# Listas normalizadas para tratamiento de valores periodificados

Marcos Barro

https://www.linkedin.com/in/marcos-barro-rico

# *Contenido*

## El problema de la periodificación de valores

Cuando en un desarrollo queremos tratar una entidad de forma unívoca y pensamos en su naturaleza, determinamos su clave de identificación; aquella propiedad o propiedades que, de forma precisa, nos identifica el valor de una instancia concreta de dicha entidad. Con ello, podemos definir métodos y procesos que nos permitan tratar esa entidad de forma coherente. Si estamos en un contexto de BD, inmediatamente pensamos en cuál sería su clave primaria (PK), o si pensamos en un objeto java, podríamos, según el caso, considerar su clave hash.

Esta tarea no suele ser complicada; desde el punto de vista teórico siempre se puede encontrar un conjunto de propiedades que identifican una entidad (en caso contrario no sería una entidad), y en un ámbito más práctico, suele haber mecanismos que nos simplifican el problema, como por ejemplo, en contextos de BD, el uso de columnas auto numéricas.

Sin embargo, en el modelado de sistemas nos solemos encontrar con una casuística recurrente que nos puede complicar esta tarea: la **periodificación de entidades** o valores de entidades; una forma de **especialización de una entidad en espacios temporales acotados**. La entidad deja de ser *lo que es* de forma general para ser *lo que es en un período determinado*.

Pensemos en una entidad LIBRO, identificable por un valor de ISBN. Si estamos en un sistema que gestiona una biblioteca, podríamos pensar en una entidad PRESTAMO como una entidad que extiende los valores de la entidad LIBRO con un PERIODO, es decir, con un inicio y un fin en el tiempo. Dicha entidad periodificada podrá tener valores adicionales, como el USUARIO de la biblioteca, que es el que disfruta el préstamo, aunque para la problemática que nos ocupa, esto es secundario.

En este caso, un PRESTAMO, en lo que se refiere a la periodificación no es problemático, básicamente porque el *valor* que toma en el período (el LIBRO, o en última instancia el ISBN) no se puede solapar en la dimensión temporal (un determinado LIBRO, no se puede prestar en el mismo período). Al darse esta circunstancia, determinar de forma unívoca un PRESTAMO, no tiene mucha complicación; bastaría considerar la identificación del LIBRO (el ISBN) y el PERIODO (el instante de inicio del período sería suficiente, al no existir solapamientos). Es decir, un PRESTAMO podría identificarse unívocamente por lo valores de ISBN e inicio del período.

Sin embargo, en muchos casos, se dan situaciones en las que sí se pueden producir **solapamientos**, y en los que **el valor de la entidad varía** cuando estos solapamientos se producen, lo cual nos da lugar a un problema importante a la hora de identificar unívocamente la entidad periodificada.

Supongamos un sistema de gestión de proyectos en el que tenemos recursos, cada uno de ellos con un coste asociado. Para el ejemplo, pensemos únicamente en una entidad COSTE (€). Supongamos también que, si en un instante de tiempo **t** tenemos **n** costes:

$$C_1, C_2, \dots, C_n$$

entonces el coste total en **t** es la suma:

$$C(t) = C_1 + C_2 + \cdots + C_n$$

En otros casos de uso podría no ser la suma, sino el máximo, el mínimo, una suma con un umbral, o cualquier otra fórmula que se nos ocurra definida sobre esos n costes.

Siguiendo el planteamiento expuesto, si periodificamos la entidad COSTE, podemos tener por ejemplo un "COSTE PERIODO" para el período $P_1 = (9, 12)$ entre las 9 y las 12 horas de una fecha determinada con un valor de 300€ :

$$C(P_1) = 300$$

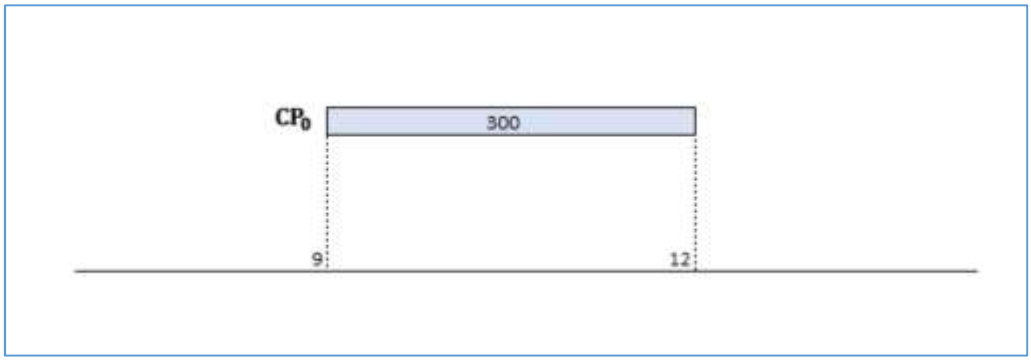
La entidad o valor periodificado ("COSTE PERIODO") se podría expresar de forma general con una terna con el inicio del período (**i**), el fin del período (**f**) y el valor (**v**):

$$CP = (i, f, v)$$

De modo que el valor antes indicado, de 300 € para el período (9,12) podría expresarse como:

$$CP_0 = (9, 12, 300)$$

Visualmente:



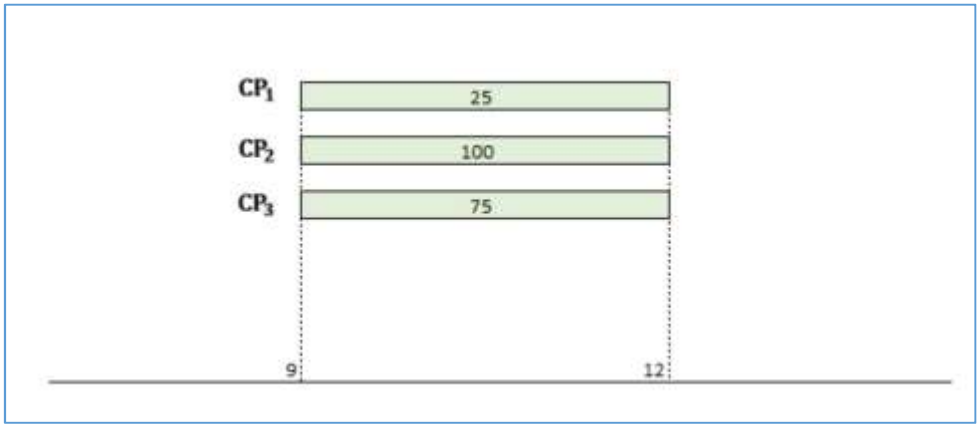

En principio, parece que estos tres valores podrían identificar unívocamente a $CP_0$, y técnicamente podríamos crear un valor que la identifique; una PK, o un hash, según el ámbito de desarrollo), pero eso no nos serviría, ya que, si tenemos por ejemplo los siguientes costes período:

$$CP_1 = (9, 12, 25)$$

$$CP_2 = (9, 12, 100)$$

$$CP_3 = (9, 12, 75)$$

Es decir:



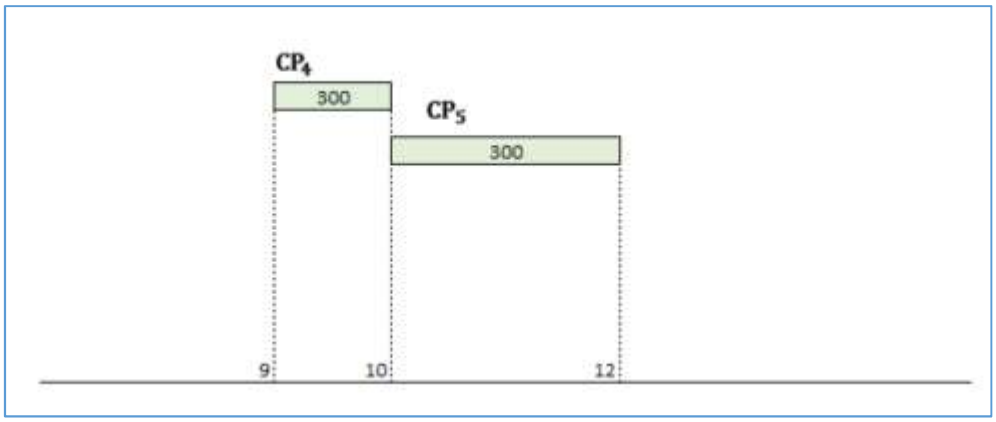

Resulta que estos tres valores periodificados está definidos en el mismo período $(9, 12)$, y su suma es $25 + 100 + 75 = 300$, por lo que la existencia del conjunto $\{ CP_1 , CP_2 , CP_3 \}$ es equivalente a $CP_0$,

También se puede jugar con los períodos en sí (en vez del *valor*), y considerar, por ejemplo:

$$CP_4 = (9, 10, 300)$$

$$CP_5 = (10, 12, 300)$$

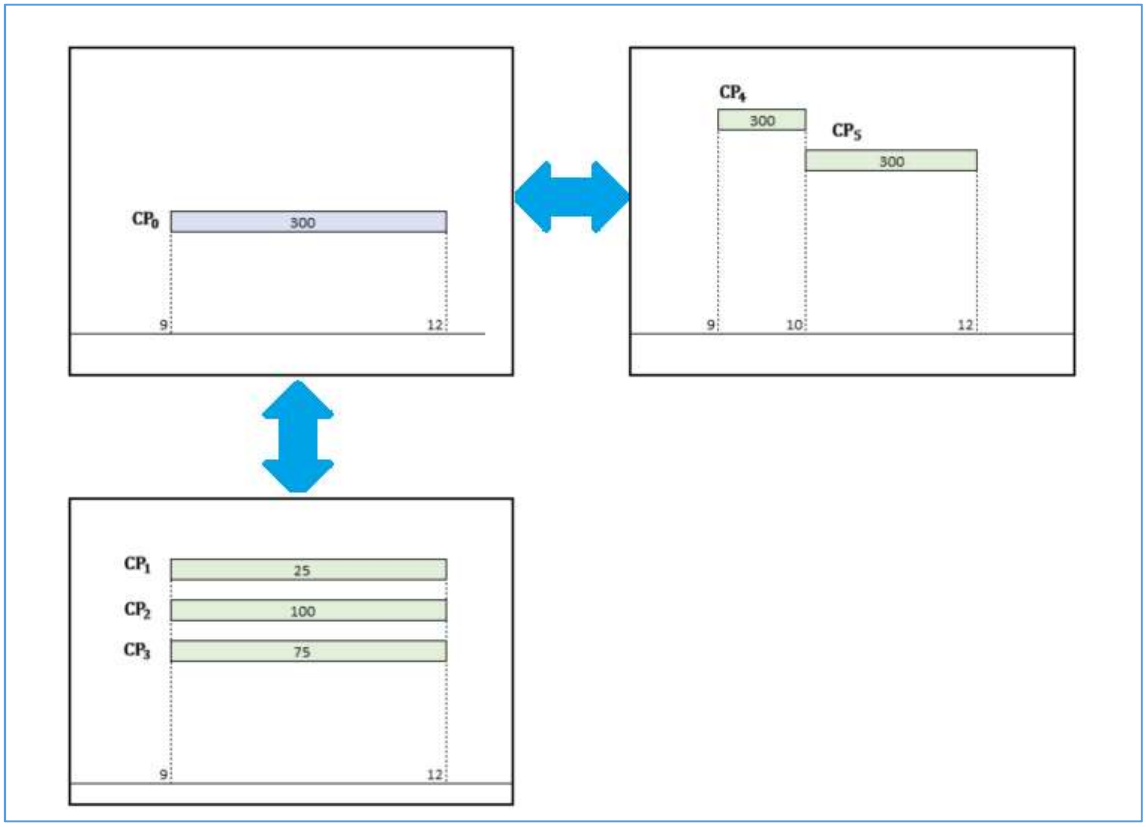
Que sería, visualmente;




Es decir, tendríamos un coste de 300 entre las 9 y las 10, y otro de 300 entre las 10 y las 12, lo cual es, de forma efectiva, lo mismo que un coste de 300 entre las 9 y las 12.

Para el ejemplo expuesto, tendríamos la equivalencia:

$$\{ CP_0 \} = \{ CP_1 , CP_2 , CP_3 \} = \{ CP_4, CP_5 \}$$




En un caso como éste, **no es posible determinar una clave unívoca para identificar un valor**, dado que ese valor se puede expresar de múltiples formas.

Esta problemática llevada a una base de datos, por ejemplo, supondría tener una tabla en la que no sería posible establecer una clave primaria *efectiva* al 100%, dado que los valores de una fila siempre podrían registrarse con otras N equivalentes; con una PK se puede garantizar que no haya filas *iguales*, pero no que no haya un conjunto de filas *equivalentes*. El uso de períodos introduce una dimensión adicional en los valores de la entidad, que hace que las equivalencias no puedan ser restringidas por mecanismos convencionales.

Por otro lado, y al margen de que el desarrollo de un artefacto en contextos como éste se implementen funciones eficientes que gestionen los valores correctamente en los períodos, el problema de base que suele complicar la solución es que no se tiene en cuenta precisamente la posibilidad de esta **variabilidad** para expresar un mismo valor. Evitar esto mediante una **estructura normalizada** es la base para poder abordar el problema de una forma óptima, dado que evita la **explosión de casuísticas** que se pueden dar, y permite orientar la solución de una

forma directa. El planteamiento de una solución de este tipo, mediante una estructura de datos normalizada, es lo que se detallará en los siguientes apartados.

Algunos ejemplos de sistemas en los que puede plantearse el problema de valores o entidades periodificadas:

- **Cálculos de costes** en un período de tiempo en el que se dan distintos costes parciales, cada uno en períodos distintos dentro del período a calcular, y que hay que integrar en su conjunto.

- **Gestión de agendas**, en las que sea necesario determinar períodos de disponibilidad para determinados recursos, en función de tareas de distinta naturaleza asignadas a cada uno de ellos en períodos de tiempo concretos.

- **Cálculo de bonificaciones** en un sistema de facturación, en función de tipos de consumo por periodos para los clientes.

- **Control de tiempos** de empleados; registro de presencia, planificación de turnos, ausencias…

- Etc.

Aunque se trate de sistemas de distinta naturaleza, todos ellos comparten el tratamiento de valores en periodos, y pueden por tanto implementarse usando una misma base común que permita construir la solución sobre un código sólido y que pueda ser validado previamente de forma unitaria.

## Una solución a la periodificación de valores

Con independencia de la tecnología a usar, e incluso de la codificación específica en cada uno de los pasos, en el este apartado se mostrará una **solución genérica** mediante el planteamiento de una **estructura de datos** que permita gestionar esta problemática de forma **unificada**.

El objetivo es definir una estructura que permita gestionar **valores periodificados**, es decir, valores de una entidad en períodos temporales, **que se puedan combinar** entre sí (en base a una lógica predefinida que determine el valor efectivo en caso de solapamiento), mediante un **tratamiento uniforme**, es decir, en el cual un determinado período valor, por construcción, sólo pueda ser especificado de una única forma (**normalizada**).

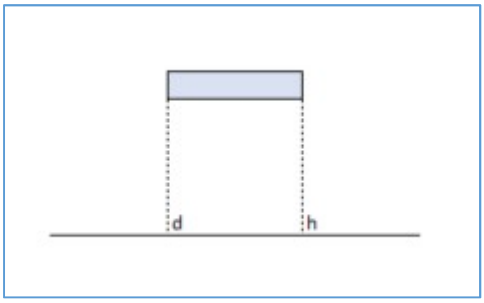
Aunque algunos puntos que se describirán a continuación puedan parecer obvios, se especifican los conceptos tal y como se debe considerar para poder definir la estructura de datos final e implementarla en un sistema real. Algunos matices aparentemente irrelevantes son totalmente necesarios en este planteamiento de la solución.

## Los períodos

Un período es un espacio temporal con un instante de comienzo y otro de finalización. En los siguientes apartados se detallan los conceptos necesarios para poder definir la solución para la periodificación de valores mediante listas normalizadas.

### Definición de período

Se define un período temporal **P** como un par **(d, h)** en el que se cumple: **h > d**.

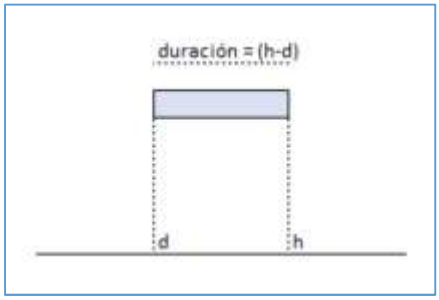



*Período P(d,h)*

Los valores **d** y **h** representan el instante inicio del período (*desde*) y fin del período (*hasta*).

### Duración de un período

La **duración** de un período (el tiempo de duración) viene dado por el valor: **h - d**.




*Duración de un período*

Un período **no puede tener duración cero**; no puede ser que: h = d.

## Magnitud temporal de los períodos

Los valores **d** y **h** se consideran valores de una **magnitud continua** (no discreta), con independencia de que en una implementación dada se utilicen tipos de datos discretos (por ejemplo, Integer en caso de Java, para representar un valor de *timestamp*). Lo que hay que considerar es que *conceptualmente,* por *definición,* no hay un valor de ***t2*** que sea "*el siguiente a* ***t1***". Si en algún caso de uso se necesita trabajar con una dimensión temporal discreta (por ejemplo, períodos de días sin considerar el componente HH:MM:SS), siempre se podrá implementar una solución sobre períodos continuos, tal y como se define aquí.

## Instantes de un período

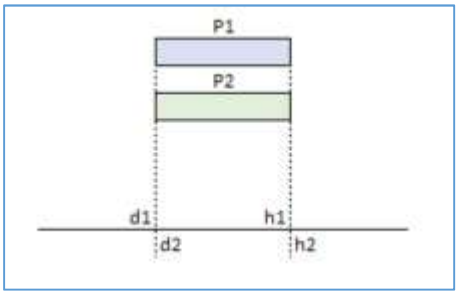
Se considera que un instante $\boldsymbol{t}$ pertenece a un período $\boldsymbol{P(d,h)}$ si, y sólo si $\boldsymbol{d \leq t < h}$. En un período $\boldsymbol{P(d,h)}$, por tanto, $\boldsymbol{d \in P}$ pero $\boldsymbol{h \notin P}$. Esto es por convención; podría haberse considerado al revés. Se asume los períodos como intervalos cerrados por la izquierda (límite por la izquierda incluido) y abiertos por la derecha (límite por la derecha no incluido).

$$t \in P(d,h) \Leftrightarrow t \geq d \wedge t < h$$

## Períodos iguales

Dos períodos P1 y P2 son iguales si, y sólo si, sus instantes de inicio y fin son iguales.

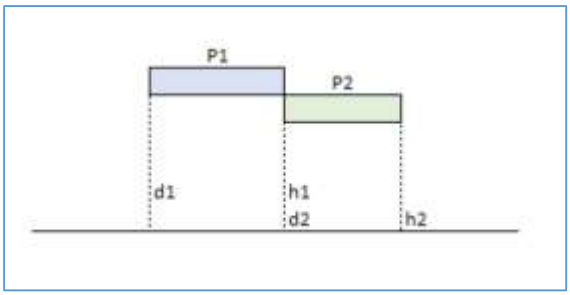
$$P_1(d_1, h_1) = P_2(d_2, h_2) \Leftrightarrow d_1 = d_2 \wedge h_1 = h_2$$




*Períodos iguales*

## Períodos contiguos

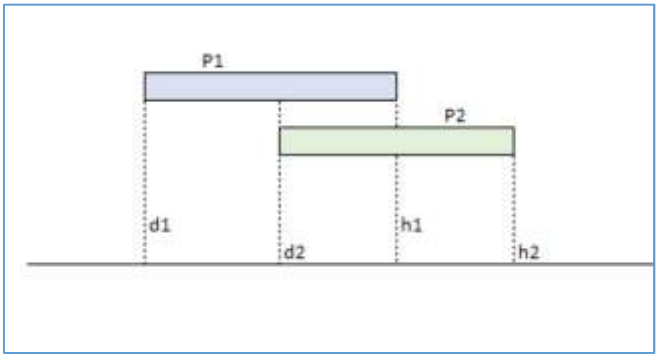
Un período $P_2(d_2, h_2)$ es **contiguo** a otro $P_1(d_1, h_1)$ si, y sólo si, se cumple $h_1 = d_2$.




*Períodos contiguos*

## Solapamiento de períodos

Dos períodos $P_1(d_1,\ h_1)$ y $P_2(d_2\ , h_2)$ **se solapan** si, y sólo si, se cumplen $(h_2 > d_1)$ y además $(d_2 < h_1)$



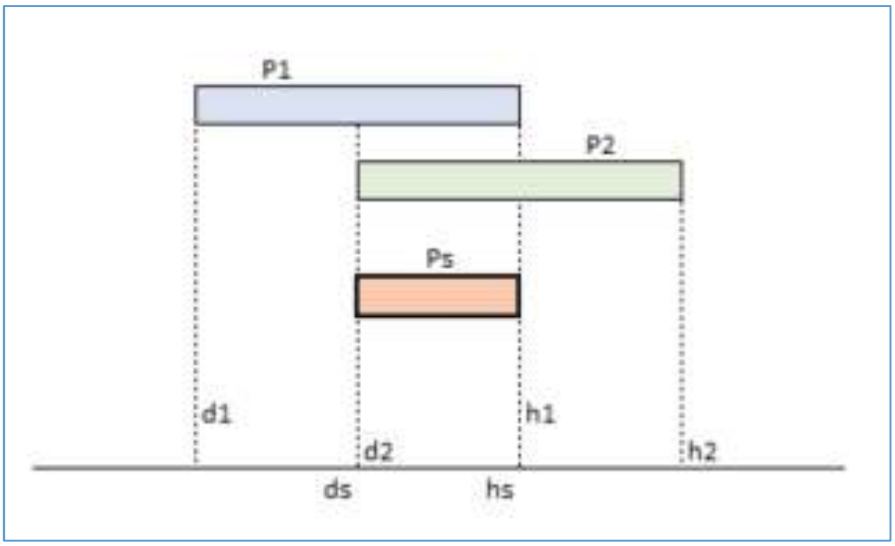

*Períodos solapados*

## Intersección de períodos

Si dos períodos $P_1(d_1,\ h_1)$ y $P_2(d_2\ , h_2)$ se solapan, se define el **período solapado o intersección** como el período $\boldsymbol{P_s(d_s,\ h_s)}$ con:

$$d_s = maximo(d_1, d_2)$$

$$h_s = minimo(h_1, h_2)$$



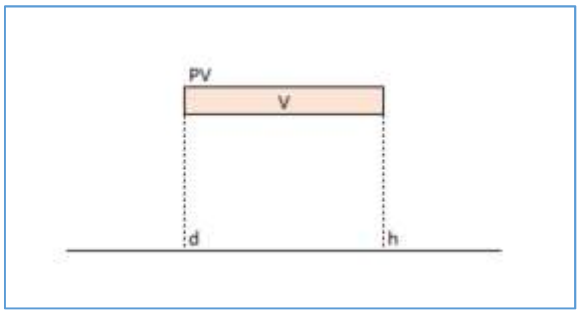

*Período solapado (Intersección de períodos)*

## Ordenación de períodos

Se considera que un período $P_1(d_1,\ h_1)$ es **anterior**, o **menor**, a otro $P_2(d_2\ , h_2)$ si "*comienza antes*", es decir, si, y sólo si, $d_1 < d_2$. Esta condición será la usada como criterio de ordenación a la hora de **ordenar listas** de períodos.

## Los períodos valor

Un período valor PV es un período con un valor asociado, es decir, un par (P, V), donde P es un período válido y V un valor cualquiera de un tipo determinado T.




*Período valor*

Los conceptos relativos a los períodos que se han descrito en el apartado anterior se pueden considerar aplicables de la misma forma a los períodos valor, dado que éstos sólo son períodos a los cuales se les asocia un valor, pero esto no interfiere en el comportamiento temporal.

Aunque en principio se pueda pensar en un valor como un tipo de dato simple (un número, una cadena, etc.), realmente el tipo T puede ser cualquiera, con la complejidad que sea necesaria; una clase con múltiples propiedades, una lista de objetos, un mapa de valores, etc.

## El valor NULL

Un período valor podrá tener un **valor especial *NULL*** que indique que en **ese período no hay valor**. En una implementación determinada se podrá usar el valor que resulte más adecuado, según la tecnología empleada ("*null*", "*nothing*", "*void*", etc.); es suficiente con que, en la implementación, este valor especial se pueda identificar como tal.

Se asume que el valor *NULL* pertenece a todos los tipos T, es decir, que cualquier valor de un tipo T puede ser nulo.

## Cálculo de valores: función *calcular()*

La solución planteada en este documento resuelve la gestión de los valores en la dimensión temporal, en el sentido de proporcionar una forma de tratar los valores de una manera uniforme en el tiempo. No obstante, y en función del caso que se trate, hay un aspecto específico de cada negocio, que es el que resuelve ***cómo calcular el valor resultante cuando hay un solapamiento de dos períodos***.

Si volvemos al ejemplo inicial, en el que se hablaba de valores de costes, puede ser que, cuando haya dos costes en un mismo período, el coste final sea la suma. Pero puede ocurrir que la fórmula sea más compleja, y haya que considerar valores medios, límites inferiores, límites máximos, o cualquier otro factor… incluso, en el caso de que los valores sean estructuras de datos más complejas, es posible que el resultado final del "valor" (que será un valor con la misma estructura) sea resultado de combinar atributos, listas, etc., de los dos valores. Todo lo complejo que se necesite, pero siempre acotado al planteamiento de "***calcular el resultado de dos valores***".

Tal y como se está planteando, es fácil deducir que, en términos de desarrollo, esto suele resolverse (en función de la plataforma y cómo se decida implementar) con una interface, una clase abstracta, o una función lambda.

En cualquier caso, en términos generales y con independencia de la plataforma de implementación, se definirá una operación $\boldsymbol{calcular}(\boldsymbol{v_1}, \boldsymbol{v_2})$ como una función que recibe dos argumentos $(v_1, v_2)$, ambos de tipo T y devuelve un valor de tipo T, cumpliendo además que, si uno de los argumentos $v_1$ ó $v_2$ es ***NULL***, entonces el resultado es el valor del otro argumento, es decir:

- $calcular(v_1, NULL) = v_1$
- $calcular(NULL, v_2) = v_2$
- $calcular(NULL, NULL) = NULL$

## Cálculo de períodos valor

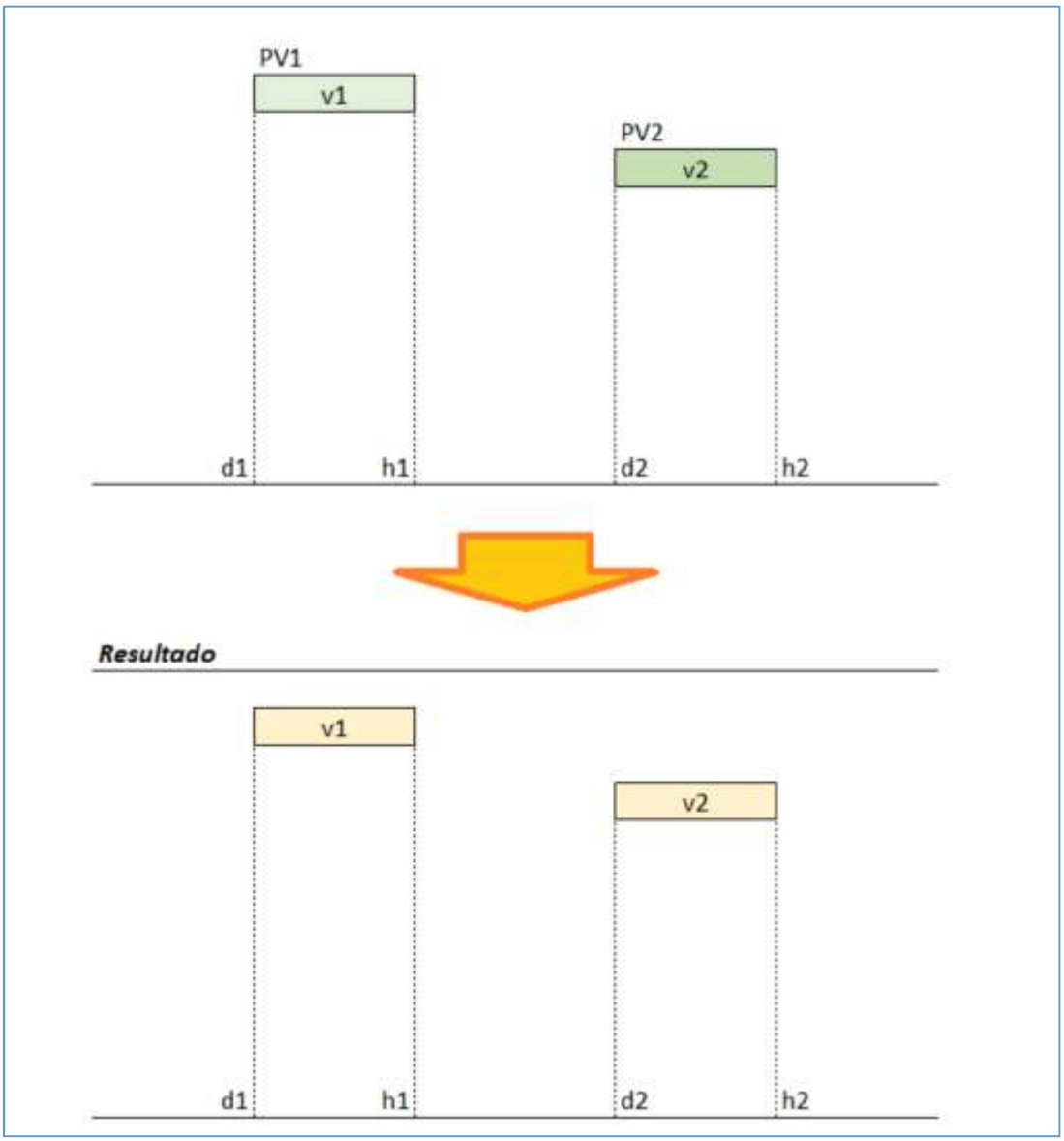
Una vez definida la función $\boldsymbol{calcular(v_1, v_2)}$ sobre los valores de tipo T, se puede extender el cálculo a **períodos valor**, de la siguiente forma.

El **resultado** de calcular dos períodos valor:

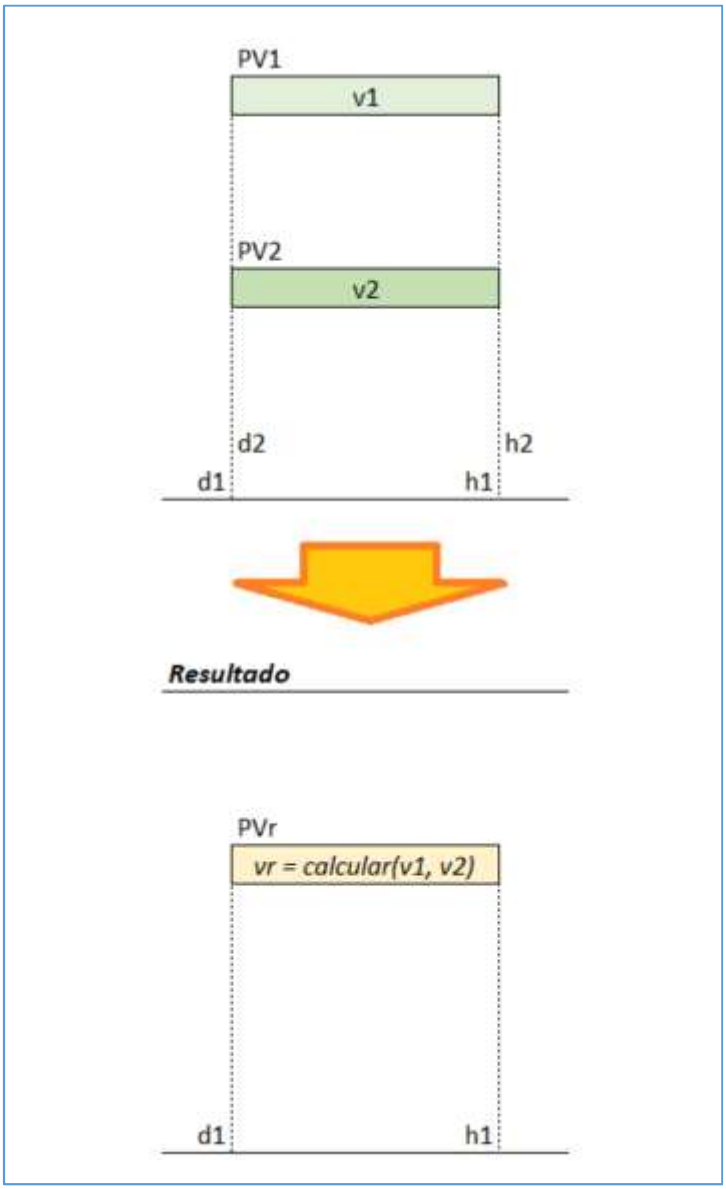
- $PV_1$ con el valor $v_1$ en el período $P_1(d_1, h_1)$
- $PV_2$ con el valor $v_2$ en el período $P_2(d_2, h_2)$

es el **conjunto** de períodos valor formado por:

1. En caso de que los dos períodos $P_1 y\, P_2$ sean **disjuntos** (no solapados), el resultado es el conjunto de los dos períodos valor originales, es decir $\{PV_1, PV_2\}$.



2. En caso de que los dos períodos $P_1 y\ P_2$ sean **iguales** $(P_1 = P_2)$ entonces el resultado es el conjunto formado por un único período valor $\{PV_r\}$, con el valor $v_r = calcular(v_1, v_2)$ en el período $P_1(d_1, h_1)$

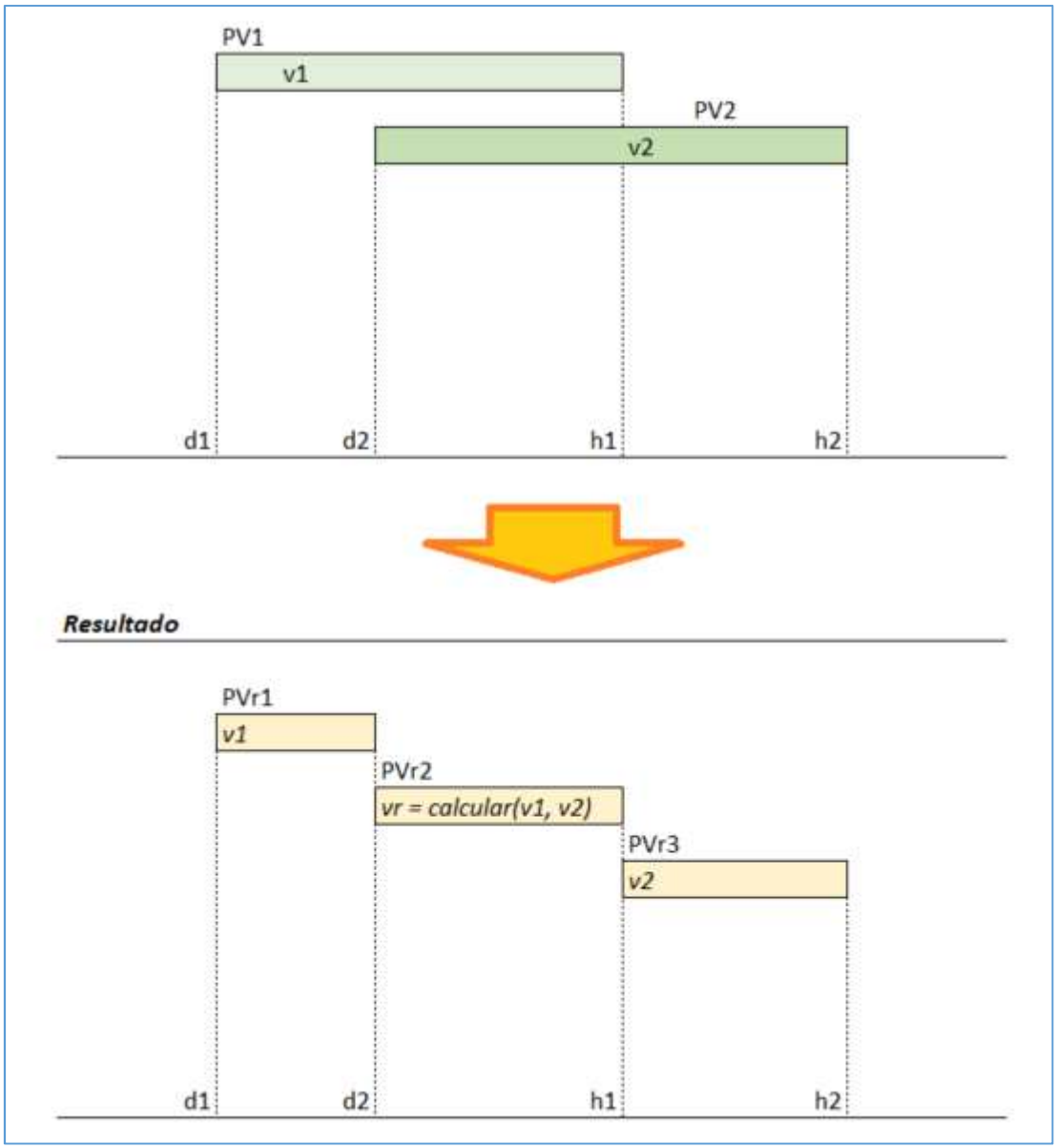



3. En el caso de que los dos períodos $P_1 y\ P_2$ se **solapen**, entonces existirá un período de intersección $P_i(d_i, h_i) = (d_2, h_1)$ y el resultado de calcular los dos períodos valor será el conjunto formado por hasta tres períodos valor $\{PV_{r1}, PV_{r2}, PV_{r3}\}$**,** en donde:

   - $PV_{r1}$ es el período valor para el período $(d_1, d_i)$ con el valor $v_1$, siempre y cuando $d_1 < d_i$ . En caso contrario, el resultado no contendría a este periodo valor.

   - $PV_{r2}$ es el período valor para el período $P_i(d_i, h_i)$, es decir, la intersección, con el valor $v_r = calcular(v_1, v_2)$

- $PV_{r3}$ es el período valor para el período $(h_i\ , h_2)$ con el valor $v_2$, siempre y cuando $h_i < h_2$. En caso contrario, el resultado no contendría este período valor.




## Listas de períodos valor equivalentes

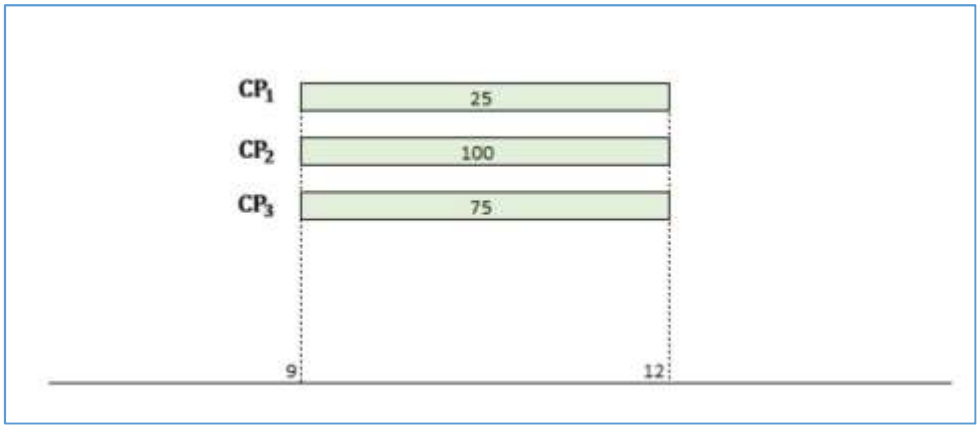
Una vez determinada la forma de calcular periodos valor, disponemos de un mecanismo mediante el cual podemos definir equivalencias entre conjuntos (o listas) de períodos valor.

Básicamente, una lista de períodos valor **L1 es equivalente a otra L2 cuando el resultado de calcular todos los períodos valor de L1 es igual al resultado de calcular todos los períodos valor de L2**.

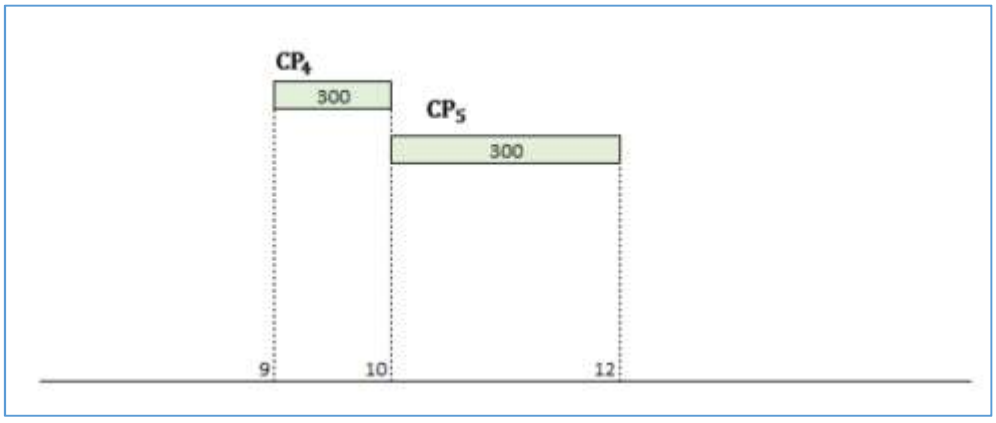
Realmente, este concepto de equivalencia ya se mostró en el primer apartado de este documento, cuando se expuso el problema de la periodificación de valores. Volviendo al mismo

ejemplo, si consideramos la función calcular() que simplemente suma los valores, podemos afirmar que las dos listas:

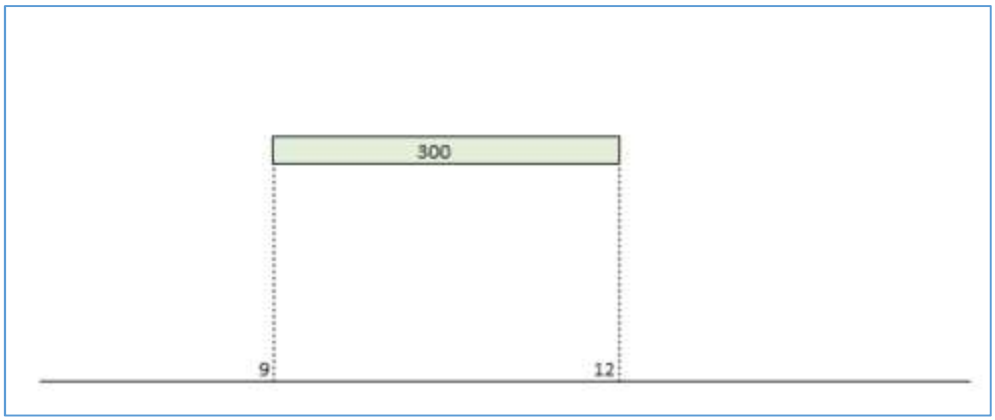
$L_1 = \{ CP_1 , CP_2 , CP_3 \}$




y $L_2 = \{ CP_4 , CP_5 \}$




Son equivalente, dado que resultado de calcular los períodos $CP_1 , CP_2 , CP_3$ y el resultado de calcular $CP_4 , CP_5$ son el mismo:




Llegados a este punto, tenemos los conceptos necesarios para poder definir una estructura que permita gestionar de forma unificada conjuntos de períodos valor.

## Listas normalizadas de períodos valor

El problema de la periodificación de valores, tal y como se ha planteado, reside en el hecho de que un mismo período valor se puede expresar de múltiples formas, lo que dificulta su tratamiento.

El objetivo por tanto será definir una estructura en la que, por construcción, los períodos valor sólo puedan ser expresados de una única forma.

### Definición de lista normalizada de períodos valor

Dada una lista de períodos valor ***L***, cuyos valores son de tipo ***T***, y hay definida una función $\boldsymbol{calcular}(\boldsymbol{v_1}, \boldsymbol{v_2})$ para los valores, se define la **lista de períodos valor $\boldsymbol{L_n}$ equivalente en un intervalo *(A, B)*** como una lista de períodos valor equivalente que cumple:

1) El primer período valor de la lista tiene inicio en el instante A.

2) El último período valor de la lista termina en B.

3) No existen solapamientos entre ningunos de los períodos valor.

4) Todos los períodos son contiguos; no hay huecos entre ninguno de los períodos valor (los huecos existentes se sustituyen por períodos valor con el valor *NULL*).

5) No existen dos períodos contiguos con el mismo valor; en caso de tener el mismo valor, los dos períodos se unifican.

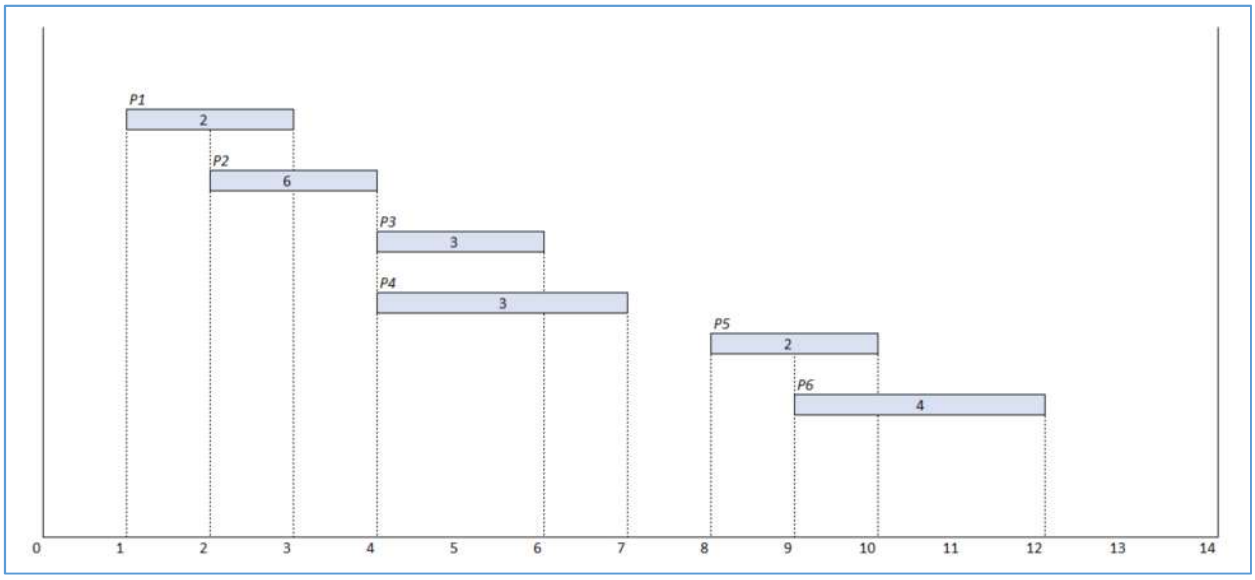
Supongamos el ejemplo inicial, en el que los valores se expresan en euros, y que la función `calcular()` que determina el valor de los costes en períodos solapados es simplemente la función suma; es decir (usando notación javascript):

```
calcular = function(v1, v2) {
   if (v1 == nothing){
       return v2;
   }
   if (v2 == nothing){
       return v1;
   }

   return (v1 + v2);
}
```

En este caso, supongamos que tenemos la lista de períodos valor siguiente (vamos a suponer, sólo para simplificar la notación, que los valores desde y hasta de los períodos se expresan sólo en términos horarios, es decir, que son valores de horas dentro de un mismo día):

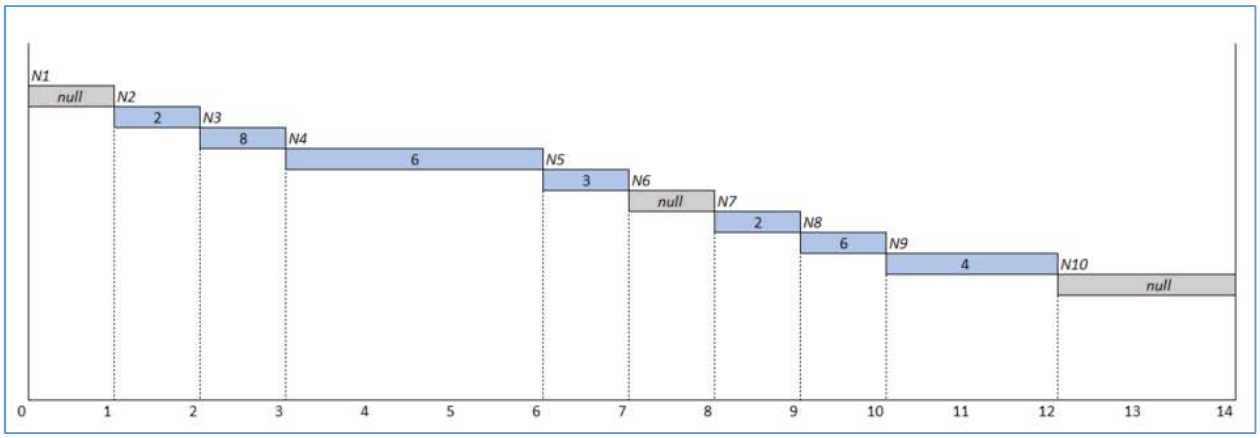
$$L = \{ P_1 , P_2 , P_3, P_4 , P_5 , P_6 \}$$




*Lista de períodos valor original*

la lista de períodos valor equivalente para el intervalo entre 0 y 14 es la siguiente:

$$L_n = \{ N_1 , N_2 , N_3, N_4 , N_5 , N_6, N_7, N_8, N_9, N_{10}, N_{11} \}$$




*Lista de períodos valor normalizada equivalente*

En este ejemplo podemos comprobar los cinco puntos que definen la lista normalizada:

1) El primer período valor de la lista tiene inicio en el instante A, es decir, cero. Se puede ver que, aunque en la lista original no hay período valor que comience en cero, se considera un período con valor *NULL* y comienzo en ese instante (período $\mathbf{N_1}$ en la imagen).

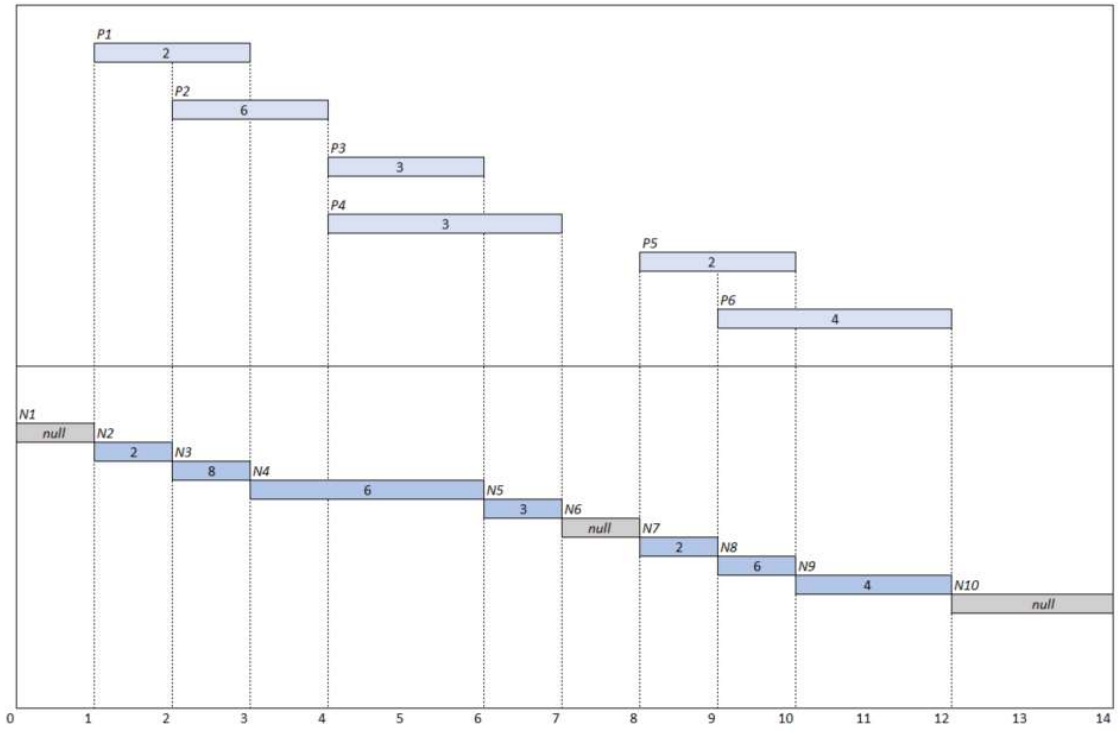
2) El último período valor de la lista termina en B, es decir, 14. Dado que originalmente no hay período valor que termine en 14, se usa un período valor con valor *NULL* y finalización en ese instante (período $\mathbf{N_{10}}$).

3) No existen solapamientos entre ningunos de los períodos valor; aunque en la lista original los períodos $\mathbf{P_3}$, $\mathbf{P_4}$ y los períodos $\mathbf{P_5}$, $\mathbf{P_6}$ sí se solapan, en la normalizada no, pues se usa el resultado de calcular los período valor solapados.

4) Todos los períodos son contiguos; no hay huecos entre ninguno de los períodos valor. En el ejemplo se puede comprobar que, dentro del intervalo (0, 14) no hay ningún hueco. Los períodos en los que no hay valor, se cubren con períodos con valor *NULL* (los períodos en color gris: $\mathbf{N_1}$, $\mathbf{N_6}$, $\mathbf{N_{10}}$

5) No existen dos períodos contiguos con el mismo valor; en caso de tener el mismo valor, los dos períodos se unifican. Si nos fijamos en el ejemplo, el período valor $\mathbf{N_4}$ (con valor seis) resulta de unificar dos períodos con el mismo valor.

En la siguiente imagen, se puede ver las dos listas (original y normalizada) para comparar:




En el siguiente apartado se describe un algoritmo para la construcción de una lista de períodos valor normalizada a partir de una lista de períodos valor cualquiera.

## Algoritmo de construcción de una lista normalizada de períodos valor

Cualquier implementación que cumpla los cinco puntos especificados en la definición anterior es válida para proporcionar una solución sólida a la periodificación de valores.

Además, partiendo de los conceptos definidos en este documento sobre períodos y valores, es fácil intuir o apoyarse e ellos para orientar la forma de programar una estructura como ésta; conceptos como la intersección, el solapamiento, la pertenencia de un instante a un período, etc. son casi guías directas para definir estructuras de datos o clases y funciones o métodos que sirvan de apoyo para solucionar el problema.

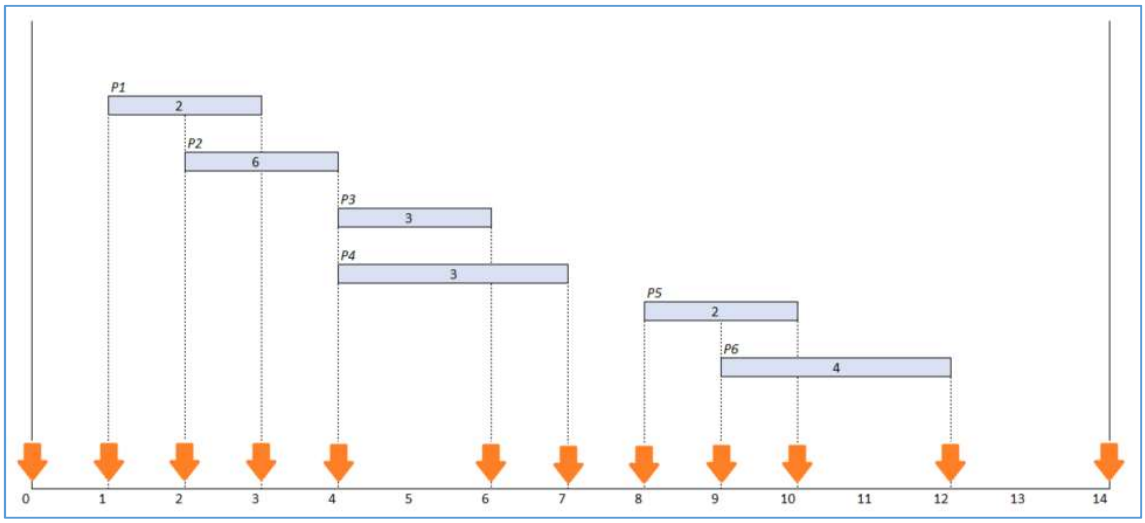
En este apartado se mostrará, mediante un ejemplo, el que seguramente sea el método más simple para construir la lista normalizada en un intervalo a partir de una lista de períodos valor cualquiera.

Este método de construcción se basa en determinar los "***puntos de corte***", o instantes que son inicio o fin de los períodos a tratar, incluyendo el inicio y fin del intervalo en el que se define la lista normalizada.

Para el caso de del ejemplo anterior, para la lista:

$$L = \{ P_1 , P_2 , P_3, P_4 , P_5 , P_6 \}$$

en el intervalo **[0, 14]**, presenta los siguientes "*puntos de corte*" (marcados con las flechas naranjas):




***Puntos de corte***

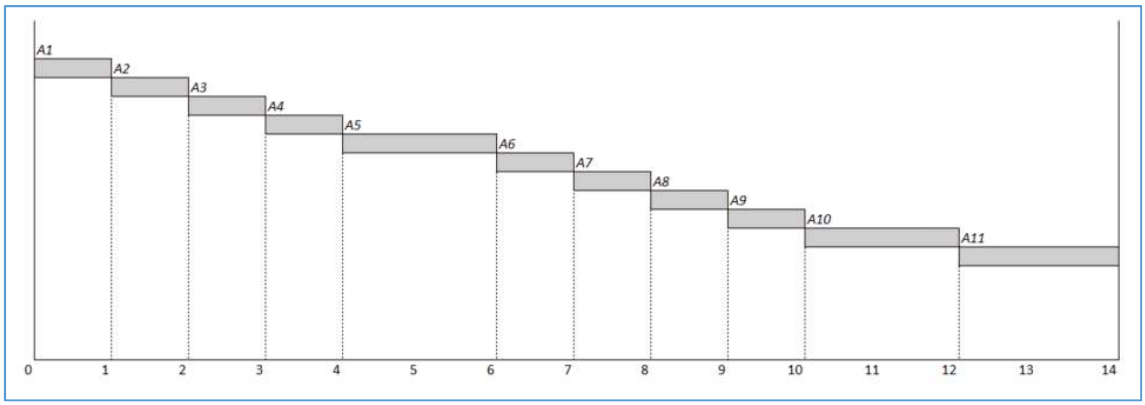
Es decir:

***PC = {0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 12, 14}***

Si alguno de los puntos aparece más de una vez (por ejemplo, los *P3* y *P4* comienzan los dos en el instante 4), se considera sólo una vez.

La clave de este método radica en que este conjunto de "*puntos de corte*", considerándola como una lista ordenada y sin duplicados, determina los períodos que van a formar parte de la lista normalizada, con una única excepción: si al final resultan períodos contiguos con el mismo valor, se unifican. Pero, en cualquier caso, en base a esto, esta lista de períodos inicial es una base sólida para construir la lista normalizada final.

Siguiendo lo comentado, la lista de períodos (sin valor) candidatos a partir de los puntos de corte sería ésta (sólo habría que recorrer la lista en un bucle e ir construyendo los períodos):




***Lista de períodos candidatos***

Una vez tenemos la lista de períodos candidatos, tenemos que determinar el valor para cada uno de ellos, mediante la función *calcular*().

Por construcción, cada período de la lista de períodos candidatos es "atómico", es decir, que no es necesario desglosar en períodos más pequeños, dado que, de la lista original, no hay períodos solapados más pequeños. Esto permite simplificar la forma de calcular el valor de cada período candidato de la siguiente forma:

1) Se considera el instante de inicio del período ***d***.

2) Se consideran los valores de todos los períodos de la lista inicial ***L*** que contienen el instante ***d***. Si no hay ningún período, se considera el valor *null*.

3) Se calcula el resultado de todos los valores obtenidos en el punto anterior, y el resultado se asigna como valor al período candidato.

Esto, que dicho así puede resultar un poco lioso, es muy sencillo; con el ejemplo actual, veamos cómo se asignan los valores a cada uno de los períodos candidatos:

- Período candidato *A1*(0, 1) ➔ Consideramos el valor del instante inicial d = 0. En la lista inicial L, se buscan todos los períodos que contengan el instante 0. No hay ninguno, por lo que el período *A1* toma el valor *NULL*.

- Período candidato *A2*(1, 2) ➔Consideramos d = 1. De la lista *L*, sólo hay un período que contenga el instante 1. Es el período *P1*, que tiene el valor 2 ➔ A2 toma el valor 2.

- Período candidato *A3*(2, 3) ➔ Consideramos d = 2; de la lista *L* hay dos períodos que contienen el instante 2; son los períodos *P1* y *P2*, con valores 2 y 6 respectivamente ➔ *A3* toma el valor calcular(2, 6) = 8.

- Período candidato *A4*(3, 4) ➔ Consideramos d = 3; de la lista *L* hay un período que contiene el instante 3; es el períodos *P2*, con valor 6 ➔ *A4* toma el valor 6.

- Período candidato *A5*(4, 6) ➔ Consideramos d = 4; de la lista *L* hay dos períodos que contienen el instante 4; son los períodos *P3* y *P4*, con valores 3 y 3 respectivamente ➔ *A5* toma el valor calcular(3, 3) = 6.

- Período candidato *A6*(6, 7) ➔ Consideramos d = 6; de la lista *L* hay un período que contiene el instante 6; es el períodos *P4*, con valor 3 ➔ *A6* toma el valor 3.

- Período candidato *A7*(7, 8) ➔ Consideramos d = 7; de la lista *L* no hay ningún período que contenga el instante 7 ➔ *A7* toma el valor *NULL*.

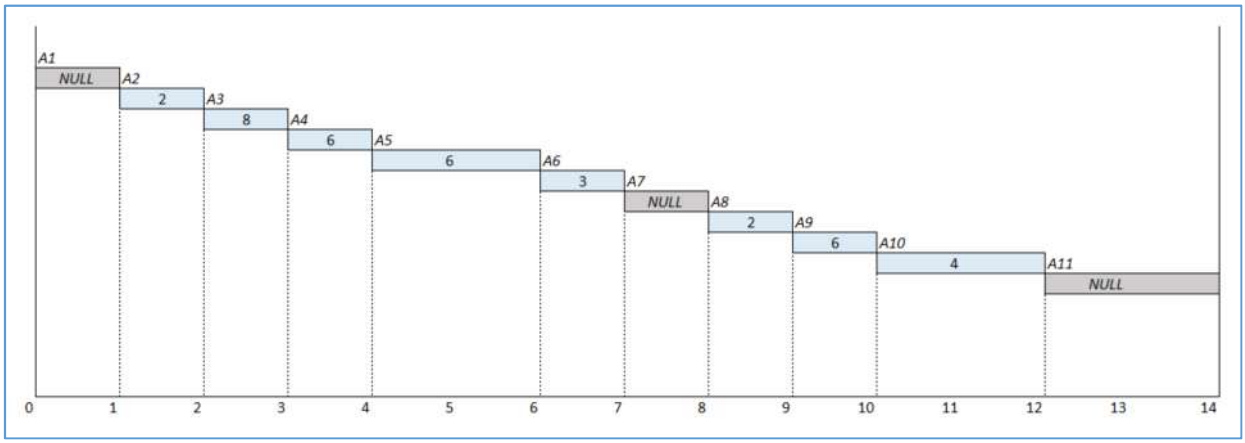
- Período candidato *A8*(8, 9) ➔ Consideramos d = 8; de la lista *L* hay un período que contiene el instante 8; es el períodos *P5*, con valor 2 ➔ *A8* toma el valor 2.

- Período candidato *A9*(9, 10) ➔ Consideramos d = 9; de la lista *L* hay dos períodos que contienen el instante 9; son los períodos *P5* y *P6*, con valores 2 y 4 respectivamente ➔ *A9* toma el valor calcular(2, 4) = 6.

- Período candidato *A10*(10, 12) ➔ Consideramos d = 10; de la lista *L* hay un período que contiene el instante 10; es el períodos *P6*, con valor 4 ➔ *A10* toma el valor 4.

- Período candidato *A11*(12, 14) ➔ Consideramos d = 12; de la lista *L* no hay ningún período que contenga el instante 12 ➔ *A11* toma el valor *NULL*.

Por tanto, aplicando los valores obtenidos para cada período candidato, tenemos la siguiente lista de *períodos valor candidatos*:



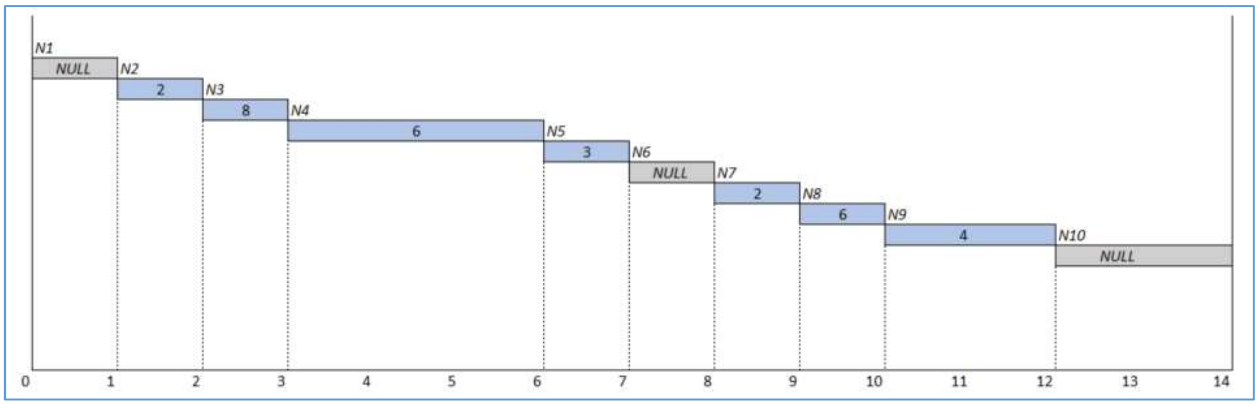

***Lista de períodos valor candidatos***

En este punto, sólo queda un último paso; la lista de períodos valor candidata cumple cuatro de los cinco puntos que definen una lista de períodos valor normalizada; el primero período empieza en el comienzo del intervalo, el último finaliza con el intervalo, no hay solapamientos y todos los períodos son contiguos, pero falta **garantizar que no haya dos períodos contiguos con el mismo valor**. De hecho, en el ejemplo, los períodos valor *A4* y *A5* son contiguos, y sus valores son iguales (6).

Por tanto, en el último paso, se recorrerán los períodos valor de la lista, unificando aquellos que, siendo contiguos, tienen el mismo valor.

En este caso, los períodos *A4*(3, 4) y *A5*(4, 6) se unifican en un único período (3, 6) con valor 6.

La lista normalizada queda por tanto de la siguiente forma:




***Lista de períodos valor normalizada***

El algoritmo para la construcción de una lista normalizada de períodos valor equivalente en un intervalo, para de lista de períodos valor se puede resumir por tanto en cuatro pasos, tal y como se ha descrito:

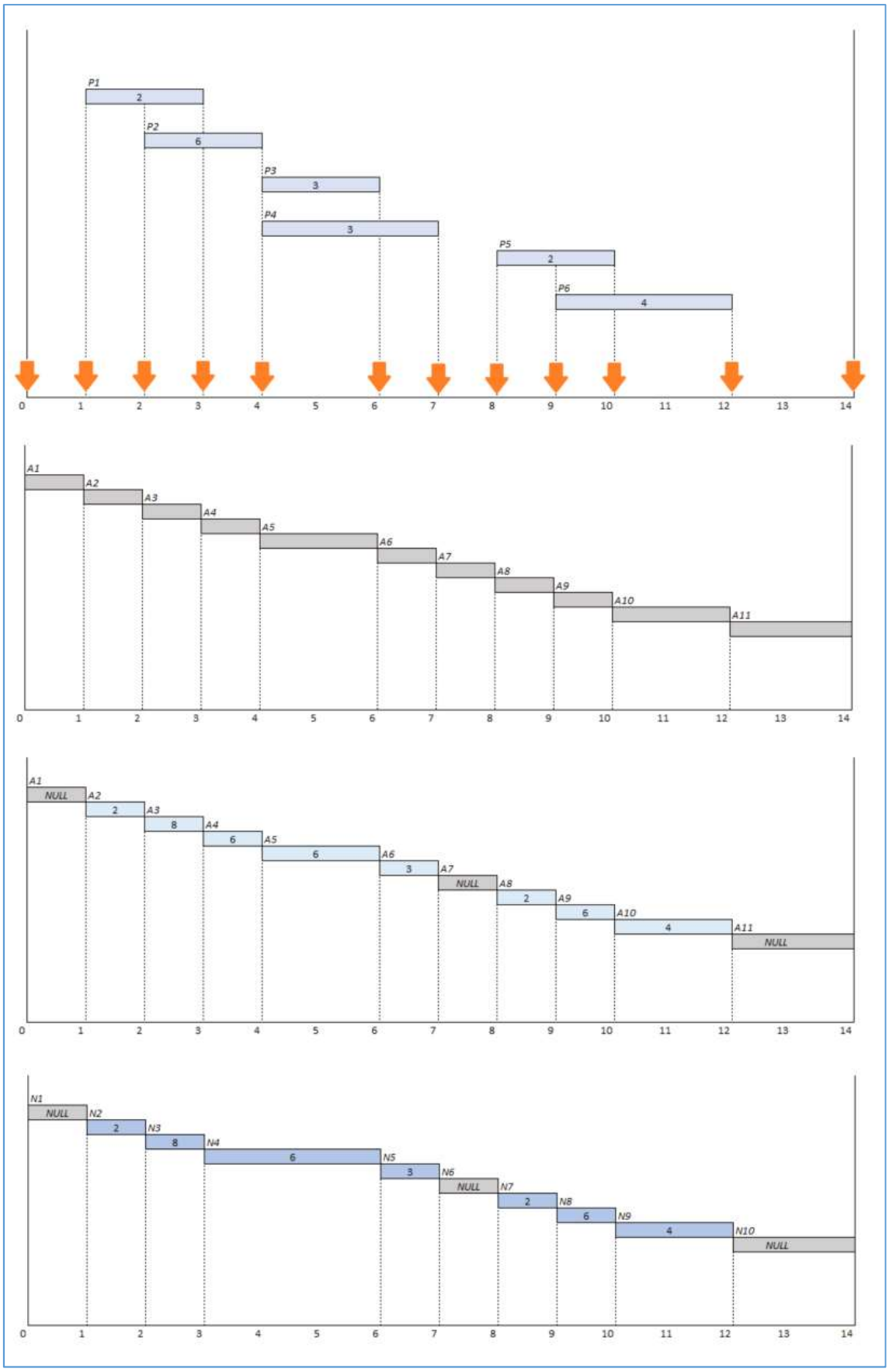
- **Paso 1**: Determinar la lista de ***puntos de corte***, en función del intervalo en el que se quiera definir la lista normalizada, y los períodos de la lista original,

- **Paso 2:** Crear, a partir de la lista de puntos de corte obtenida en el paso 1, una lista con los ***períodos candidatos***, de forma secuencial.

- **Paso 3**: Asignar el valor de cada uno de los períodos del paso 2, teniendo en cuenta para ello los valores de los períodos de la lista original que contienen el inicio de cada período, obteniendo con ello la lista de ***períodos valor candidatos***.

- **Paso 4**: En la lista del paso anterior, ***unificar los períodos contiguos con el mismo valor***.

La siguiente imagen reúne los cuatro pasos para el ejemplo anterior:


P1
2
P2
6
P3
3
P4
3
P5
2
P6
4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
A1
NULL
A2
2
A3
8
A4
6
A5
6
A6
3
A7
NULL
A8
2
A9
6
A10
4
A11
NULL
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
N1
NULL
N2
2
N3
8
N4
6
N5
3
N6
NULL
N7
2
N8
6
N9
4
N10
NULL
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

## Conclusión

Con una estructura como ésta, podemos tratar de manera uniforme valores periodificados sin problemas, dado que la forma de gestionarlos, tanto en lo referente al cálculo como al aspecto temporal está totalmente definida, y no da lugar a error.

Siguiendo estas pautas, no sólo se simplifica cualquier tipo de operación, validación, extracción, registro, etc. con los valores en el tiempo a nivel individual, sino que se podrían ejecutar artefactos distintos de un mismo sistema, por ejemplo un front-end con JavaScript y un back-end con java, teniendo la seguridad de que los datos se tratan de la misma manera; del mismo modo que una lista de libros de una biblioteca cualquiera se espera que sea la misma en los dos lados, con el mismo número de elementos y los mismos valores, una lista de períodos valor también lo puede ser si usamos listas normalizadas.

Evidentemente, en un caso real, lo normal es que la cosa no termine aquí; las listas de períodos valor normalizadas serán normalmente una base sobre la que se pueda trabajar y definir funcionalidades nuevas, seguramente con nuevas estructuras más complejas; estas listas normalizadas son sólo una pieza del puzle, pero una pieza sólida, que sin duda evitará muchos quebraderos de cabeza a la hora de ampliar los casos de uso del sistema.