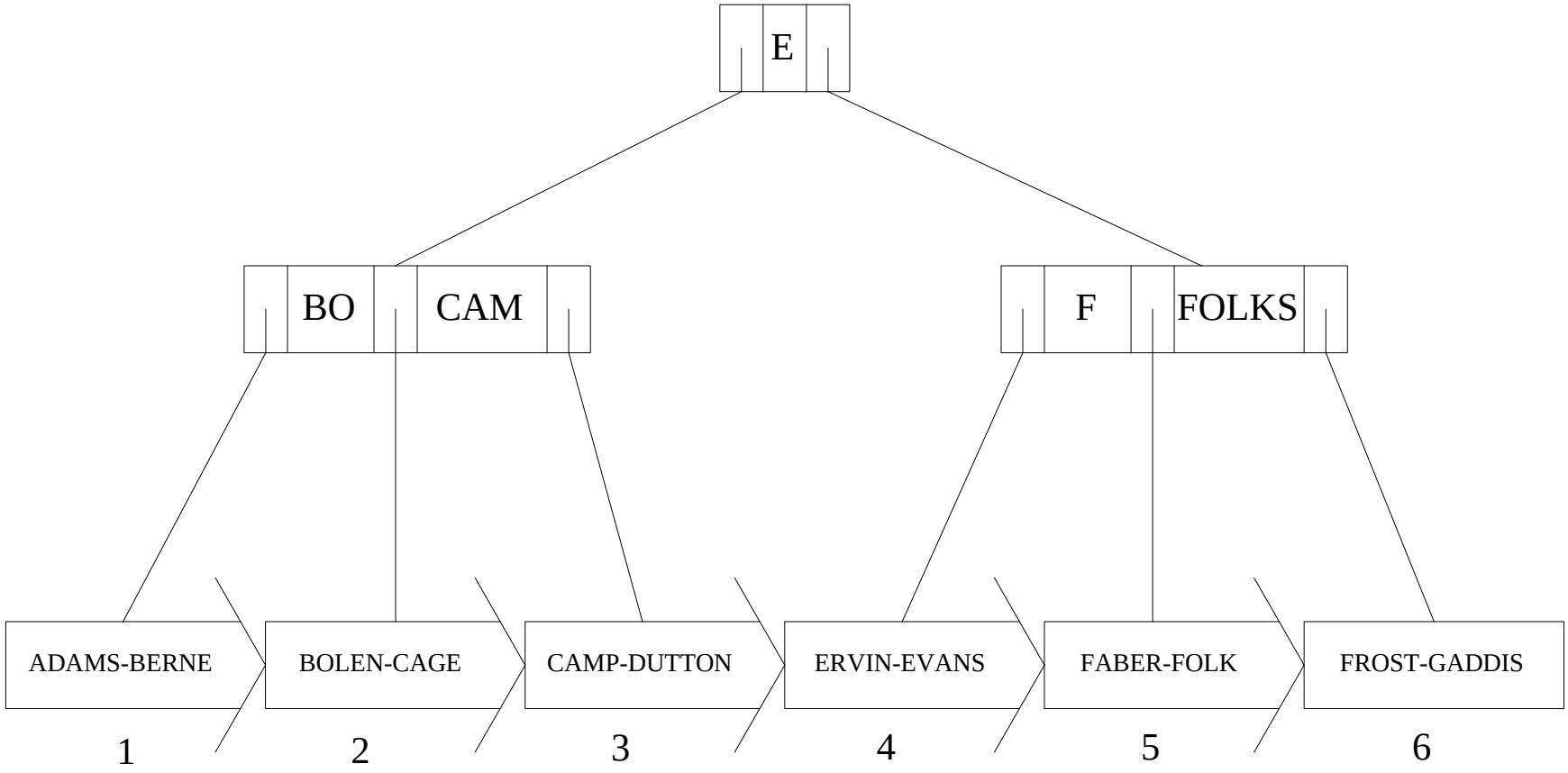
- Es posible que la adición o eliminación de registros alteren el número de bloques en el conjunto secuencia
- Si se tienen más bloques, se necesitan separadores extra en el índice y si se tienen menos bloques, se necesitan menos separadores
- Cambiar el número de separadores tiene un efecto en el conjunto de índices donde los separadores están almacenados

Inserción

- Dado que el conjunto de índices para un árbol B+ de prefijos simple es un árbol B normal, los cambios al índice son realizados de acuerdo a las reglas que se tienen para el manejo de árboles B

- Se asume que el conjunto índice es un árbol B de orden tres, lo que significa que el máximo número de separadores que se pueden tener en un nodo es dos

Ejemplo


E
BO
CAM
F
FOLKS
ADAMS-BERNE
BOLEN-CAGE
CAMP-DUTTON
ERVIN-EVANS
FABER-FOLK
FROST-GADDIS
1
2
3
4
5
6

Ejemplo, cont...

- Se asume que se realiza una inserción en el primer bloque y que esta inserción ocasiona que el bloque se divida

- Un nuevo bloque aparece para contener la mitad de los elementos que originalmente se encontraban en el primer bloque, este nuevo bloque está ligado en la posición correcta en el conjunto secuencia, siguiendo al bloque 1 y antes del bloque 2

Ejemplo, cont...

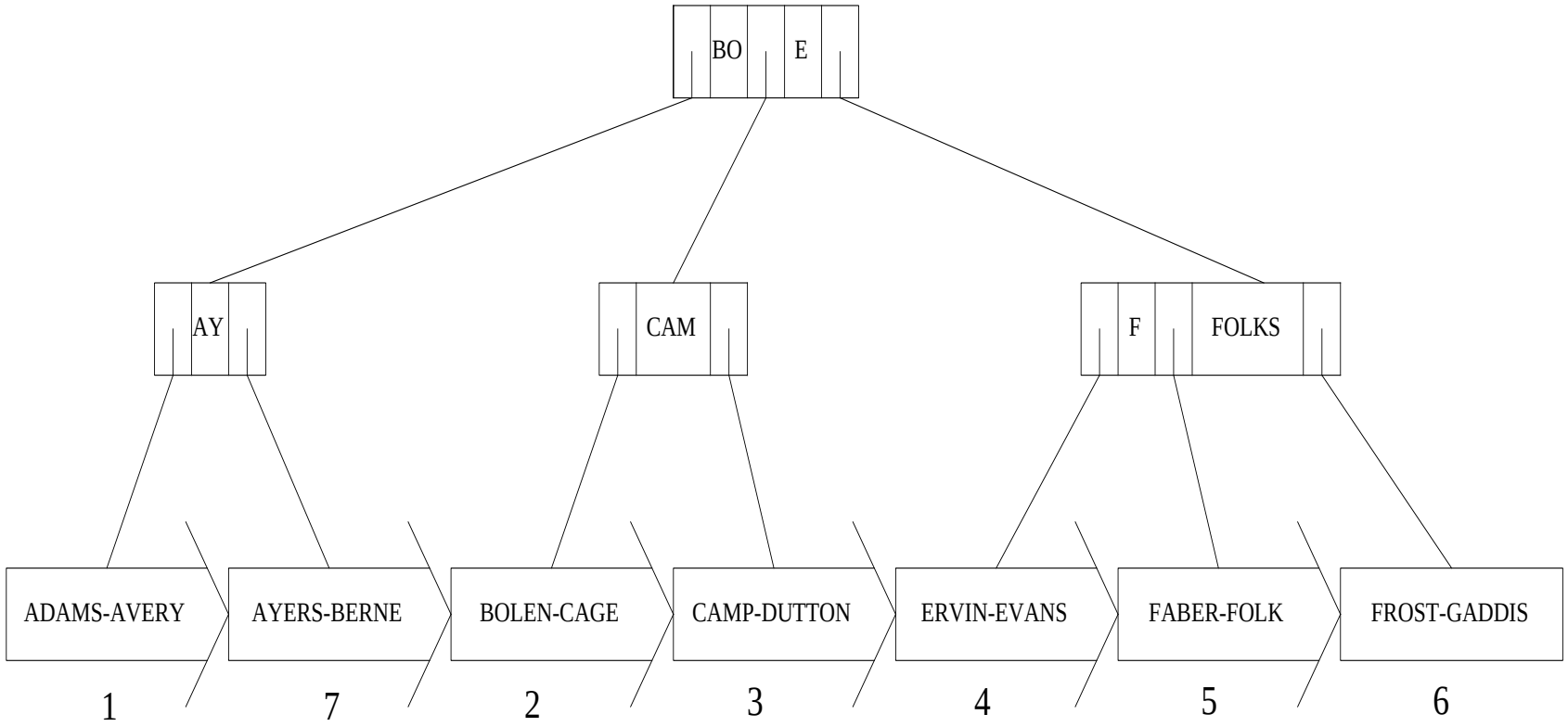
- El separador que anteriormente distinguía entre los bloques 1 y 2, BO, ahora es el separador entre los bloques 7 y 2

- Es necesario un nuevo separador con un valor de AY para distinguir entre los bloques 1 y 7

Ejemplo, cont...

- Cuando se coloca este separador en el conjunto de índices, se encuentra que el nodo donde debería colocarse, el que contiene BO y CAM, está lleno
- La inserción de este nuevo separador causa una división y una promoción
- De acuerdo a las reglas de los árboles B, el separador promovido BO, es colocado en el índice de la raíz

Ejemplo, cont...



Eliminación

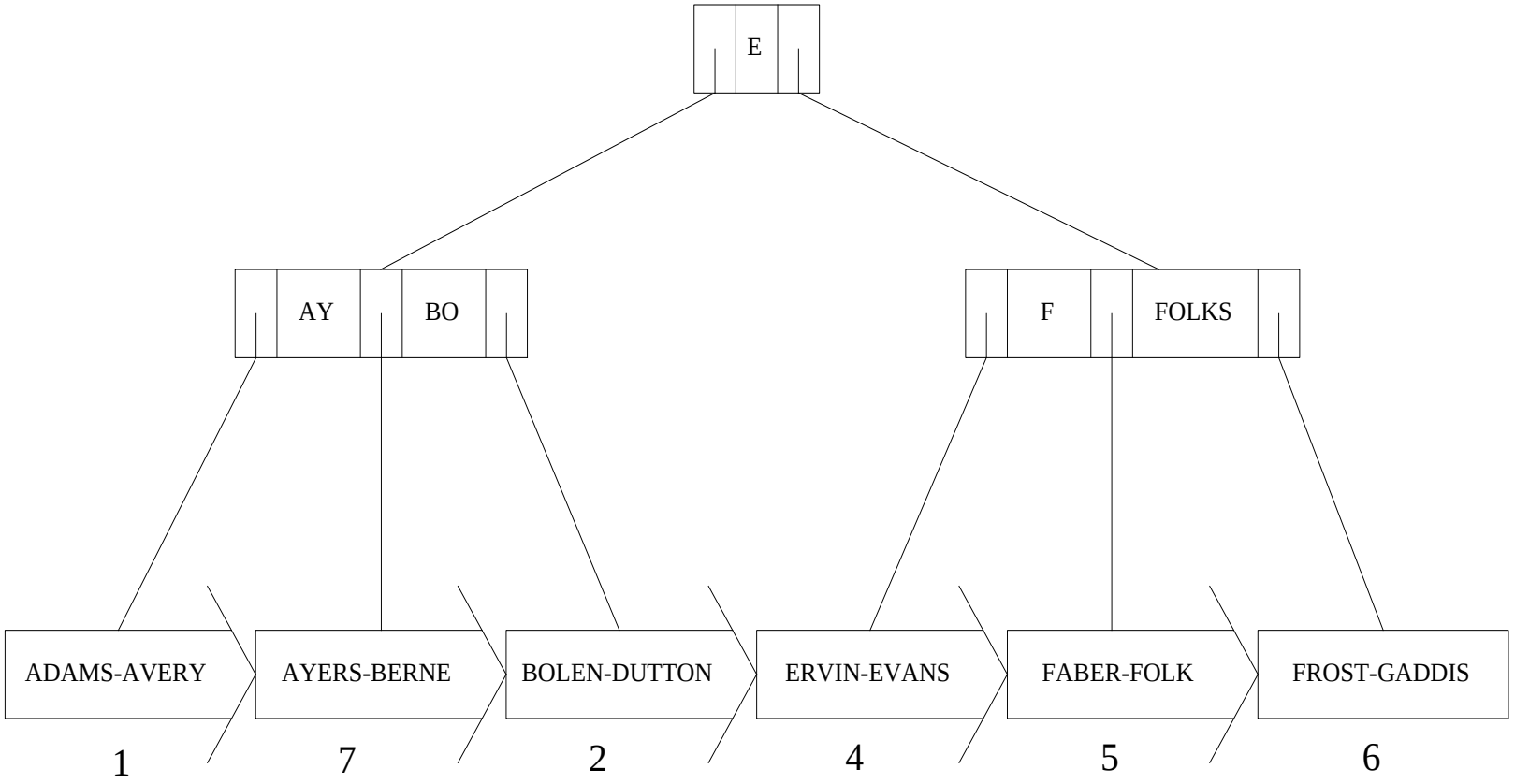
- Suponga que se elimina un registro del bloque 2 del conjunto secuencia y que esto ocasiona una condición de sub-flujo y la consecuente unión de los bloques 2 y 3

- Una vez que la mezcla se ha completado, el bloque 3 ya no es necesario y el separador que distinguía entre los bloques 2 y 3 debe ser removido del conjunto de índices

Ejemplo, cont...

- Por consecuencia existe otra mezcla, esta vez en el conjunto de índices que resulta en la degradación del separador BO de la raíz, bajándolo al nodo con el separador AY

Ejemplo, cont...



Cargando un árbol B+ de prefijos simples

Construcción

- Una manera de construir un árbol B+ de prefijos simple es a través de una serie de inserciones sucesivas

- La dificultad con esta aproximación es que la partición y la mezcla son operaciones relativamente costosas, ya que involucran una búsqueda hacia abajo del árbol para cada inserción y luego una reorganización del árbol hacia arriba si fuera necesario

Construcción

- Estas operaciones están bien para el mantenimiento del árbol conforme es actualizado, pero cuando se está cargando un árbol no se tiene que lidiar con una inserción aleatoria

- Por lo tanto no se necesitan procedimientos tan costosos

Construcción

- Trabajando con un archivo ordenado, se pueden colocar los registros en conjuntos de bloques consecutivos, uno a uno, comenzando un nuevo bloque cuando con el que estemos trabajando se encuentre lleno

- Conforme se realiza la transición entre dos conjuntos de bloques, se puede determinar el separador más pequeño para los bloques

Árboles B+

Definiciones

- Los árboles B+ de prefijos simples son una variante de una aproximación a la organización de archivos conocida como árboles B+

- La diferencia entre ellos es que un árbol B+ no involucra el uso de prefijos como separadores, en lugar de eso los separadores en el conjunto índice son simples copias de las llaves actuales

- Las operaciones realizadas en los árboles B+ son esencialmente las mismas que las analizadas para los árboles B+ de prefijos simples

- La única diferencia es que en un árbol B+ de prefijos simple se construye un conjunto de índices de separadores más cortos formado por los prefijos de las llaves

Árboles B+ contra Árboles B+ de prefijos simple

- Los árboles B+ de prefijos simples son una alternativa más deseable que los árboles B+

- Se desea un árbol lo menos profundo posible, lo que implica que se desean colocar tantos separadores en el bloque de índices como sea posible, es por eso que es preferible utilizar un simple prefijo

Árboles B+ contra Árboles B+ de prefijos simples

- Existen dos factores que favorecen a los árboles B+ que utilizan copias de las llaves como separadores:
  - Para algunas aplicaciones, el costo del mantenimiento de los árboles B+, en especial el mantener las estructuras de longitud variable, puede ser mayor que los beneficios que genera

Árboles B+ contra Árboles B+ de prefijos simples

- Algunos conjuntos de llaves no muestran mucha compresión cuando se utiliza un método de prefijos simples y se requiere utilizar un método que elimine la redundancia, desafortunadamente éste tipo de métodos pueden llegar a ser muy costosos
- Ejemplo:

34C18K756, 34C18K757, 34C18K758