Dispersión: Es un campo elegido para realizar una búsqueda.

	A		X
0	1	2	3

f(X) = 3; f(A) = 1

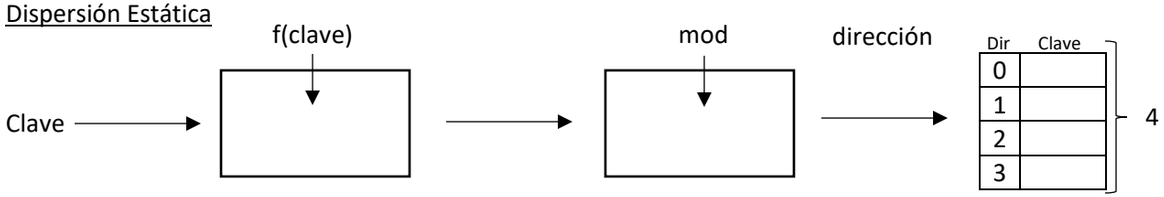
Se realiza búsqueda directa, la función de la determinada letra automáticamente da su posición guardada.

Esta búsqueda es más rápida, pero de buscar mediante otro campo debemos acudir a las formas de búsqueda tradicionales.

La clave de dispersión debe ser una clave o un dato único para no pisar datos, ej: mismo nombre daría la misma ubicación y pisaría uno.

El bloque determina cuantos registros puede tener cada campo.

Dispersión Estática



Para adecuar el número de la conversión de la clave a un valor dentro del bloque utilizamos el “mod” de la cantidad de bloques. Ese “mod” nos da el resto y guardamos en esa posición.

Ej: Clave da 19

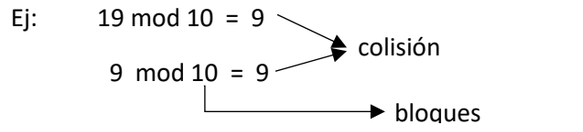
Conversión a dirección → 19 mod 4 = 19 | 4 → Guarda el valor 19 en la posición 3.

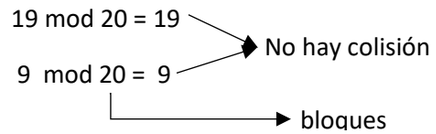
3 | 4

Si en otra clave ingresada posteriormente se realiza el mod y nos da el mismo valor resto (3) se intentaría guardar en el mismo lugar y se genera lo denominado “colisión”.

Un método útil para minimizar las colisiones es aumentar los bloques, aunque aumenta el desperdicio, es decir, los lugares vacíos que no se aprovechan.

Ej:





El <u>empaquetamiento</u> da el <u>porcentaje</u> utilizado teniendo en cuenta datos guardados y espacio desperdiciado.

$$\text{Densidad de Empaquetamiento} = \frac{\#\text{RegistrosOcupados}}{\#\text{RegistrosPorBloque} \times \#\ \text{Direcciones}}$$

<u>Tipos de Dispersión</u>:

- Estática.
  - Saturación Progresiva.
  - Saturación Progresiva Encadenada.
    - Con intruso.
    - Sin intruso.
    - Con área de desborde.
  - Dispersión Doble.
  - Redispersión.
- Dinámica.

Saturación Progresiva:

- De estar ocupado un lugar se guarda en el lugar siguiente hasta encontrar uno vacío, al llegar al final vuelve al primero. De llegar al lugar de donde partió significa que no hay lugar disponible.
- Se detiene la búsqueda cuando encuentra el nombre o dato buscado, cuando llega al inicio de donde empezó la búsqueda y también si encuentra una celda vacía. Estos últimos dos casos indican que el valor no está (ya sea porque recorrió toda la tabla y volvió al lugar de donde inició la búsqueda o porque encontró un lugar vacío debido a que en Saturación Progresiva el valor que se ingresa se guarda en una determinada dirección y de estar ocupada se busca desde allí la primera posición vacía lo que indica que si en la búsqueda de un valor hay un lugar vacío ese valor no está).

Ej: Pedro = 2

Dir	Clave
0	Malena
1	Lucho
2	Camila
3	Gabriel
4	Pedro
5	
6	

Como la posición 2 está ocupada busca a partir de allí la primera posición vacía que encuentre para guardar el valor “Pedro” (que es la posición 4).

Dir	Clave
0	Malena
1	Lucho
2	Camila
3	Gabriel
4	Pedro
5	
6	

Poniendo el mismo ejemplo, si buscáramos la clave “Carlos” comenzaría la búsqueda y cortaría el proceso al llegar a la posición 5, debido a que no encontró el resultado “Carlos” pero si encontró una celda vacía (lo que indica que “Carlos” no está cargado porque debería estar en esa posición, es decir, en su correspondiente o en la primera vacía).

Saturación Progresiva Encadenada **Sin Intruso**

Se recorren todos los registros y se guardan los que tienen posiciones vacías dejando en suspenso los que van en lugares ocupados.

Los registros colocados se escriben con un "-1" en su enlace indicando que no tienen sinónimos.

Cuando se realizó la primera pasada se toman los que quedaron en suspenso y se acomodan normalmente buscando una posición vacía con en Saturación Progresiva, ahí el -1 del registro que ocupaba la posición original se cambia por el valor de la posición donde se guardó el segundo registro.

Ej:

H(Juan) = 1

H(XXX) = 2

H(Maria) = 1

H(ZZZ) = 3

Dir	Clave	Enlace
0		
1	Juan	-1
2	XXX	-1
3	ZZZ	-1
4		
5		

→ Queda en suspenso porque la posición 1 ya está ocupada por Juan.

Una vez acomodados todos los valores colocamos los que quedaron en suspenso (en este caso María) haciendo como en Saturación Progresiva, es decir, colocándolo en el primer lugar vacío que se encuentre, pero modificando el enlace de su posición original (es decir el enlace del valor que está en la posición 1).

H(Maria) = 1

Dir	Clave	Enlace
0		
1	Juan	~~-1~~ 4
2	XXX	-1
3	ZZZ	-1
4	María	-1
5		

Esta forma de Dispersión hace que del valor Juan (que está en la celda 1) se enlace directamente con los demás valores que ocuparían ese lugar si estuviese disponible (en este caso María). En otras palabras, la búsqueda funciona de la siguiente manera: Se quiere buscar a María (que tiene valor 1), entonces se busca en la tabla en la celda 1 (donde está Juan) y de Juan salta la

búsqueda hasta la celda 4 utilizando los enlaces (donde se encuentra María). Es más óptimo que la Saturación Progresiva porque gracias a los enlaces no se tiene que buscar celda por celda para encontrar un valor.

De ocurrir esto varias veces la posición original va a tomar el valor del último registro que está en suspenso y ese toma el valor del anterior. Se podría decir que es como un insertar al inicio pero detrás del valor original (en este caso Juan).

Ej:

H(Carlos) = 1

Dir	Clave	Enlace
0		
1	Juan	~~-1~~ ~~4~~ 5
2	XXX	-1
3	ZZZ	-1
4	María	-1
5	Carlos	4

Ahora la búsqueda de María iniciaría por la celda 1 (donde debería estar), como es Juan enlaza con la celda 5 (que es Carlos) y de ahí enlaza con la celda 4 que es María. Y siguiendo con esta lógica podemos ver como el -1 indica que es el final.

Saturación Progresiva Encadenada **Con Intruso**

Se acomodan todos los registros y de repetir la posición se le debe realizar la función de dispersión al registro que ya se encuentra guardado ocupando la celda para ver si es intruso, de no serlo el registro que queríamos acomodar se guarda en la primera posición vacía que encuentra y se modifican los enlaces (igual que en Saturación Progresiva Encadenada Sin Intruso). Mientras que si es intruso se desplaza el valor intruso que estaba cargado a otra posición vacía y se pone el nuevo valor correspondiente en la posición.

Ej: h(x) = producto de los dígitos de X.

h(23) = 6

h(32) = 6

h(71) = 7

h(16) = 6

Dir	Clave	Enlace
0	32	-1
1	16	0
2		
3		
4		
5		
6	23	~~-1~~ ~~7~~ ~~0~~ 1
7	71	-1

Explicación procedimiento:

-Colocamos el valor 23 en la celda 6.

-El valor 32 también corresponde a la dirección 6, pero como esta ocupada por el valor 23 corroboramos si es intruso. Como ese valor 23 no es intruso (porque le corresponde la dirección 6) colocamos el valor 32 en el primer lugar vacío (en este instante de tiempo seria en la dirección 7) y modificamos los enlaces.

-Luego viene el valor 71 que corresponde a la dirección 7, pero como esta ocupada por el valor 32 corroboramos si es intruso. Como ese valor 32 es intruso (porque esta en la dirección 7 pero le corresponde la dirección 6) se mueve a la primera posición vacía (la dirección 0), modificamos los enlaces y colocamos en la dirección 7 el valor 71.

-Por último, viene el valor 16 que corresponde a la dirección 6 que esta ocupada. Como ese valor 23 no es intruso buscamos la primera dirección vacía para acomodarlo (la dirección 1) y actualizamos los enlaces.

Saturación Progresiva Encadenada Con Área de Desborde

Este tipo de dispersión cuenta con dos tablas, el área principal y el área de desborde. En el mismo se acomodan los registros en la tabla principal mientras que de repetirse la posición se coloca en el primer lugar vacío, pero de la tabla de desborde y se actualizan los enlaces.

Ej: h(x) = producto de los dígitos de X.
Área principal = 6
Área de desborde = 4

h(22) = 4
h(13) = 3
h(14) = 4
h(41) = 4

Área Principal

Dir	Clave	Enlace
0		
1		
2		
3	13	-1
4	22	~~-1~~ ~~0~~ 1
5		

Área de Desborde

Dir	Clave	Enlace
0	14	-1
1	41	0
2		
3		

Dispersión Doble

Se realiza la función de dispersión dos veces. De estar ocupada una posición se va a colocar tantas posiciones después como lo dice la segunda función de dispersión. Para buscar un valor se aplica la función de dispersión y de no encontrarlo se suma el valor de la segunda función de dispersión hasta encontrarlo o volver al lugar de partida por recorrer toda la tabla. En otras palabras, la búsqueda se da de a saltos según el valor de la segunda función de dispersión.

h(x) = Suma de los dígitos pares.

$h_2(x)$ = x mod 8

h(47) = 4
h(233) = 2
h(814) = 12 mod 11 = 1
h(202) = 4 → $h_2(202)$ = 202 mod 8 = 2
h(207) = 2 → $h_2(207)$ = 207 mod 8 = 7

Dir	Clave
0	
1	814
2	233
3	
4	47
5	
6	202
7	
8	
9	207
10	

Explicación procedimiento:

(Tomamos de ejemplo el valor 233 y 207 pero la lógica es la misma para los demás)

-Colocamos el valor 233 en la dirección 2 haciendo la función de dispersión.

-Realizando la función de dispersión al valor 207 también nos da la dirección 2 pero se encuentra ocupada, entonces se realiza la segunda función de dispersión que da 7. Eso indica que este valor (207) se va a colocar 7 posiciones debajo de la dirección original (2), es decir la dirección 9.

-De estar ocupada la posición 9 se vuelven a contar 7 posiciones para ubicar el valor (es decir quedaría en la posición 5).

-Contando de a 7 posiciones va a llegar un momento donde la dirección resultante va a volver a ser la 2, si para ese instante de tiempo el número no pudo ser ubicado en ese caso no se puede ubicar y hay que redispersar. Lo mismo pasa con la búsqueda, si el numero no se encuentra el proceso se corta cuando se llega al punto de partida (es decir luego de buscar en todas las posibles direcciones y donde se empieza a repetir el bucle por volver a partir de la posición inicial).

-En este caso el bloque NO utiliza enlaces, debido que el enlace entre los diferentes valores que corresponden a una misma dirección lo da la segunda función de dispersión.

Redispersar

De estar lleno o no poder ubicar un valor duplicamos el fichero (o tabla) con el doble de posiciones y de ahí buscamos, sumando, el valor primo más cercano.

El orden en que agarramos cada registro para ubicarlo en la nueva tabla va a ser en el orden que quedaron en la tabla anterior (que quedó chica de espacio), y luego continuamos con los valores que no pudieron ser agregados por falta de espacio.

Densidad de Empaquetamiento

Como dijimos arriba la Densidad de Empaquetamiento da el porcentaje utilizado teniendo en cuenta datos guardados y espacio desperdiciado, y su fórmula es:

$$\text{Densidad de Empaquetamiento} = \frac{\#\text{RegistrosOcupados}}{\#\text{RegistrosPorBloque} \times \#\text{ Direcciones}}$$

Ej:

Si nos pide que calculemos la DE dado un fichero (o tabla) entonces:

Dir	Clave	Clave
0	222	
1		
2	655	
3		
4		
5	729	113
6		
7	520	425
8		
9	498	
10	347	

$$\frac{\#\text{RegistrosOcupados}}{\#\text{RegistrosPorBloque} \times \#\text{ Direcciones}} = \frac{8}{2 \times 11} = \frac{8}{22} = \boxed{36\%}$$

→ Densidad de Empaquetamiento.

Por otro lado, en general, los ejercicios suelen decir, por ejemplo, que se debe cumplir que **DE <= 50%** y de superarse se debe redispersar. Eso lo calculamos con una regla de 3 simples:

(Utilizamos de ejemplo el mismo fichero de arriba)

22 ---------- 100%

X ---------- 50%

→ 11

Por lo tanto, debemos colocar 11 valores y luego redispersar (porque se supera el 50% pedido).

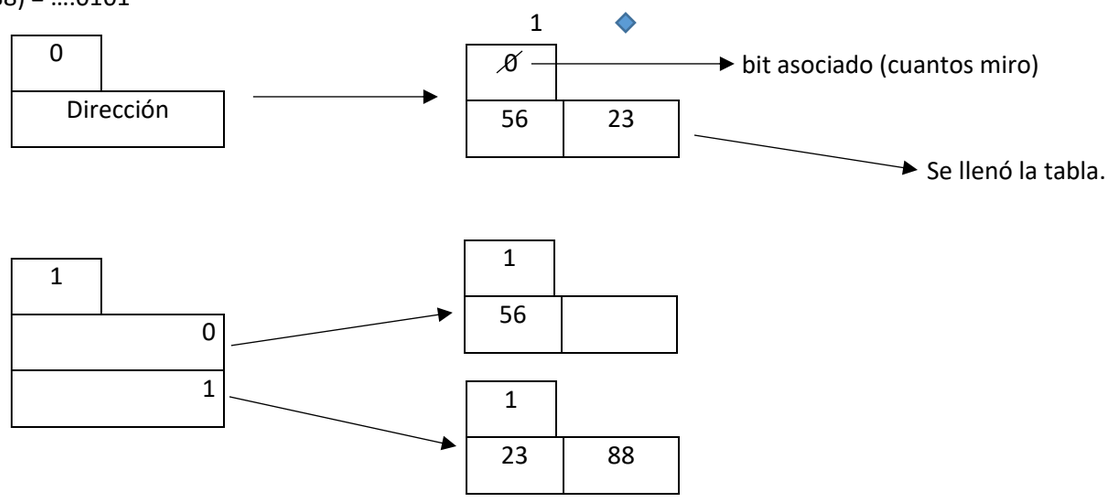
Dispersión Dinámica

- El espacio crece en base a la necesidad del archivo.
- Las claves son dispuestas en función de la cantidad de direcciones disponibles en cada momento.
- La función de dispersión retorna una cadena de bits. Eso indica la cantidad máxima de direcciones a las que puede acceder el método.

H(56) = ….0010

H(23) = ….0111

H(88) = ….0101



Explicación

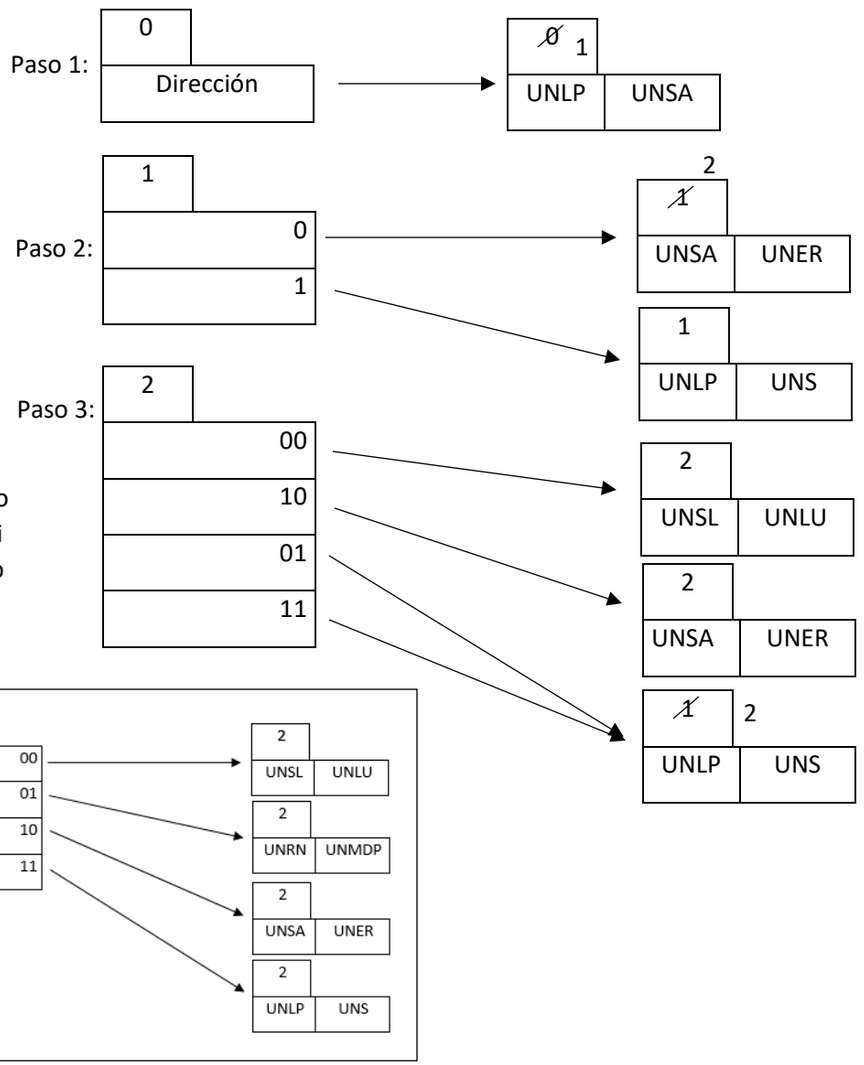
- Como el bit asociado es “0” quiere decir que no se mira ningún bit del valor a agregar (es decir, se agrega indiferentemente del conjunto de bits).
- Cuando se llena la tabla de la derecha ◆ el bit asociado se incrementa en 1. Como ahora la tabla de la derecha tiene el bit asociado en 1 y la tabla de la izquierda tiene un bit asociado menor (porque es 0), se incrementa también a 1 y entonces la tabla de la izquierda va a mirar 1 solo bit (que puede ser 0 o 1) y por eso se divide en 2, y se agregan valores mirando el último bit.
- Cuando se vuelve a llenar una de las tablas de la derecha (porque hace falta lugar para agregar otro valor) se vuelve a incrementar el bit asociado (en este caso a 2).

- Como nuevamente la tabla de la izquierda va a tener el bit asociado menor (porque es 1) se incrementa a 2 y ahora se miran los últimos 2 bits del valor a agregar. Por ende, se va a dividir en 4 (es decir al mirar 2 bits tenemos 4 combinaciones posibles 00, 01, 10, 11).
- Si al incrementarse el bit asociado de la tabla de la derecha ese valor no es mayor que el bit asociado de la tabla de la izquierda solo se subdivide la tabla de la derecha sin modificar la izquierda.
- Cada vez que aumentamos el bit asociado debemos acomodar todos los valores desde cero nuevamente (porque ahora van a agruparse mirando más bits que antes).

Ejemplo práctico para mayor comprensión:

Valores a agregar:

1	UNLP	00001011
2	UNSA	01010010
3	UNER	10101110
4	UNS	00011011
5	UNSL	01100100
6	UNLU	11000100
7	UNRN	11100101
8	UNMDP	00111001



//No miro ningún bit (bit asociado es 0) entonces se acomoda sin criterio.
//Al llenarse cambia el bit asociado a 1.

//Se acomoda mirando 1 bit (el último).

//Las tablas de la izquierda agrandan su tamaño respecto al número del bit asociado. Es decir, si el bit asociado es 2 va a haber tantas filas como combinaciones de 2 dígitos binarios haya.

//La tabla que se llenó en el paso anterior sale de la fila del "0" por ende ahora se miran 2 bits (como lo indica el bit asociado) y se agregan 2 tablas para las dos posibles combinaciones que terminan en "0".

//La ultima tabla tiene dos flechas hacia ella porque el bit asociado sigue siendo 1, es decir, que sigue mirando el ultimo bit (por ende, las dos combinaciones terminadas en "1" van hacia ella).

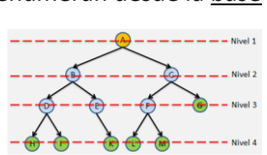
//Ahora como la tabla que se llenó es la que tiene como bit asociado un 1 (que pasa a ser 2) la tabla de la izquierda no se modifica porque ya tenía un 2. Es decir, solo se subdivide la tabla de la derecha separando las dos posibles combinaciones terminadas en "1" (igual que como pasó anteriormente con el "0").

Arboles:

- Estructura no lineal, es decir, un nodo puede tener más de un hijo. Cada rama se abre y toma varios valores, es decir, se ramifica.

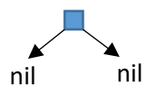
Árbol binario:

- Estructura dinámica no lineal en la que cada nodo puede tener 0, 1 o máximo 2 hijos.
- El árbol tiene distintos niveles, se enumeran desde la base hasta las hojas (que son los nodos que no tienen hijos).



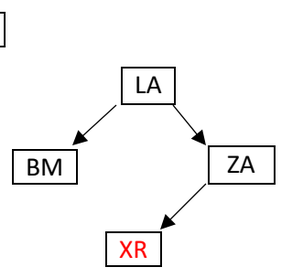
Árbol Unario:

- Es el árbol que la base también es hoja.



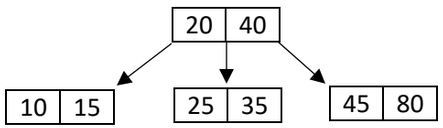
Árbol Binario de Búsqueda:

Para ordenar se mira alfabéticamente las letras, de ser menor al padre se pasa a la rama izquierda y se vuelve a repetir el proceso hasta encontrar un lugar vacío donde queda el nodo (mientras de ser mayor se pasa a la rama derecha).



Árbol Multicamino:

- Estructura de datos en la cual cada nodo puede contener **K** elementos y **K + 1** hijos.



Árbol B:

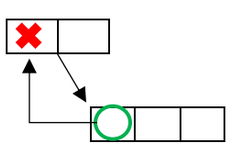
Ponemos de ejemplo un árbol de **Orden 6** ⟶ 6 – 1 = 5 claves

Inserción:

- Siempre insertamos a nivel hoja.
- Metemos en la raíz los primeros 5 valores ordenados hasta llenarlo.
- Incorporamos el nuevo valor (quedarían 6 claves), lo partimos en dos y tomamos el valor de la derecha que sube como padre del resto. Si al incorporar el nuevo valor quedan números de claves impares tomamos el que queda en el medio y sube como padre del resto.
- Cuando una hoja no tiene espacio se hace lo mismo que en el paso anterior y el que sube queda como padre con el valor que ya estaba.
- Cuando se llena el hijo y el padre no tiene lugar se parte al medio el hijo, sube el valor de la derecha, se parte el padre (o la raíz) incluyendo el valor que subió y vuelve a subir el de la derecha.
- Básicamente cuando se llena se parte y se promociona el de la derecha.

Eliminación:

- Hay que tener en cuenta el número mínimo de claves dado por la formula $\left[\frac{n}{2}\right] - 1.$ (Orden)
- Si eliminamos un valor de un nivel que no es hoja y no se cumple el número mínimo de claves agarramos la rama hoja que sale de la derecha del número a eliminar y reemplazamos el valor eliminado por el más chico de la hoja.

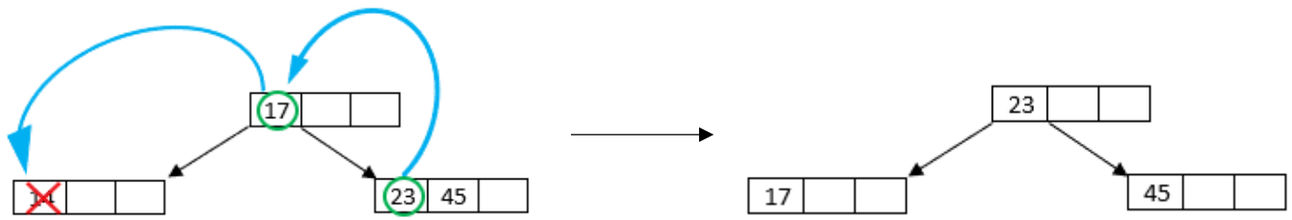


- Para eliminar a nivel hoja si no cumple la cantidad mínima de claves se pide al hermano derecho, de no tener o no poder se pide al hermano izquierdo y si no puede se concatena (con el de la derecha excepto que sea el extremo derecho y concatena con el de la izquierda). Mientras que de poder prestar se redistribuye.
- Si elimino a la raíz se reemplaza con el valor más chico de la rama derecha a nivel hoja.

Cuando un hermano presta una clave se hace así:

Cant. Min. Claves = 1

Eliminamos el 14 y pide al hermano derecho:



//Al pedirle al hermano derecho se redistribuye, es decir, que el padre baja donde fue eliminado el valor y valor más chico de su hermano derecho sube como padre.

Concatenar:

Se junta el que pide con el hermano de la derecha y el padre en común incluido.

Ejercitación Árbol B:

Dado el siguiente conjunto de claves: 56, 67, 23, 89, 40, 34, 102, 2, 10, 95, 60

a- Generar un árbol B de orden 4.

b- Mostrar cómo queda el árbol generado en el punto anterior, luego de eliminar 102, 34, 67, 60.

a-

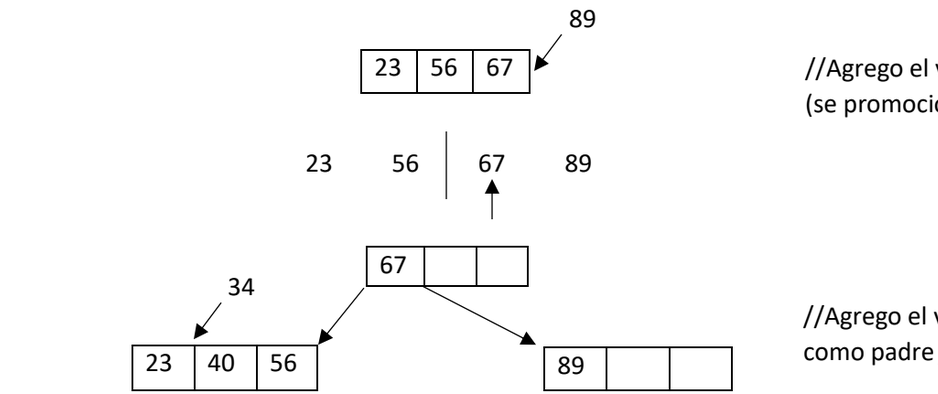
Orden 4

Cant. Max. Claves = **n – 1** = 3

Cant. Min. Claves = $\left\lceil \frac{n}{2} \right\rceil - 1 = 1$

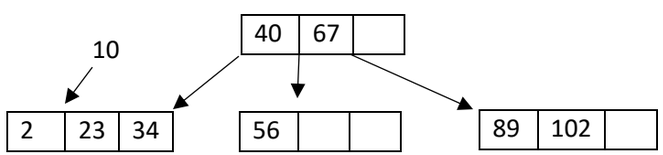
Cant. Min. Hijos = $\left\lceil \frac{n}{2} \right\rceil = \left\lceil \frac{4}{2} \right\rceil = 2$

Cant. Max. Hijos = **n** = 4

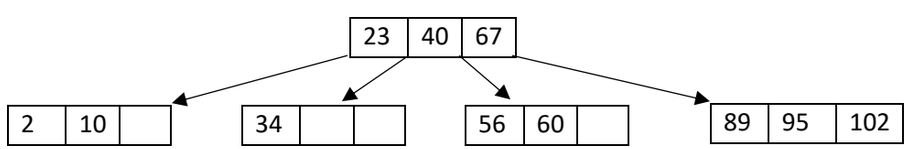


//Agrego el valor 89, parto al medio y sube el de la derecha como padre (se promociona).

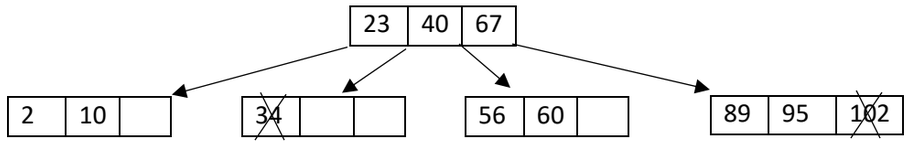
//Agrego el valor 34, parto al medio al hijo y sube el de la derecha (40) como padre junto al 67



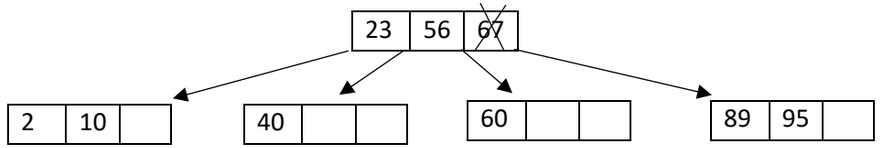
//Agrego el valor 10, parto al medio al hijo y sube el de la derecha (23) como padre junto a los otros.



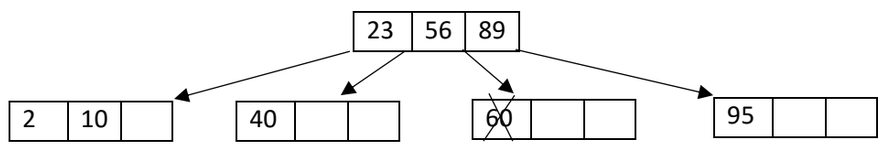
b- Eliminar 102, 34, 67, 60



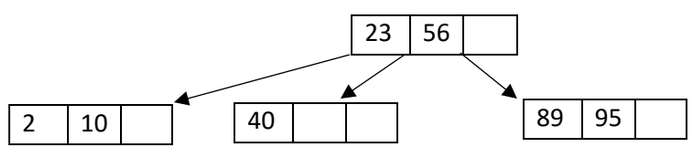
//No se puede eliminar el 34 porque no cumple la Cant. Min. Claves (1). Entonces le pide al hermano derecho y se redistribuye.



//Se elimina el 67 y sube el hijo derecho más chico en su lugar.



//Al eliminar el 60 ninguno de sus hermanos puede prestarle entonces se concatena (junto nodo que pide con hermano derecho y padre en común).



Árbol B *:

SOLO VEMOS LA INSERCIÓN DEBIDO QUE LA ELIMINACIÓN ES IGUAL QUE EN ARBOL B.

Inserción:

- En la inserción necesitamos tener dos páginas llenas.
- En este tipo de árboles tenemos lo denominado “políticas”. Estas políticas pueden ser: de “derecha”, “izquierda”, “izquierda derecha” o “derecha izquierda”.
- Las políticas “derecha” e “izquierda” implican que hermano mirar a excepción de los extremos que pueden mirar el hermano contrario a la política (por no tener el hermano que la política dice). Es decir, si es política de derecha se mira el hermano derecho, pero si la página es el extremo derecho mira el hermano izquierdo (porque no tiene derecho).
- Mientras que la política de “izquierda derecha” o “derecha izquierda” mira el hermano que corresponde y de no tener lugar mira al otro hermano, es decir, por ejemplo, la política “izquierda derecha” mira el hermano izquierdo y si no tiene lugar mira el hermano derecho.
- Si por ejemplo usamos política “izquierda derecha” y ninguno de los dos tiene lugar se realiza el proceso de armado de secuencia con el hermano izquierdo por ser política “IZQUIERDA derecha”.
- Cuando insertamos un valor y la pagina está llena le pasamos al hermano un valor para poder incorporar el nuevo. Es decir, aplicamos redistribución.

Ejercitación Árbol B*:

Orden 6 con política de izquierda

Orden 6

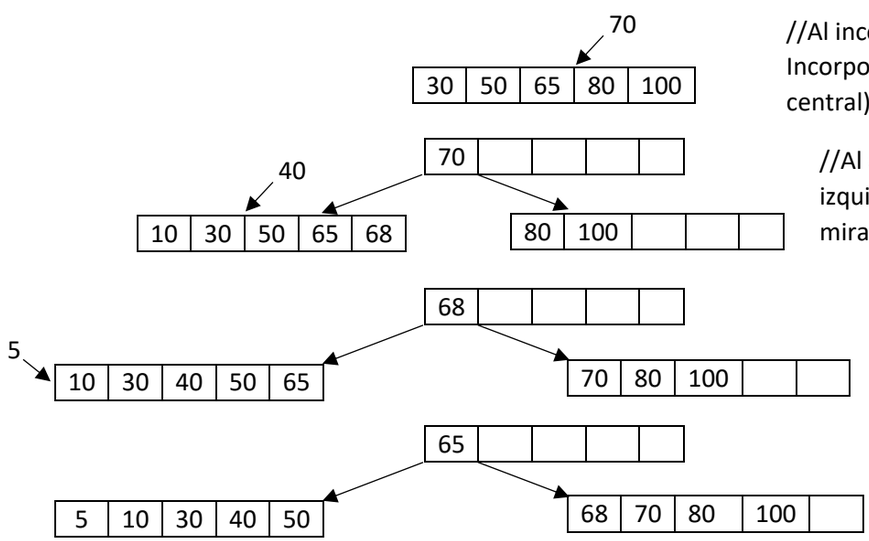
Cant. Max. Claves = **n – 1** = 5

Cant. Min. Claves = $\left[\frac{2}{3}n\right] - 1 = 3$

Cant. Min. Hijos = $\left[\frac{2}{3}n\right] = \left[\frac{2}{3}6\right] = 4$

Cant. Max. Hijos = **n** = 6

Agrego: 50, 30, 65, 100, 80, 70, 10, 68, 40, 5

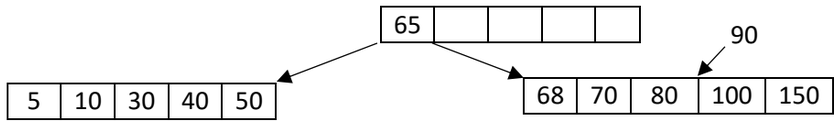


//Al incorporar el 70 a nivel base se trabaja igual que el árbol B. Incorporamos el valor, partimos al medio y sube el valor de la derecha (o el central) dependiendo cantidad de claves.

//Al agregar el valor 40 el nodo está lleno, y hay política de izquierda, pero como no tiene hermano izquierdo por ser extremo mira al hermano derecho. Como tiene lugar se redistribuye.

//Al agregar el valor 5 pasa exactamente lo mismo que en el paso anterior.

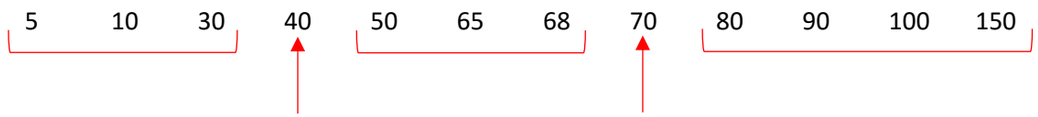
Agrego el 150, 90



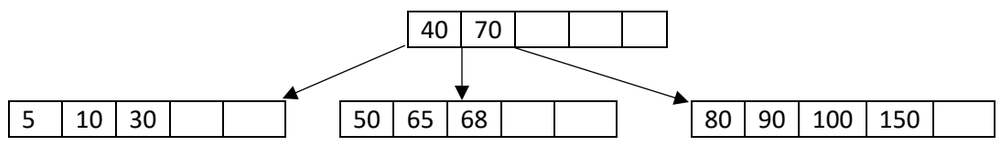
//Al agregar el 90 no hay lugar. Tiene política de izquierda, asique miramos el hermano izquierdo y tampoco tiene lugar. Entonces:

//Armamos la secuencia de valores de los dos nodos llenos incluyendo al padre en común y al valor que queríamos agregar (90).

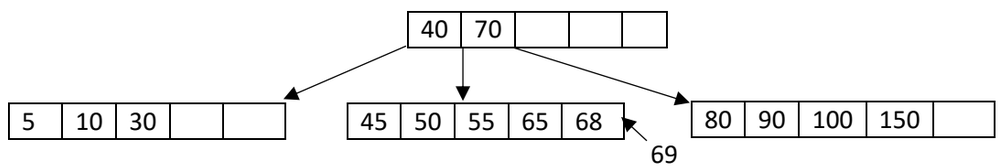
En el siguiente paso necesitamos promocionar 2 claves como padres, miramos la Cant. Min. Claves (en este caso 3), entonces contamos 3 claves de la secuencia de izquierda a derecha y promocionamos el siguiente.



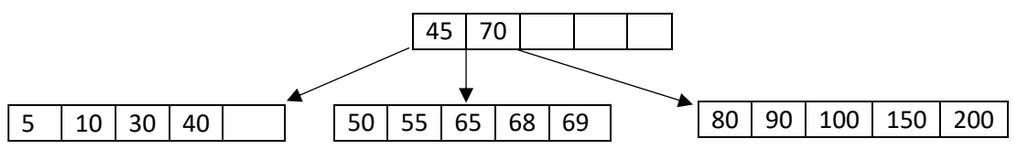
//El ultimo nodo esta agrupado de cuatro claves porque como dijimos se promocionan solo 2 claves como padre y ya promocionamos el 40 y 70.



Agrego el 45, 55, 69

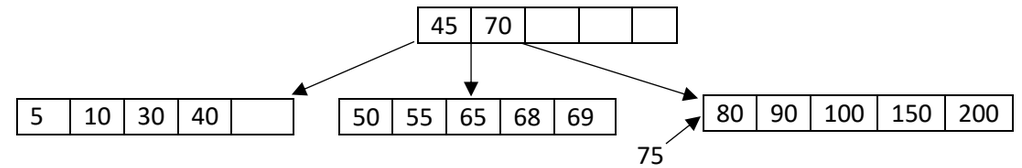


Agrego el 200

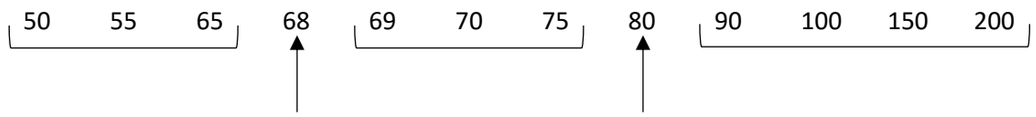


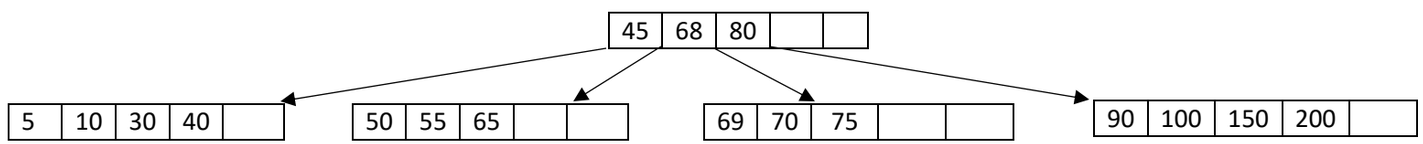
//Al agregar el valor 69 la página está llena, y como hay política de izquierda se mira el hermano izquierdo que tiene lugar, entonces se redistribuye.

Agrego el 75



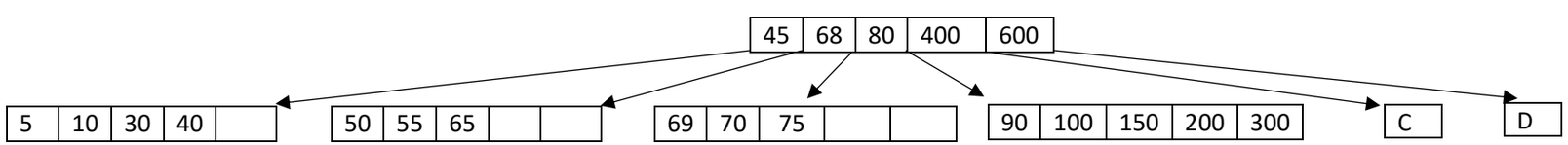
//Al agregar el 75 no hay lugar. Tiene política de izquierda, asique miramos el hermano izquierdo y tampoco tiene lugar. Entonces armamos nuevamente la secuencia de valores y promocionamos 2.



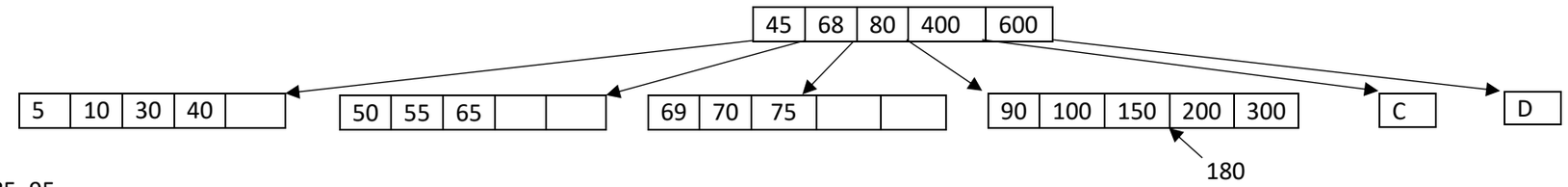


¿Ahora que hacemos si llenamos la raíz? Para eso la completamos agregando el valor 400 y 600.

Agregamos el valor 300

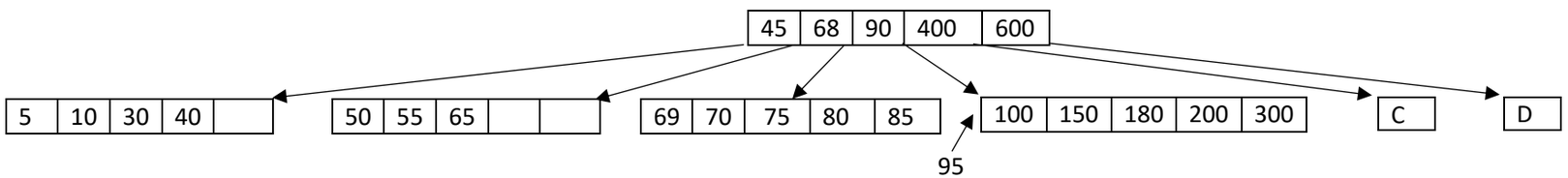


Agrego el 180

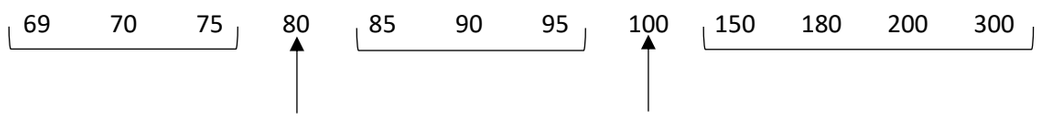


//Al agregar el valor 180 la página está llena, y como hay política de izquierda se mira el hermano izquierdo que tiene lugar, entonces se redistribuye.

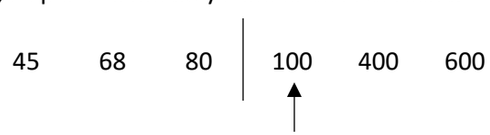
Agrego el 85, 95



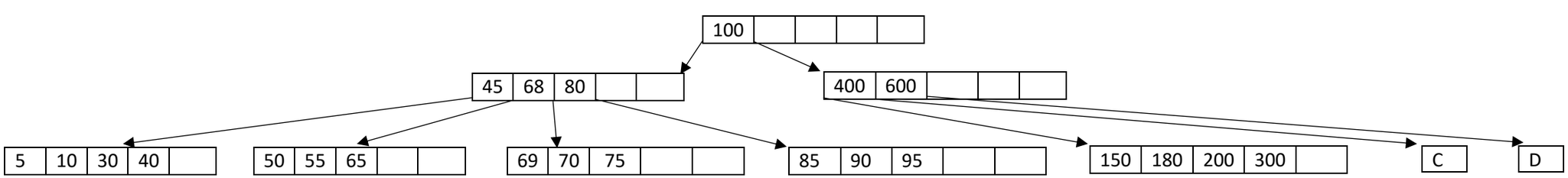
//Al agregar el 95 no hay lugar. Tiene política de izquierda, asique miramos el hermano izquierdo y tampoco tiene lugar. Entonces armamos nuevamente la secuencia de valores y promocionamos 2.



//Al promocionar el valor 80 y 100 el padre no tiene lugar. Entonces se arma la secuencia de valores del nodo padre, se parte al medio y sube el de la derecha.

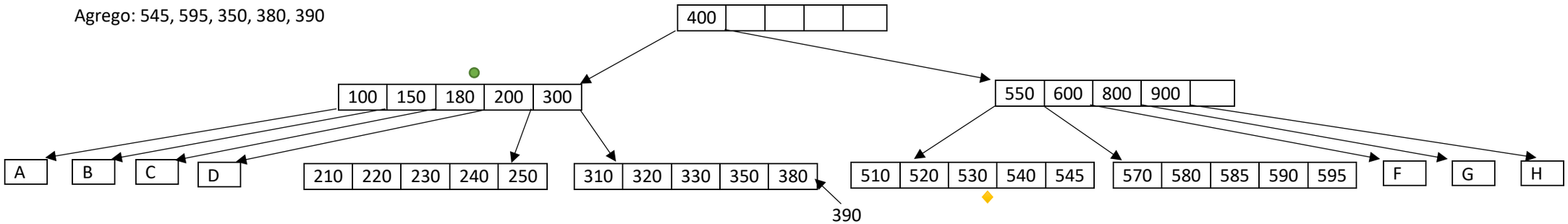


//En esta secuencia no esta el 90 porque en el paso anterior bajó y se promocionó el 80 y 100 (osea ya no pertenece al nodo padre).



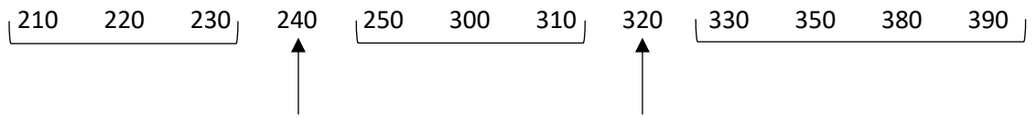
Ultimo ejemplo:

Agrego: 545, 595, 350, 380, 390

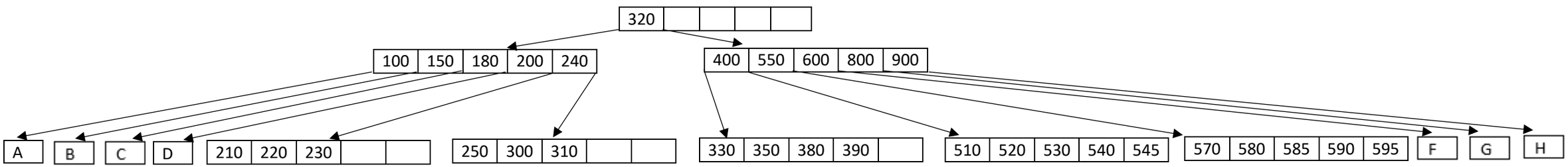


//Al agregar el valor 390 la página no tiene lugar. Estamos con política izquierda entonces miramos el hermano izquierdo que tampoco tiene lugar, entonces armamos la secuencia de claves y promocionamos dos como padre.

//Si la página llena donde queremos insertar un valor fuese ◆ esa página no tiene hermano izquierdo, ya que la página de la izquierda tiene otro padre y no es hermano. Por ende, es un extremo izquierdo y tiene permitido ver su hermano derecho más allá de la política.



//Como el padre no tiene espacio y no es nivel base ● tenemos que mirar nuevamente por la política si tiene hermano izquierdo para pasarle claves, no tiene, pero al ser extremo izquierdo puede mirar su hermano derecho que tiene un lugar, entonces se redistribuye.

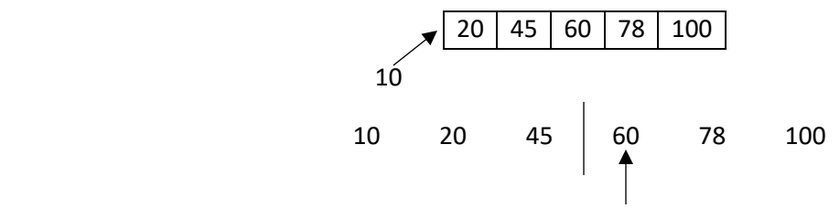


Árbol B+:

- Los datos se guardan a nivel hoja, es decir, las demás paginas están cargadas con valores orientativos para llegar a las hojas. Si busco un valor que es menor a una clave de una página que no es nivel hoja tomamos la rama izquierda y si buscamos un valor que es mayor tomamos la rama derecha.
- Las hojas están enlazadas con "punteros" (⟶).
- La eliminación también se da a nivel hoja.
- Cuando promocionamos algún valor se dice que promocionamos una copia, es decir, el valor que se promociona debe seguir estando a nivel hoja entonces se promociona una copia.

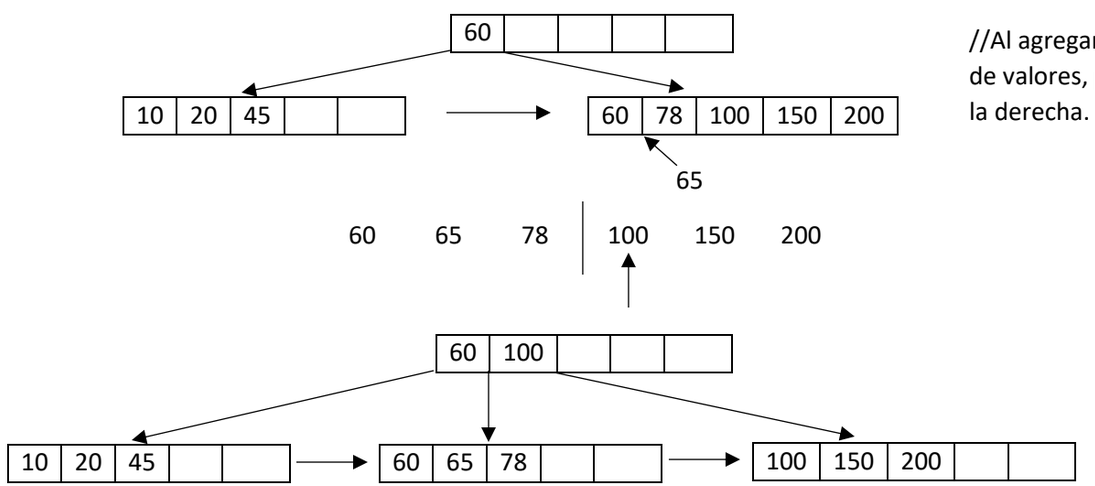
Árbol B+ orden 6:

Agrego: 10



//Al agregar el 10 la página no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos UNA COPIA del valor de la derecha.

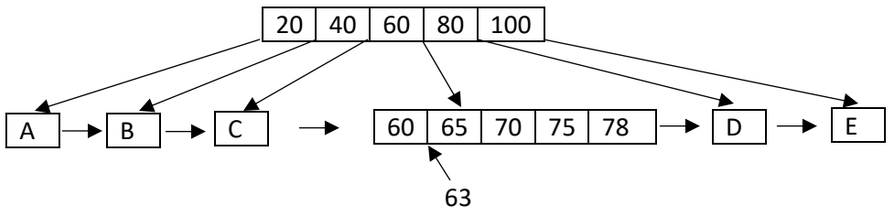
Agrego: 150, 200, 65



//Al agregar el 65 la página no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos UNA COPIA del valor de la derecha.

Suponemos otro ejemplo para mostrar un caso:

Agrego: 63



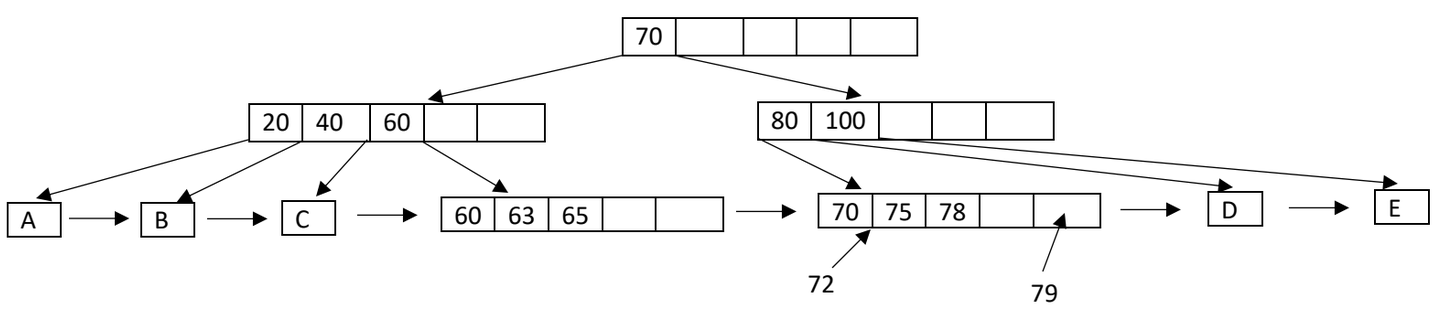
//Al agregar el 63 la página no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos UNA COPIA del valor de la derecha.

60 63 65 | 70 75 78

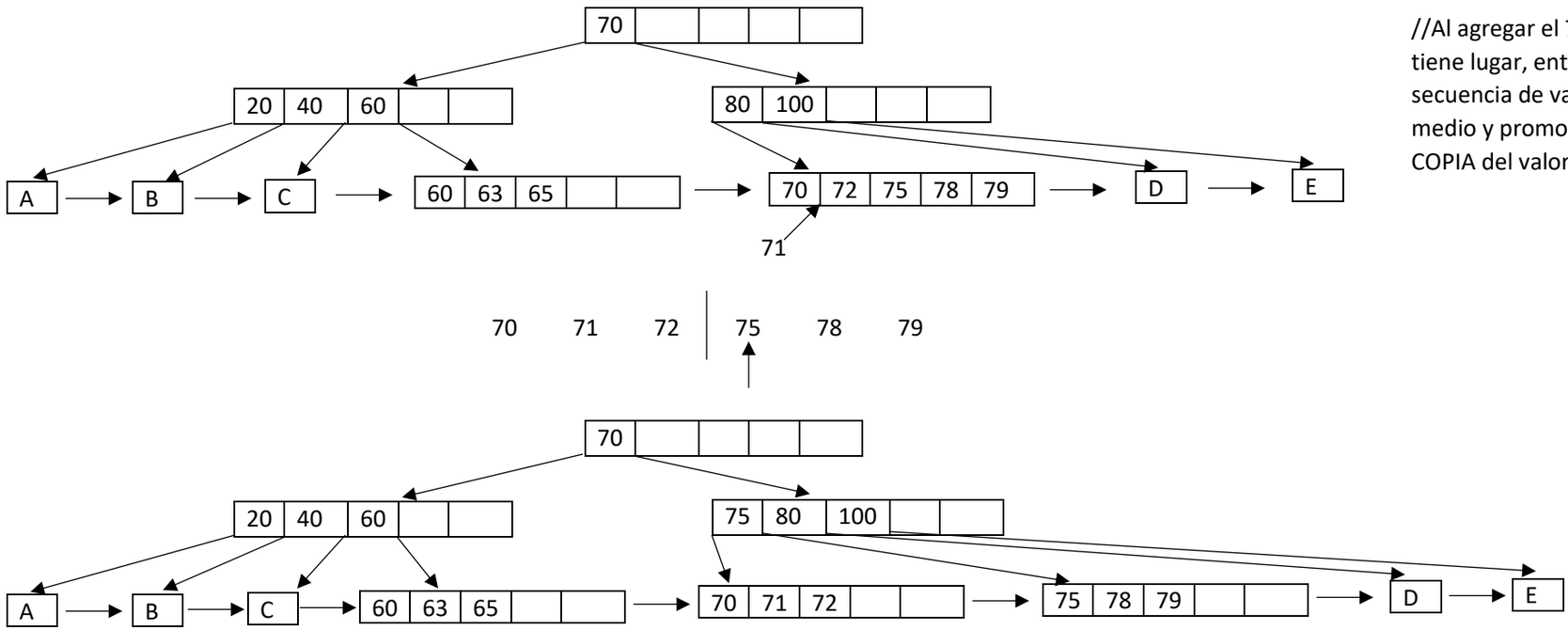
//Al promocionar la copia del 70 la página raíz no tiene lugar, entonces armamos la secuencia de valores de la raíz, partimos al medio y promocionamos el valor de la derecha. NO PROMOCIONAMOS UNA COPIA, PORQUE TODO NIVEL NO HOJA SE COMPORTA COMO ARBOL B.

20 40 60 | 70 80 100

Agrego: 72, 79



Agrego: 71



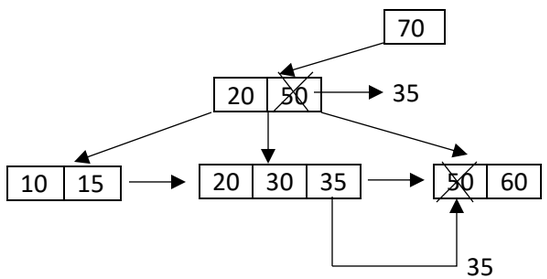
//Al agregar el 71 la página no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos UNA COPIA del valor de la derecha.

Eliminación:

- Si cumple la cantidad mínima de claves se elimina a nivel hoja y listo. Si se repite el valor en otro lado (osea no a nivel hoja) no pasa nada.
- Si al eliminar no se cumple la cantidad mínima de claves se le pide al hermano derecho y sino al izquierdo. Si el hermano le presta se lo pasa directamente y sube una copia del primer valor que quedó reemplazando al anterior.

Ej:

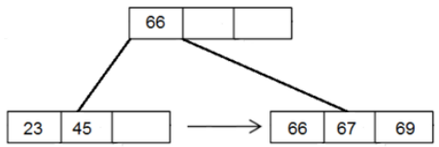
Elimino: 50



- Si no puede prestar ningún hermano se concatena con el hermano derecho, pero solo los hijos sin que baje el padre y el padre se eliminan.
- Si queda con menos de la cantidad mínima de claves un nivel no hoja por esto se concatena, pero acá SI incluimos al padre

Ejercitación Árbol B+:

Dado el siguiente árbol B+ de Orden 4 mostrar cómo quedaría después de las siguientes operaciones:



+120, +110, +52, +70, +15, -45, -52, +22, +19, -66, -22, -19, -23.

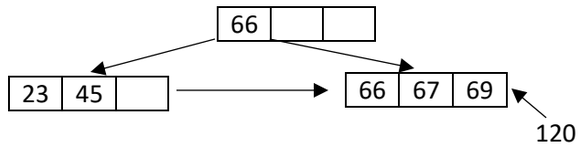
Orden 4

Cant. Max. Claves = **n – 1** = 3

Cant. Min. Claves = $\left\lceil \frac{n}{2} \right\rceil - 1 = 1$

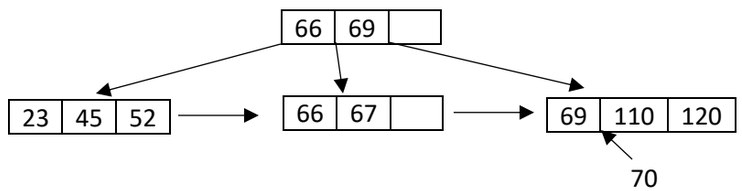
Cant. Min. Hijos = $\left\lceil \frac{n}{2} \right\rceil = \left\lceil \frac{4}{2} \right\rceil = 2$

Cant. Max. Hijos = **n** = 4



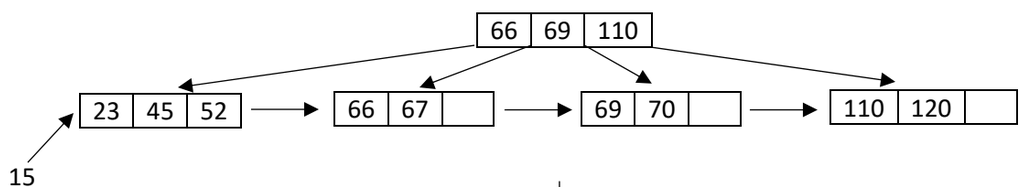
//Al agregar el 120 la página no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos UNA COPIA del valor de la derecha.

66 67 | 69 120



//Al agregar el 70 la página no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos UNA COPIA del valor de la derecha.

69 70 | 110 120

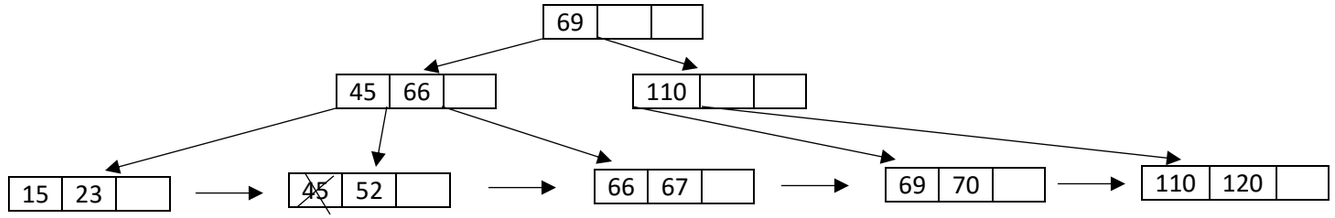


//Al agregar el 15 la página no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos UNA COPIA del valor de la derecha.

15 23 | 45 52

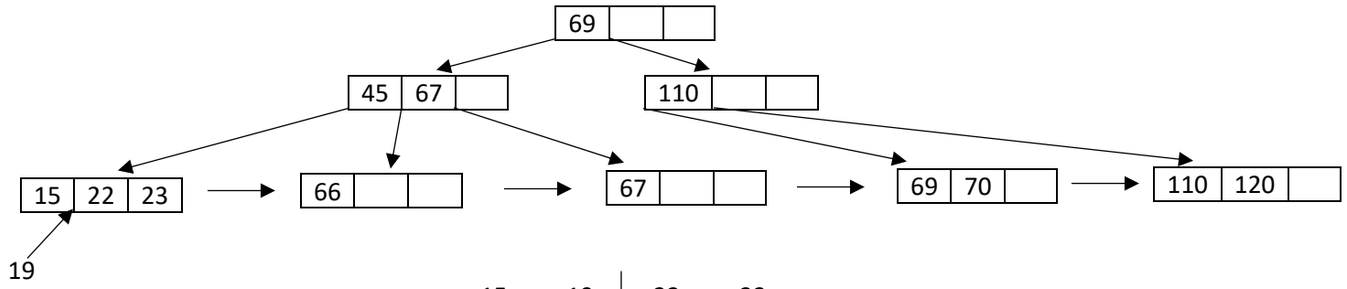
//Al subir una copia del 45 la raíz no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos el valor de la derecha (NO UNA COPIA, PORQUE EN NIVEL NO HOJA SE COMPORTA COMO ARBOL B)

45 66 | 69 110



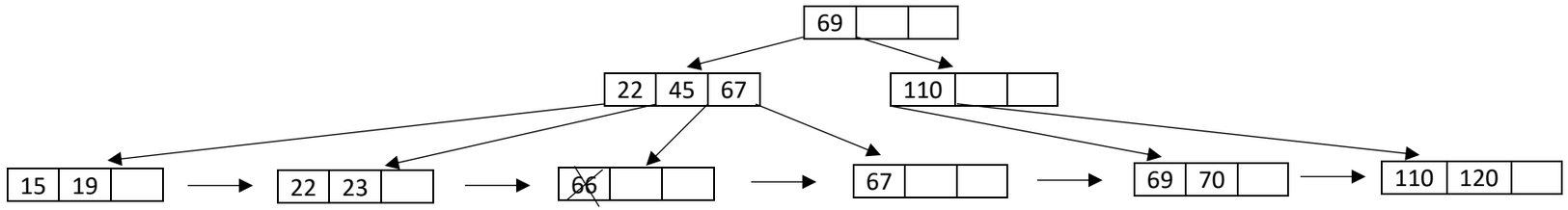
//Al eliminar el 45 se cumple la cant. Minima de claves (1) asique no pasa nada.

//Al eliminar el 52 en el paso de abajo, le pide al hermano derecho. Le pasa uno y se actualiza el padre con una copia del primer valor del hijo derecho.

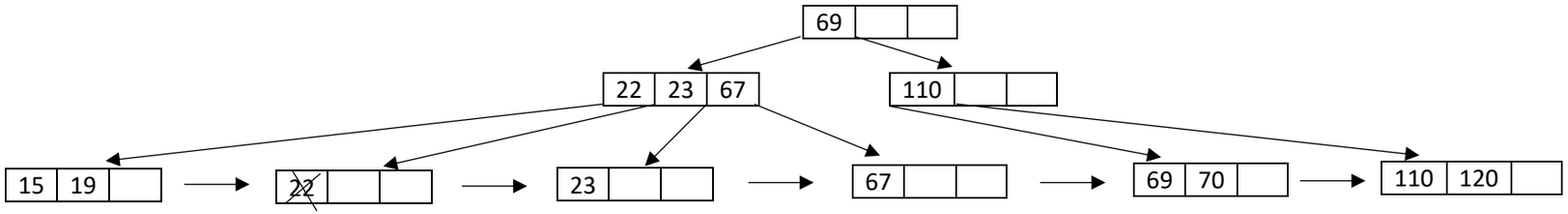


//Al agregar el 19 la página no tiene lugar, entonces armamos la secuencia de valores, partimos al medio y promocionamos UNA COPIA del valor de la derecha.

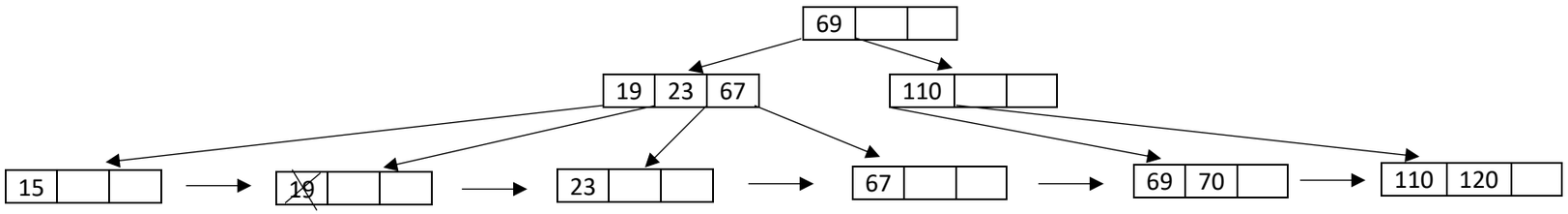
15 19 22 23



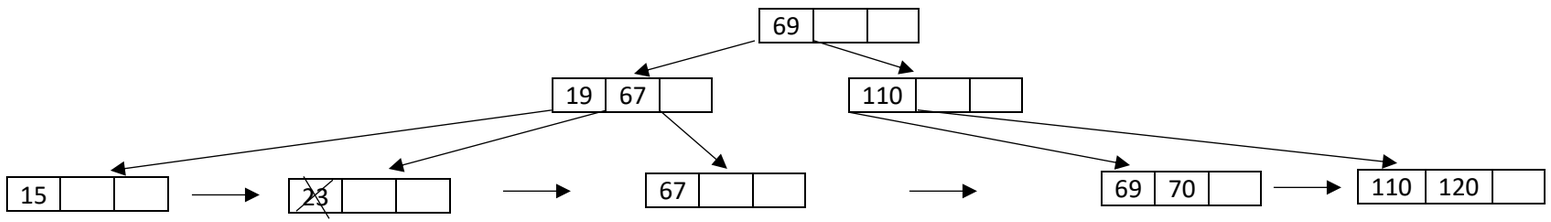
//Al eliminar el 66 no se cumple la cant. Mínima de claves, le pide uno a su hermano derecho, como no tiene le pide uno al hermano izquierdo que se lo pasa directamente y sube una copia del primer valor que quedó reemplazando al anterior.



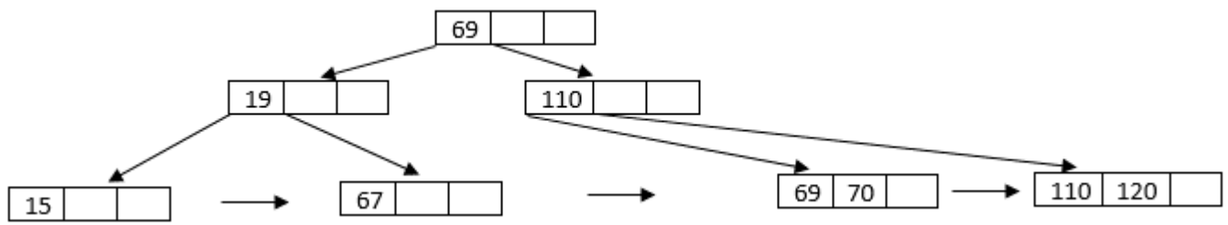
//Al eliminar el 22 no se cumple la cant. Mínima de claves, le pide uno a su hermano derecho, como no tiene le pide uno al hermano izquierdo que se lo pasa directamente y sube una copia del primer valor que quedó reemplazando al anterior.



//Al eliminar el 19 no se cumple la cant. Mínima de claves, le pide uno a su hermano derecho, como no tiene le pide al hermano izquierdo y como tampoco tiene se concatena con el hermano derecho y se elimina el padre.



//Al eliminar el 23 no se cumple la cant. Mínima de claves, le pide uno a su hermano derecho, como no tiene le pide al hermano izquierdo y como tampoco tiene se concatena con el hermano derecho y se elimina el padre.



Algebra Relacional:

Tablas

-Identificador: Es la conocida Clave Primaria (CP) que debe ser un campo único, de haber muchos campos únicos para determinar cuales se elige el que ocupa menor espacio.

Operadores:

- Selección (σ):

  Se emplea para seleccionar un subconjunto de las tuplas (lista ordenada de elementos) que satisfacen un predicado.
  Se puede considerar esta operación como un filtro que mantiene solo las tuplas que satisfacen una determinada condición.

  Ej:

  $$\sigma_{\text{ciudadSuc = 'Madrid' and Activo > 8000}} (\text{Sucursales})$$

  //El renglón de arriba selecciona (σ) de la tabla Sucursales aquellas tuplas que cumplen la condición que el campo ciudadSuc = 'Madrid' y que el campo Activo sea mayor que 8000.

- Proyección (π):

  Permite presentar los atributos especificados en una relación y no presentar el resto.
  Esta operación selecciona ciertas columnas de la tabla y descarta otras.

  Ej:

  $$\pi_{\text{nombre,apellido}} (\text{Socios})$$

  //El renglón de arriba proyecta (es decir, devuelve) solo las columnas que especificamos. En este caso devuelve la columna nombre y la columna apellido de la tabla Socios.

- Renombrar(ρ):

  Define un alias para una relación, es decir, se define un nombre para la devolución de una consulta.

  Ej:

  $$\rho_{\text{Ubicación}}\ (\text{UbicaciónDeptos})$$

  //Se renombra la tabla UbicaciónDeptos a Ubicación.

- Producto Cartesiano (χ):

  Se utiliza para cruzar información.
  El producto cartesiano es una operación que, a partir de dos relaciones vincula cada tupla de una de las relaciones con cada tupla de la otra relación. En otras palabras, cruza todas las columnas de ambas tablas.
  Para evitar ambigüedades se utiliza la sintaxis tabla.columna

  Ej: R χ S

  Numero de columnas de R χ S = Consult. de R + Consult. de S

  Numero de tuplas de R χ S = Tuplas de R * Tuplas de S.

  Ej. Practico:
  Obtener para cada DNI de director los nombres de los proyectos correspondientes.

  //No hay una sola tabla que tenga toda la información entonces tenemos que cruzar las tablas Departamentos y Proyectos, seleccionar aquellos que el número de departamento sea igual al número de departamento pero que tiene proyecto (es decir nroDepto de la tabla proyectos) y por último proyectar lo pedido (el nombre de director y los proyectos).

- Unión ($\cup$):

  Equivale a la unión matemática de conjuntos.
  Las tuplas repetidas son eliminadas.
  Las relaciones a unir deben ser compatibles (es decir, deben tener igual número de atributos (misma cantidad de columnas en las tablas) e igual dominio. El dominio quiere decir que, si tenemos en una tabla DNI de cliente y en otra DNI de empleado, aunque sean diferentes personas el campo sigue teniendo guardado un DNI).

  Para poder utilizar la unión una opción es proyectar por ejemplo la columna nombre de dos tablas diferentes y ahí conseguimos el mismo numero de atributos (ya que quedan dos tablas que tienen una columna que es nombre) y el dominio es el mismo (strings de nombres).

- Diferencia ($-$):

  Equivale a la diferencia matemática de conjuntos.
  Incluye tuplas que están en una relación, pero no en la otra.
  Las relaciones a unir deben ser compatibles en atributos y dominio (lo mismo que en unión).

  Ej:

  R $-$ S incluye las tuplas que están en R y no están en S.

  En otras palabras, si en R hay tuplas en común con S entonces a R esas tuplas se le eliminan.

  Ej Practico:

  Si nos piden nombres de beneficiarios que no son nombres de empleados proyectamos la columna nombre de ambas tablas y ahora que es compatible en atributos (por tener dos tablas con una columna que es nombre) y dominio (porque el campo tiene guardado strings) aplicamos la diferencia (que nos va a eliminar los elementos que están repetidos en ambas tablas). Por ende, se eliminan los nombres de beneficiarios que también son nombres de empleados.

- Producto Natural (⋈):
  Combina los elementos de la primera relacion que se relacionan con los elementos de la segunda relacion.
  Simplifica consultas que combinan varias relaciones.
  Es una reunión con el operador de igualdad para combinar los atributos comunes de las relaciones.
  No especifica explícitamente el predicado de combinación y la selección se hace en base a los campos comunes.

  A ⋈ B

  Es lo mismo que realizar una operación de selección posterior a un producto cartesiano.

  Ej:
  Si tenemos dos tablas que tienen una columna con DNI adjuntos pero una se llama EmpleadoDNI y la otra ClienteDNI aclaramos la selección de campos:

  Departamentos ⋈ EmpleadoDNI = ClienteDNI Empleados

  El renglón de arriba es lo mismo que hacer:

  $\sigma_{EmpleadoDNI = ClienteDNI}$ (Departamentos χ Empleados)

  =

  → Cruza la tabla Departamentos con la tabla Empelados y devuelve una nueva tabla donde se cumple que EmpleadoDNI = ClienteDNI.

- Intersección (∩):
  Equivale a la intersección matemática de conjuntos.
  Incluye tuplas que están en todas las relaciones.

  $R \cap S$
  Incluye las tuplas que están en R y en S.

- Asignación (←):
  Crea una nueva relación a partir de otra.
  Lo que se asigna puede ser una relación existente o el resultado de una operación.

  Serian como variables auxiliares donde se guarda una consulta hecha para poder utilizarla en otra consulta.

- División (÷):
  Retorna aquellos elementos de la primera relación que se relacionan con **TODOS** los elementos de la segunda relación.
  A ÷ B retorna los elementos de A que se relacionan con todos los elementos de B.

- Actualizaciones:
  Altas: A través de la operación de unión (U) y la operación de asignación (←).
  Bajas: A través de la operación de diferencia (-) y la operación de asignación (←).
  Modificación (δ):

$$\delta_{\text{atributo = 'nuevo valor'}}\,(\text{Relación})$$

Ejercicio 1

Clientes = (nroCliente, dni, apellido, nombre, dirección, localidad, teléfono)
Sucursales = (nroSucursal, nombre, dirección, localidad, teléfono)
Cuentas = (nroCuenta, nroCliente, nroSucursal, fechaApertura, tipo, saldo)
Movimientos = (nroMovimiento, nroCuenta, nroSucursal, fecha, hora, tipo, monto)

1) Listar los números de clientes cuya localidad sea 'La Plata'**.**

$$\pi_{\text{nroCliente}}(\sigma_{\text{localidad = 'La Plata'}}(\text{Clientes}))$$

2) Listar los números y nombres de las sucursales de la localidad de 'Olavarria'.

$$\pi_{\text{nroSucursal, nombre}}(\sigma_{\text{localidad = 'Olavarria'}}(\text{Sucursales}))$$

3) Listar los números de documento, apellidos y nombres de los clientes de las sucursales de La Plata.

$$R1 \leftarrow \pi_{\text{nroSucursal}}(\sigma_{\text{localidad = 'La Plata'}}(\text{Sucursales}))$$

$$R2 \leftarrow \pi_{\text{nroCliente}}(\text{Cuentas} \bowtie R1)$$

$$\pi_{\text{dni, apellido, nombre}}(\text{Clientes} \bowtie R2)$$

//En este renglón ponemos el producto natural sin aclarar nada porque se cruza R1 (que contiene nroSucursal) con la tabla Cuentas que también posee nroSucursal. Es decir, se llaman igual los campos que compara

4) Listar los números de cuentas del cliente de apellido ´López´ y nombre ‘Juan’.

$$P1 \leftarrow \pi_{nroCliente} (\sigma_{apellido = 'Lopez' \text{ and } nombre = 'Juan'} (Clientes))$$

$$\pi_{nroCuenta} (Cuentas \bowtie P1)$$

5) Listar los números de cuentas cuya sucursal sea La Plata y tengan saldo mayor a 1000 pesos.

$$L1 \leftarrow \pi_{nroSucursal} (\sigma_{localidad = 'La\ Plata'} (Sucursales))$$

$$L2 \leftarrow \pi_{nroSucursal} (\sigma_{saldo>1000} (Cuentas))$$

$$\pi_{nroCuenta} (L1 \bowtie L2)$$

6) Listar el número de cuenta y el nombre de sucursal de las cuentas tipo ‘CC’ tales que el saldo se encuentre entre $1000 y $2000.

$$M1 \leftarrow \pi_{nroCuenta,\ nroSucursal} (\sigma_{tipo = 'CC' \text{ and } saldo >= 1000 \text{ and } saldo <= 2000} (Cuentas))$$

$$\pi_{M1.nroCuenta,\ Sucursales.nombre} (Sucursales \bowtie M1)$$

7) Listar los números de cuentas del tipo ‘CC’ junto con el dni, apellido y nombre del cliente.

$$K1 \leftarrow \pi_{nroCuenta,\ nroCliente}\left(\sigma_{tipo = 'CC'}(Cuentas)\right)$$

$$\pi_{K1.nroCuenta,\ Clientes.dni,\ Clientes.apellido,\ Clientes.nombre}(Clientes \bowtie K1)$$

8) Listar los movimientos de la cuenta de Pérez Juan.

$$B1 \leftarrow \pi_{nroCliente}\left(\sigma_{apellido = 'Perez' \text{ and } nombre = 'Juan'}(Clientes)\right)$$

$$B2 \leftarrow \pi_{Cuentas.nroCuenta}(Cuentas \bowtie B1)$$

$$\pi_{Movimientos.nroMovimiento,\ Movimientos.nroCuenta,\ Movimientos.nroSucursal,\ Movimientos.fecha,\ Movimientos.hora,\ Movimientos.tipo,\ Movimientos.monto}(Movimientos \bowtie B2)$$

Ejercicio 1
Clientes = (nroCliente, dni, apellido, nombre, dirección, localidad, teléfono)
Sucursales = (nroSucursal, nombre, dirección, localidad, teléfono)
Cuentas = (nroCuenta, nroCliente, nroSucursal, fechaApertura, tipo, saldo)
Movimientos = (nroMovimiento, nroCuenta, nroSucursal, fecha, hora, tipo, monto)

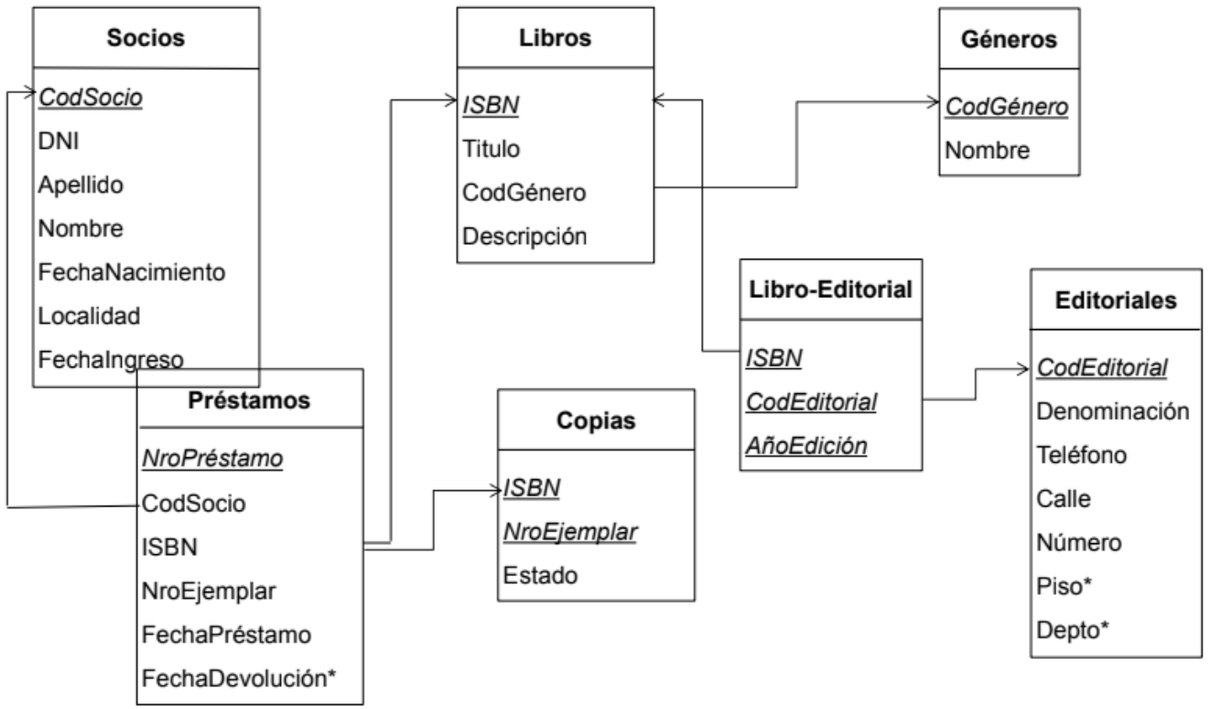
9) Listar la fecha, tipo y monto de los movimientos de la cuenta de García Jorge tal que el monto se encuentre entre 200 y 500 pesos.

$$F1 \leftarrow \pi_{nroCliente}\left(\sigma_{apellido = 'Garcia' \text{ and } nombre = 'Jorge'}(Clientes)\right)$$

$$F2 \leftarrow \pi_{nroCuenta}(Cuentas \bowtie F1)$$

$$F3 \leftarrow (Movimientos \bowtie F2)$$

$$\pi_{F3.fecha,\ F3.tipo,\ F3.monto}\left(\sigma_{F3.monto >= 200 \text{ and } F3.monto <= 500}(F3)\right)$$



1. Obtener el apellido y nombre de los socios que ingresaron a partir de marzo de 2019

$$\pi_{\text{apellido, nombre}} (\sigma_{\text{FechaIngreso >= 01/03/2019}} (\text{Socios}))$$

2. Obtener el título de los libros que se editaron en el año 2015.

$$H1 \leftarrow \pi_{\text{ISBN}} (\sigma_{\text{AñoEdición = 2015}} (\text{Libro-Editorial}))$$

$$\pi_{\text{Libros.Titulo}} (\text{Libros} \bowtie H1)$$

3. Listar el DNI, Apellido y Nombre de aquellos socios que deban libros.

$$A1 \leftarrow \pi_{CodSocio}(\sigma_{FechaDevolución = null}(Prestamos))$$

$$\pi_{Socios.DNI, Socios.Apellido, Socios.Nombre}(Socios \bowtie A1)$$

4. Listar el Titulo, el Nombre del Género y Descripción de aquellos libros editados por la editorial “Ediciones A”.

$$P1 \leftarrow \pi_{CodEditorial}(\sigma_{Denominación = 'Ediciones A'}(Editoriales))$$

$$P2 \leftarrow \pi_{Libro\text{-}Editorial.ISBN}(Libro\text{-}Editorial \bowtie P1)$$

$$P3 \leftarrow \pi_{Libros.Titulo, Libros.Descripción, Libros.CodGenero}(Libros \bowtie P2)$$

$$\pi_{P3.Titulo, P3.Descripción, Generos.nombre}(Generos \bowtie P3)$$

5. Listar el DNI, Apellido y Nombre de aquellos socios que deban libros de la editorial “Ediciones A”.

$$Q1 \leftarrow \pi_{CodEditorial}(\sigma_{Denominación = 'Ediciones A'}(Editoriales))$$

$$Q2 \leftarrow \pi_{Libro\text{-}Editorial.ISBN}(Libro\text{-}Editorial \bowtie P1)$$

$$Q3 \leftarrow \pi_{ISBN,CodSocio}(\sigma_{FechaDevolución = null}(Prestamos))$$

$$Q4 \leftarrow (Q2 \bowtie Q3)$$

$$\pi_{Socios.DNI, Socios.Apellido, Socios.Nombre}(Socios \bowtie Q4)$$

6. Conocer el DNI, nombre y apellido de los socios cuyo DNI es mayor que el DNI del socio con código 1111.

$$W1 \leftarrow \pi_{DNI}(\sigma_{CodSocio = 1111}(Socios))$$

$$\pi_{Socios.DNI,\ Socios.Nombre,\ Socios.Apellido}(\sigma_{Socios.DNI > W1.DNI}(Socios\ \chi\ W1))$$

7. Conocer los códigos y denominaciones de las editoriales cuyos libros <u>son sólo</u> del género “Cocina”.

$$E1 \leftarrow \pi_{CodGénero}(\sigma_{nombre = \text{‘cocina’}}(Géneros))$$

$$E2 \leftarrow \pi_{Libros.ISBN}(Libros \bowtie E1)$$

$$E3 \leftarrow \pi_{CodGénero}(\sigma_{nombre <> \text{‘cocina’}}(Géneros))$$

$$E4 \leftarrow \pi_{Libros.ISBN}(Libros \bowtie E3)$$

$$E5 \leftarrow \pi_{Libro\text{-}Editorial.CodEditorial}(Libro\text{-}Editorial \bowtie E2)$$ //Guardo CodEditorial de los libros que son de Cocina.

$$E6 \leftarrow \pi_{Libro\text{-}Editorial.CodEditorial}(Libro\text{-}Editorial \bowtie E3)$$ //Guardo CodEditorial de los libros que NO son de Cocina.

$$E7 \leftarrow \pi_{Libro\text{-}Editorial.CodEditorial}(E5 - E6)$$ //Guardo CodEditorial de los libros que SON de cocina menos los que NO son de cocina
(Esto es para eliminar las editoriales que hacen algún libro que no es de cocina (ya que dice SOLO genero Cocina))

$$\pi_{Editoriales.CodEditorial,\ Editoriales.Denominación}(Editoriales \bowtie E7)$$

8. Obtener los ISBN de los libros editados por todas las editoriales.

$$Y1 \leftarrow \pi_{CodEditorial}(Editoriales)$$

$$(\pi_{ISBN,\ CodEditorial}(Libro\text{-}Editorial)) \div Y1$$

9. Conocer el apellido y nombre de los socios que <u>únicamente</u> sacaron copias con estado “Regular”.

$$K1 \leftarrow \pi_{ISBN}(\sigma_{Estado = \text{‘Regular’}}(Copias))$$

$$K2 \leftarrow \pi_{Préstamos.CodSocio}(Préstamos \bowtie K1)$$

$$K3 \leftarrow \pi_{ISBN}(\sigma_{Estado <> \text{‘Regular’}}(Copias))$$

$$K4 \leftarrow \pi_{Préstamos.CodSocio}(Préstamos \bowtie K3)$$

$$K5 \leftarrow (K2 - K4)$$ //A los CodSocio = regular le resto los CodSocio <> de regular. Es decir, en K5 tengo solo los CodSocio que sacaron copias en estado regular.

$$\pi_{Socios.apellido,\ Socios.nombre}(Socios \bowtie K5)$$

10. Conocer los títulos de los libros que nunca se prestaron.

$R1 \leftarrow \pi_{ISBN}(Préstamos)$

$R2 \leftarrow \pi_{ISBN}(Libros)$

$R3 \leftarrow (R2 - R1)$ //A los ISBN de todos los libros (R2) le resto los ISBN de los libros que aparecen en Prestamos (R1). Es decir, en R3 guardo los ISBN de libros que no están en préstamos (osea nunca se prestaron)

$\pi_{Libros.titulo}(Libros \bowtie R3)$ //Cruzo la tabla Libros con los libros que nunca se prestaron (comparando los ISBN) y proyecto lo pedido.

11. Incorporar el libro de ISBN 555555, título XXXXX, Código de género 111 y descripción ZZZ.

//Para incorporar un renglón debemos usar el selector UNION (U) y deben ser compatibles (es decir, el renglón a agregar debe tener igual número de atributos (misma cantidad de columnas en las tablas) e igual dominio (tipo de dato de cada una de las columnas).

$$Libros \cup \{(555555, 'XXXXX', 111, 'ZZZ')\}$$

//Como dijimos arriba agregamos datos en todos los campos de la tabla Libros (los datos dados) pero de haber un dato sin especificar deberíamos agregarlo manualmente, aunque sea poniendo null. Por otro lado, hay que observar como los datos de los campos título y descripción tienen '' porque esos datos son strings y necesitamos igual dominio.

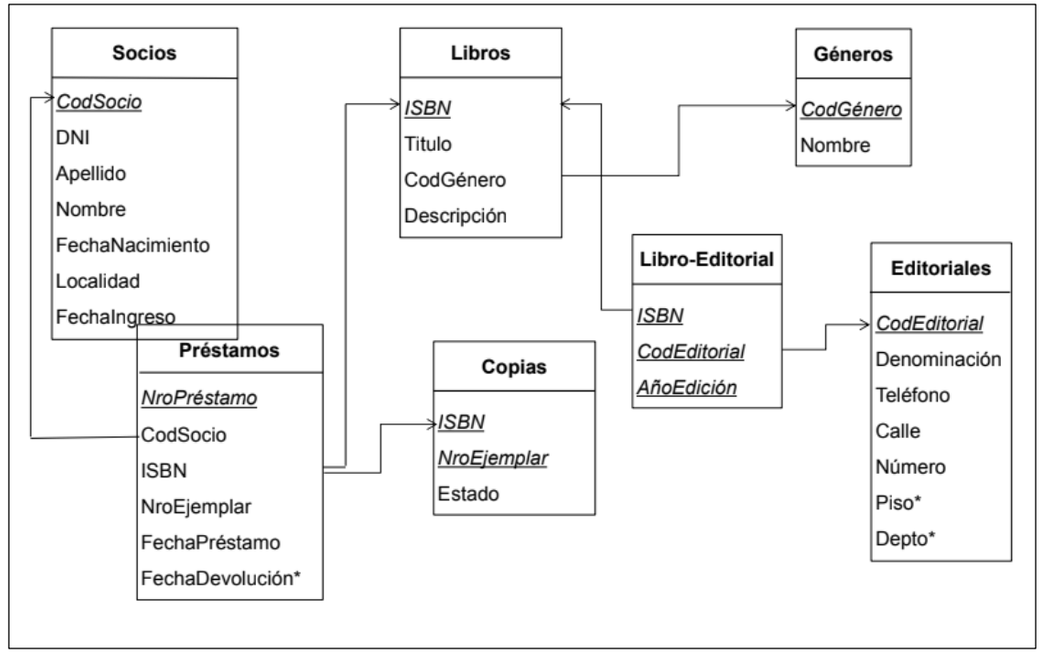
//Una vez cargados los datos lo unimos a la tabla Libros con el selector UNION (U).

12. Eliminar el préstamo de número 999.

//Para eliminar utilizamos el selector de diferencia (-).

$B1 \leftarrow (\sigma_{NroPréstamo = 999}(Préstamos))$

$Prestamos \leftarrow (Préstamos - B1)$



13. Modificar el estado de la copia 8 del ISBN 444444 al estado «YYY»

$Modificar \leftarrow \sigma_{ISBN = 444444 \text{ and } nroEjemplar = 8}(Copias)$

$Copias \leftarrow Copias - Modificar$

$Copias \leftarrow Copias \cup (\delta_{estado = \text{“YYY”}}(Modificar))$

Presentar los idFactura que incluyan únicamente productos de marca M1 y productos con marca M2 (ambas marcas).

Clientes = (idCliente, nombre, dni, localidad)

Productos = (idProducto, nombre, marca, precioUnitario, stock)

Factura = (idFactura, idCliente, fecha, montoTotal)

DetalleFactura = (idFactura, idProducto, cantidad)

$L1 \leftarrow \pi_{idFactura,\ marca}\ (Productos \bowtie DetalleFactura)$

$L2 \leftarrow \pi_{marca}\ (\sigma_{marca = \text{'M1'}\ or\ marca = \text{'M2'}}\ (Productos))$

$L3 \leftarrow \pi_{idFactura}\ (L1 \div L2)$

$L4 \leftarrow \pi_{idFactura}\ (\sigma_{marca <> \text{'M1'}\ or\ marca <> \text{'M2'}}\ (L1))$

$\pi_{idFactura}\ (L3 - L4)$

Resolver la siguiente consulta utilizando el Álgebra Relacional.

Hoteles=(codigoHotel, nombre, añoInauguración, codigoCiudad)

Ciudad=(codigoCiudad, nombre, nombrePais)

Habitaciones=(codigoHabitación, codigoHotel, tipo, capacidad, precioPorDíaActual)

Turistas =(DNI, nombre, apellido, añoNacimiento, codigoCiudad)

TuristasHospedados (codigoHabitación, codigoHotel, DNI, desde, cantidadDías, precioPagadoPorDía)

Eliminar los turistas que se hayan hospedado en todos los hoteles de Argentina y todo lo necesario para mantener la consistencia de las relaciones.

$H1 \leftarrow \pi_{codigoCiudad}(\sigma_{nombrePais = 'Argentina'}(Ciudad))$

$H2 \leftarrow \pi_{codigoHotel}(Hoteles \bowtie H1)$

$H3 \leftarrow \pi_{DNI,\ codigoHotel}(TuristasHospedados)$

$H4 \leftarrow \pi_{DNI}(H3 \div H2)$ //DNI de todos los turistas hospedados en todos los hoteles de Argentina (y quizás algún otro lado)

$H5 \leftarrow \pi_{codigoCiudad}(\sigma_{nombrePais <> 'Argentina'}(Ciudad))$

$H6 \leftarrow \pi_{codigoHotel}(Hoteles \bowtie H5)$

$H7 \leftarrow \pi_{TuristasHospedados.DNI}(TuristasHospedados \bowtie H6)$

$H8 \leftarrow \pi_{DNI}(H4 - H7)$ //Elimino de la lista de turistas hospedados en TODOS los hoteles de Argentina los que se hospedaron en algún lado que no sea Argentina.

$H9 \leftarrow (H8 \bowtie TuristasHospedados)$ //Tabla de todos los datos de turistas hospedados en todos los hoteles.

$TuristasHospedados \leftarrow (TuristasHospedados - H9)$

$H10 \leftarrow (H8 \bowtie Turistas)$ //Tabla de todos los datos de turistas hospedados en todos los hoteles.

$Turistas \leftarrow (Turistas - H10)$

Resolver la siguiente consulta utilizando el Álgebra Relacional.

Hoteles=(codigoHotel, nombre, añoInauguración, codigoCiudad)

Ciudad=(codigoCiudad, nombre, nombrePais)

Habitaciones=(codigoHabitación, codigoHotel, tipo, capacidad, precioPorDíaActual)

Turistas =(DNI, nombre, apellido, añoNacimiento, codigoCiudad)

TuristasHospedados (codigoHabitación, codigoHotel, DNI, desde, cantidadDías, precioPagadoPorDía)

Listar el nombre y apellido de aquellos turistas que únicamente se hayan hospedado en todos los hoteles de Argentina y sólo hayan permanecido por 3 días.

$K1 \leftarrow \pi_{codigoCiudad} (\sigma_{nombrePais = \text{'Argentina'}} (Ciudad))$

$K2 \leftarrow \pi_{Hoteles.codigoHotel} (Hoteles \bowtie K1)$

$K3 \leftarrow \pi_{DNI,\ codigoHotel} (TuristasHospedados)$

$K4 \leftarrow \pi_{DNI} (K3 \div K2)$ //DNI de todos los turistas hospedados en TODOS los hoteles de Argentina (y quizás otro lado)

$K5 \leftarrow \pi_{codigoCiudad} (\sigma_{nombrePais <> \text{'Argentina'}} (Ciudad))$

$K6 \leftarrow \pi_{Hoteles.codigoHotel} (Hoteles \bowtie K5)$

$K7 \leftarrow \pi_{TuristasHospedados.DNI} (TuristasHospedados \bowtie K6)$

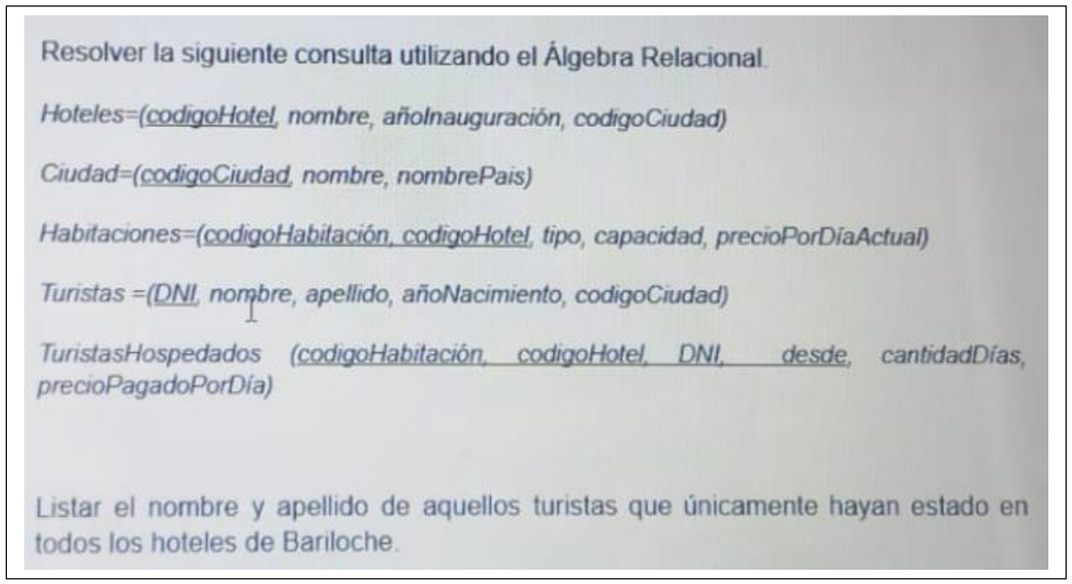
$K8 \leftarrow \pi_{DNI} (K4 - K7)$ //Elimino de la lista de turistas hospedados en TODOS los hoteles de Argentina los que se hospedaron en algún lado que no sea Argentina.

$K9 \leftarrow \pi_{DNI} (\sigma_{cantidadDias = 3} (TuristasHospedados \bowtie K8))$

$K10 \leftarrow \pi_{DNI} (\sigma_{cantidadDias <> 3} (TuristasHospedados \bowtie K8))$

$K11 \leftarrow \pi_{DNI} (K9 - K10)$ //Elimino de la lista de turistas hospedados en TODOS los hoteles de Argentina los que permanecieron <> de 3 días.

$\pi_{Turistas.nombre,\ Turistas.apellido} (Turistas \bowtie K11)$



Resolver la siguiente consulta utilizando el Álgebra Relacional.

Hoteles=(codigoHotel, nombre, añoInauguración, codigoCiudad)

Ciudad=(codigoCiudad, nombre, nombrePais)

Habitaciones=(codigoHabitación, codigoHotel, tipo, capacidad, precioPorDíaActual)

Turistas =(DNI, nombre, apellido, añoNacimiento, codigoCiudad)

TuristasHospedados (codigoHabitación, codigoHotel, DNI, desde, cantidadDías, precioPagadoPorDía)

Listar el nombre y apellido de aquellos turistas que únicamente hayan estado en todos los hoteles de Bariloche.

$N1 \leftarrow \pi_{codigoCiudad} (\sigma_{nombre = \text{'Bariloche'}} (Ciudad))$

$N2 \leftarrow \pi_{Hoteles.codigoHotel} (Hoteles \bowtie N1)$

$N3 \leftarrow \pi_{DNI,\ codigoHotel} (TuristasHospedados)$

$N4 \leftarrow \pi_{DNI} (N3 \div N2)$

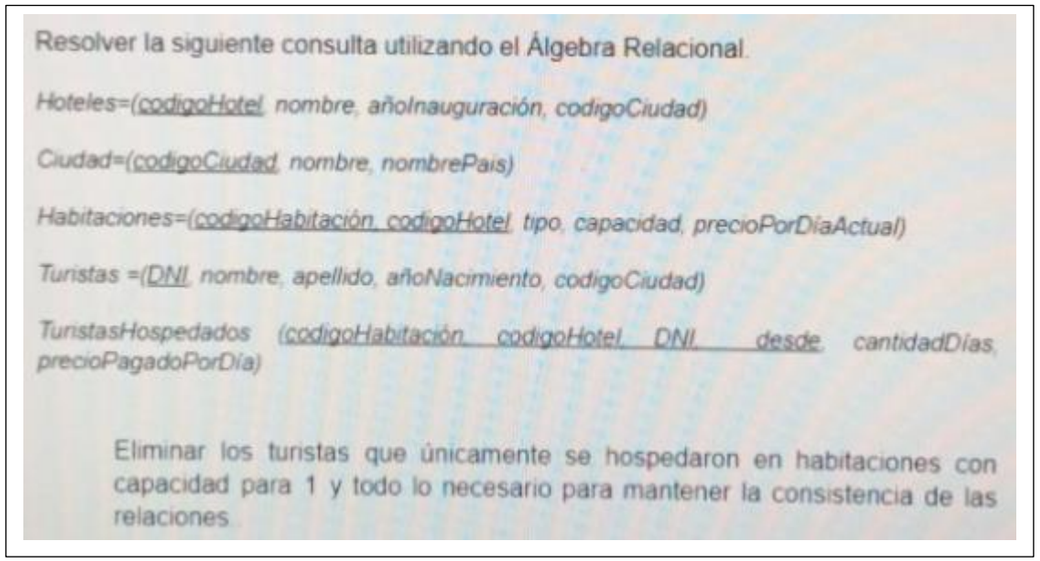
$N5 \leftarrow \pi_{codigoCiudad} (\sigma_{nombre <> \text{'Bariloche'}} (Ciudad))$

$N6 \leftarrow \pi_{Hoteles.codigoHotel} (Hoteles \bowtie N5)$

$N7 \leftarrow \pi_{DNI} (TuristasHospedados \bowtie N6)$

$N8 \leftarrow \pi_{DNI} (N4 - N7)$

$\pi_{Turistas.nombre,\ Turistas.apellido} (Turistas \bowtie N8)$



Resolver la siguiente consulta utilizando el Álgebra Relacional.

Hoteles=(codigoHotel, nombre, añoInauguración, codigoCiudad)

Ciudad=(codigoCiudad, nombre, nombrePais)

Habitaciones=(codigoHabitación, codigoHotel, tipo, capacidad, precioPorDíaActual)

Turistas =(DNI, nombre, apellido, añoNacimiento, codigoCiudad)

TuristasHospedados (codigoHabitación, codigoHotel, DNI, desde, cantidadDías, precioPagadoPorDía)

Eliminar los turistas que únicamente se hospedaron en habitaciones con capacidad para 1 y todo lo necesario para mantener la consistencia de las relaciones.

$P1 \leftarrow \pi_{codigoHotel}(\sigma_{capacidad = 1}(Habitaciones))$

$P2 \leftarrow \pi_{TuristasHospedados.DNI}(TuristasHospedados \bowtie P1)$

$P3 \leftarrow \pi_{codigoHotel}(\sigma_{capacidad <> 1}(Habitaciones))$

$P4 \leftarrow \pi_{TuristasHospedados.DNI}(TuristasHospedados \bowtie P3)$

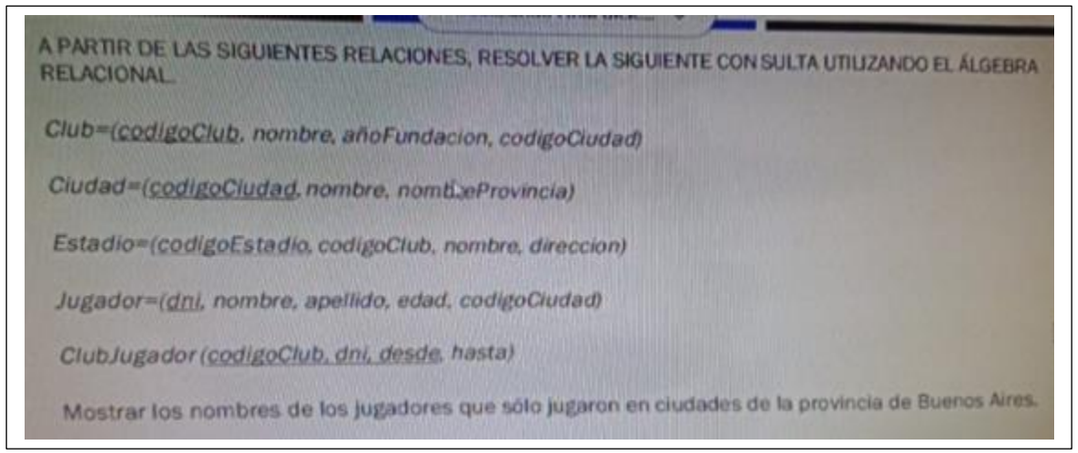
$P5 \leftarrow \pi_{DNI}(P2 - P4)$ //DNI de Turistas que solo se hospedaron en habitaciones con capacidad para 1.

$P6 \leftarrow (TuristasHospedados \bowtie P5)$

$TuristasHospedados \leftarrow (TuristasHospedados - P6)$

$P7 \leftarrow (Turistas \bowtie P5)$

$Turistas \leftarrow (Turistas - P7)$



$Y1 \leftarrow \pi_{codigoCiudad} (\sigma_{nombreProvincia = \text{'Buenos Aires'}} (Ciudad))$

$Y2 \leftarrow \pi_{Jugador.DNI} (Jugador \bowtie Y1)$

$Y3 \leftarrow \pi_{codigoCiudad} (\sigma_{nombreProvincia <> \text{'Buenos Aires'}} (Ciudad))$

$Y4 \leftarrow \pi_{Jugador.DNI} (Jugador \bowtie Y3)$

$Y5 \leftarrow \pi_{DNI} (Y2 - Y4)$ //Me quedo con los jugadores que SOLO jugaron en equipos de Bs As.

$\pi_{Jugador.nombre} (Jugador \bowtie Y5)$

A partir de las siguientes relaciones, resolver la siguiente consulta utilizando el álgebra relacional.

Club=(codigoClub, nombre, añoFundacion, codigoCiudad)

Ciudad=(codigoCiudad, nombre)

Estadio=(codigoEstadio, codigoClub, nombre, direccion)

Jugador=(dni, nombre, apellido, edad, codigoCiudad)

ClubJugador(codigoClub, dni, desde, hasta)

Agregar el club "Estrella de Tandil" con codigoClub 1234, que se fundó en 1921 y que pertenece a la ciudad de Tandil. Asumir que el codigoClub 1234 no existe en la tabla Club.

$T1 \leftarrow \pi_{codigoCiudad} (\sigma_{nombre = 'Tandil'} (Ciudad))$

$Club \leftarrow Club \cup \{(1234, 'Estrella\ de\ Tandil', 1921, T1)\}$

Resolver la siguiente consulta utilizando el Álgebra Relacional.

Modificar la localidad del socio 444 con la localidad del socio 666.

Préstamos = (codPréstamo, fecha, ISBN, nroSocio)

Libros = (ISBN, título, editorial, autor, tema)

Socios = (nroSocio, nombre, localidad)

$T1 \leftarrow \sigma_{nroSocio = 444} (Socios)$

$T2 \leftarrow \pi_{nroSocio,\ nombre} (\sigma_{nroSocio = 444} (Socios))$

$T3 \leftarrow \pi_{localidad} (\sigma_{nroSocio = 666} (Socios))$

$T4 \leftarrow (T2\ \chi\ T3)$

$Socios \leftarrow Socios - T1$

$Socios \leftarrow Socios \cup (T3)$

$//T1 \leftarrow \sigma_{nroSocio = 666} (Socios)$

$T2 \leftarrow \sigma_{nroSocio = 444} (Socios)$

$Socios \leftarrow (Socios - T2)$

$//(Socios \bowtie T1)$

$Socios \leftarrow Socios \cup \delta_{localidad = LA\ DEL\ SOCIO\ 666} (T2)$

Resolver la siguiente consulta utilizando el Álgebra Relacional.

Eliminar los libros que únicamente fueron prestados a todos los socios de la localidad de La Plata y todo lo necesario para mantener la consistencia de las relaciones.

Préstamos = (codPréstamo, fecha, ISBN, nroSocio)

Libros = (ISBN, título, editorial, autor, tema)

Socios = (nroSocio, nombre, localidad)

$J1 \leftarrow \pi_{nroSocio}(\sigma_{localidad = \text{'La Plata'}}(Socios))$

$J2 \leftarrow \pi_{ISBN,\ nroSocio}(Prestamos)$

$J3 \leftarrow \pi_{ISBN}(J2 \div J1)$

$J4 \leftarrow \pi_{nroSocio}(\sigma_{localidad <> \text{'La Plata'}}(Socios))$

$J5 \leftarrow \pi_{Prestamos.ISBN}(Prestamos \bowtie J4)$

$J6 \leftarrow (J3 - J5)$

$J7 \leftarrow (Prestamos \bowtie J6)$

$Prestamos \leftarrow (Prestamos - J7)$

$J8 \leftarrow (Libros \bowtie J6)$

$Libros \leftarrow (Libros - J8)$

Resolver la siguiente consulta utilizando el Álgebra Relacional.

Incorporar la venta realizada el "10/06/2021" para el cliente con dni 1234, por un monto total de $ 2000 de 3 productos con id 1111 y 2 productos con id 6666 para la factura con id 4545.

Clientes = (idCliente, nombre, dni, localidad)

Productos = (idProducto, nombre, marca, precioUnitario, stock)

Factura = (idFactura, idCliente, fecha, montoTotal)

DetalleFactura = (idFactura, idProducto, cantidad)

Ejercicio 1

Resolver la siguiente consulta utilizando el Álgebra Relacional.

Hoteles=(codigoHotel, nombre, añoInauguración, codigoCiudad)

Ciudad=(codigoCiudad, nombre, nombrePais)

Habitaciones=(codigoHabitación, codigoHotel, tipo, capacidad, precioPorDíaActual)

Turistas =(DNI, nombre, apellido, añoNacimiento, codigoCiudad)

TuristasHospedados (codigoHabitación, codigoHotel, DNI, desde, cantidadDías, precioPagadoPorDía)

Listar los nombres de los hoteles donde todos los turistas se hospedaron solamente 2 días.