

# Tema 4

El Nivel Interno



# Apartado 3

Métodos de organización y acceso a los datos: Acceso Directo

# Contenidos

- Acceso directo
- Hashing básico
- Hashing dinámico

# Contenidos

- Acceso directo
- Hashing básico
- Hashing dinámico

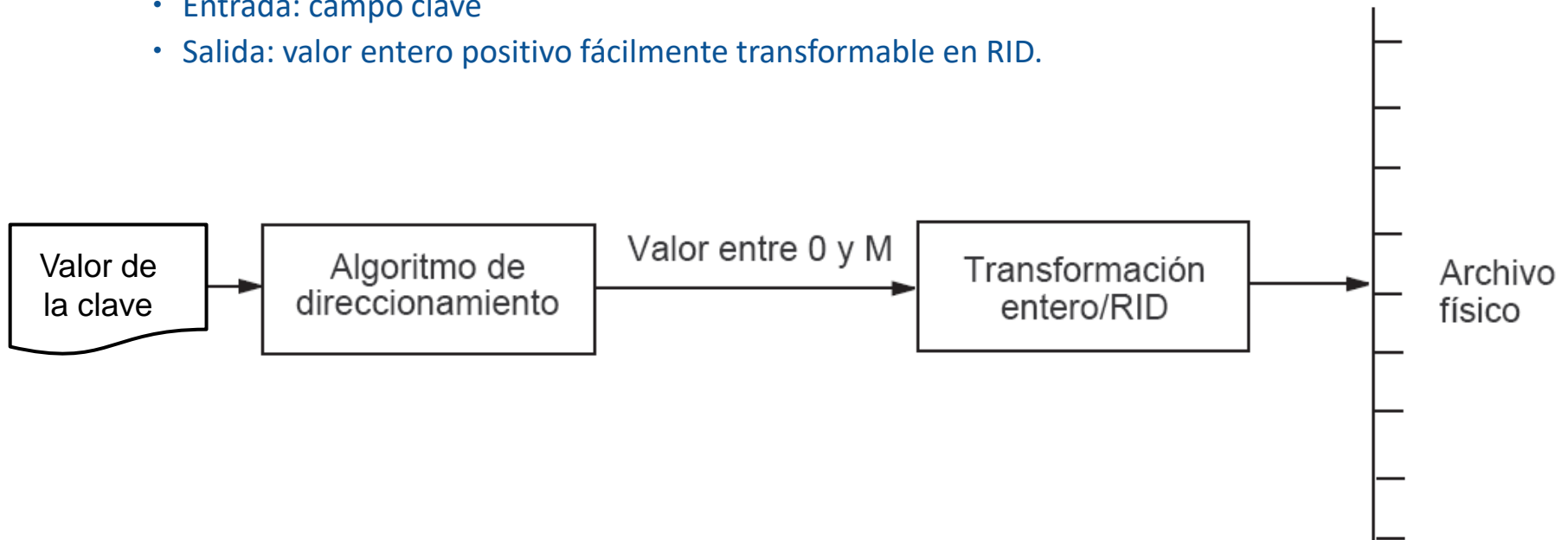
# Acceso directo

- Otra forma de acceder a un registro almacenado:
  - No hay una estructura adicional.
  - Se usa un algoritmo que nos indique directamente la posición del registro deseado.
- Acceso directo:
  - Calcular directamente la dirección de un registro mediante la aplicación de algún algoritmo o función sobre un campo determinado del mismo.

# Acceso directo

- Funcionamiento:

- Normalmente no es posible establecer una clave que sea totalmente correlativa y única para cada registro.
- Hay que buscar un algoritmo que transforme los valores de un cierto campo en una dirección.
  - Entrada: campo clave
  - Salida: valor entero positivo fácilmente transformable en RID.



# Acceso directo

- Los algoritmos de direccionamiento no suelen mantener el orden de la clave.
  - Los registros no están almacenados según el orden de la clave.
  - Problemas con la recuperación por intervalos.

# Acceso directo

- Hay una gran variedad de algoritmos:
  - Dependen del tipo de clave:
    - Si la clave es alfanumérica, hay que transformarla a un valor numérico.
  - Suelen estar basados en un mecanismo de generación de números pseudoaleatorios:
    - *Cuadrados centrales:*
      - Se eleva la clave al cuadrado y se eligen tantos dígitos centrales como sea necesario.
    - *Congruencias:*
      - Se divide la clave por M y se toma el resto (M suele ser primo).
    - *Desplazamiento:*
      - Se superponen adecuadamente los dígitos binarios de la clave y luego se suman.
    - *Conversión de base:*
      - Se cambia la base de numeración y se suprimen algunos dígitos resultantes.



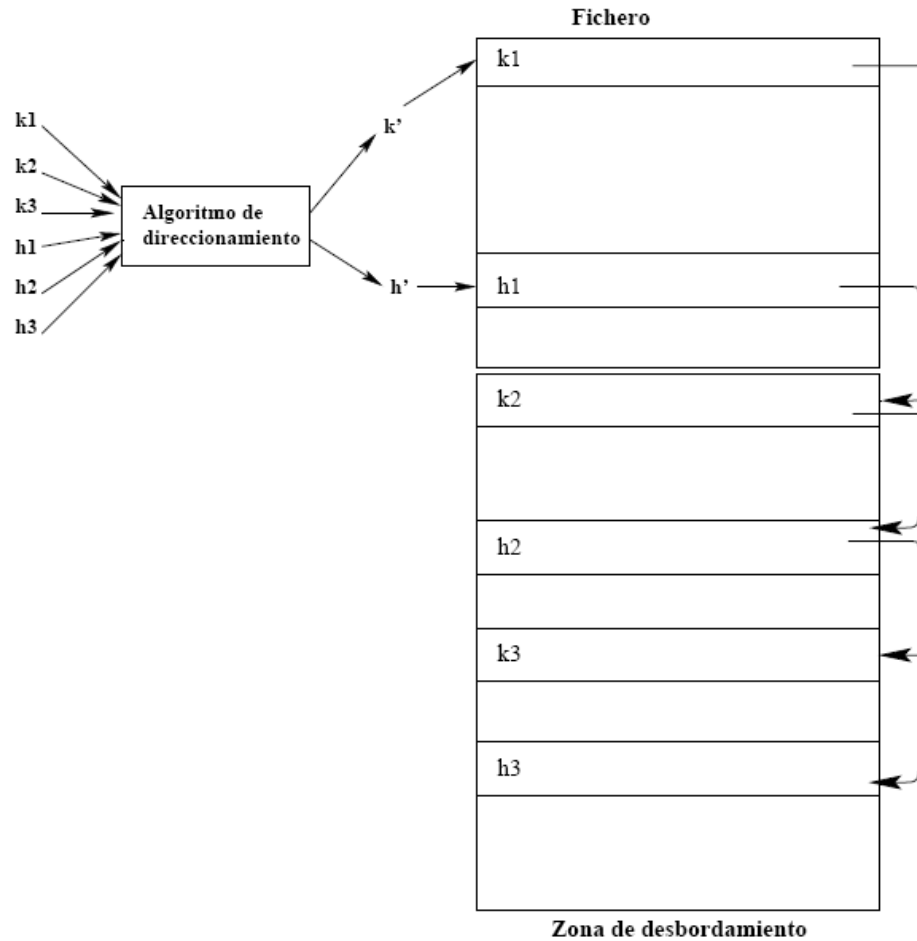
# Acceso directo

- Problemas
  - Salvo que el campo clave se diseñe para ello, es prácticamente imposible encontrar una transformación que dé un valor entero positivo en un rango de valores limitado tal que no haya dos valores distintos de clave que den lugar al mismo número.
    - Por lo que se producen colisiones.
  - Los algoritmos también producen huecos:
    - Zonas vacías del rango de salida, no asignadas por el algoritmo.
    - Se traducen en huecos en el fichero de datos.

# Acceso directo

- Para gestionar colisiones y huecos:
  - Combinar el acceso directo con una gestión mediante listas de colisión.
    - Zona de desbordamiento.
    - Al producirse una colisión:
      - El registro problemático se almacena en la zona de desbordamiento.
      - Los sinónimos (registros con claves que producen colisión) se conectan mediante una lista.

# Acceso directo



# Acceso directo

- Si crecen las listas de sinónimos:
  - El acceso directo puro no resulta adecuado.
    - Mantener listas.
    - Zona de desbordamiento casi como el fichero original.
- Han aparecido técnicas más sofisticadas:
  - Hashing

# Contenidos

- Acceso directo
- Hashing básico
- Hashing dinámico

# Hashing básico

- Si el problema principal es que los valores de las claves no están uniformemente distribuidos en el intervalo  $[0, M]$ :
  - Se acumulan en una parte de este intervalo.
  - Solución:
    - Asignar más espacio a esa parte del intervalo.
- Técnica:
  - Se divide el espacio del fichero en “cubos” (buckets).
  - El algoritmo de direccionamiento asigna cubos, no direcciones concretas.
  - En cada “cubo” puede haber más de un registro.
  - Ciertos rangos de valores tienen asignados más cubos que otros.
  - Se complementa con el uso de “cubos de desbordamiento”.

# Hashing básico

- Parámetros:
  - Número de cubos.
  - Tamaño de los cubos (relación con bloques físicos).
  - La transformación clave/dirección, que debe tener en cuenta la distribución de la clave según rangos para evitar que unos cubos no se llenen mucho y otros se queden muy vacíos.

# Hashing básico

- Para insertar un registro:
  - Transformar la clave.
  - Localizar el cubo correspondiente.
  - Si hay sitio se inserta el registro y hemos terminado.
  - Si no hay sitio, se sitúa el registro en un cubo de desbordamiento conectándolo con el cubo que realmente le corresponde mediante punteros.



# Hashing básico

- El proceso de búsqueda:
  - Transformar la clave.
  - Localizar el cubo correspondiente.
  - Realizar una búsqueda secuencial dentro del cubo.
    - Si hemos encontrado el registro, el proceso termina.
    - En caso contrario, se impone “un barrido por punteros” a través de los cubos de desbordamiento.

# Contenidos

- Acceso directo
- Hashing básico
- Hashing dinámico

# Hashing dinámico

- El hashing básico sigue teniendo problemas:
  - Es necesario conocer la distribución previa de las claves para asignar adecuadamente los cubos.
  - En otro caso siguen apareciendo huecos/colisiones.
  - Al aumentar el número de registros, aumentan los registros en páginas de desbordamiento.
  - Se hacen necesarias las reorganizaciones.

# Hashing dinámico

- Solución:
  - Trabajar de forma dinámica.
- Se parte de una configuración uniforme y de pocos cubos.
- Los restantes se van generando conforme se necesiten.
  - Se asignan a los rangos conforme la afluencia de registros lo demanda.
  - Hashing dinámico o extensible

# Hashing dinámico

- Técnica:

- El valor transformado del campo clave nos lleva a la entrada de una tabla índice que se almacena en memoria.
- Allí está la dirección del cubo donde se encuentran los registros que tienen asociado este valor transformado.
- Puede ocurrir que varias entradas de la tabla conduzcan al mismo cubo.

- Proceso:

- Se toma un conjunto inicial de cubos, direccionados por la tabla índice. Por ejemplo, inicialmente, todas las entradas pueden apuntar al mismo cubo.
- A medida que vamos insertando registros, se van generando nuevos cubos y cambiando las salidas de la tabla índice.

# Hashing dinámico

- Algoritmo de Hashing Dinámico

- Datos de partida:

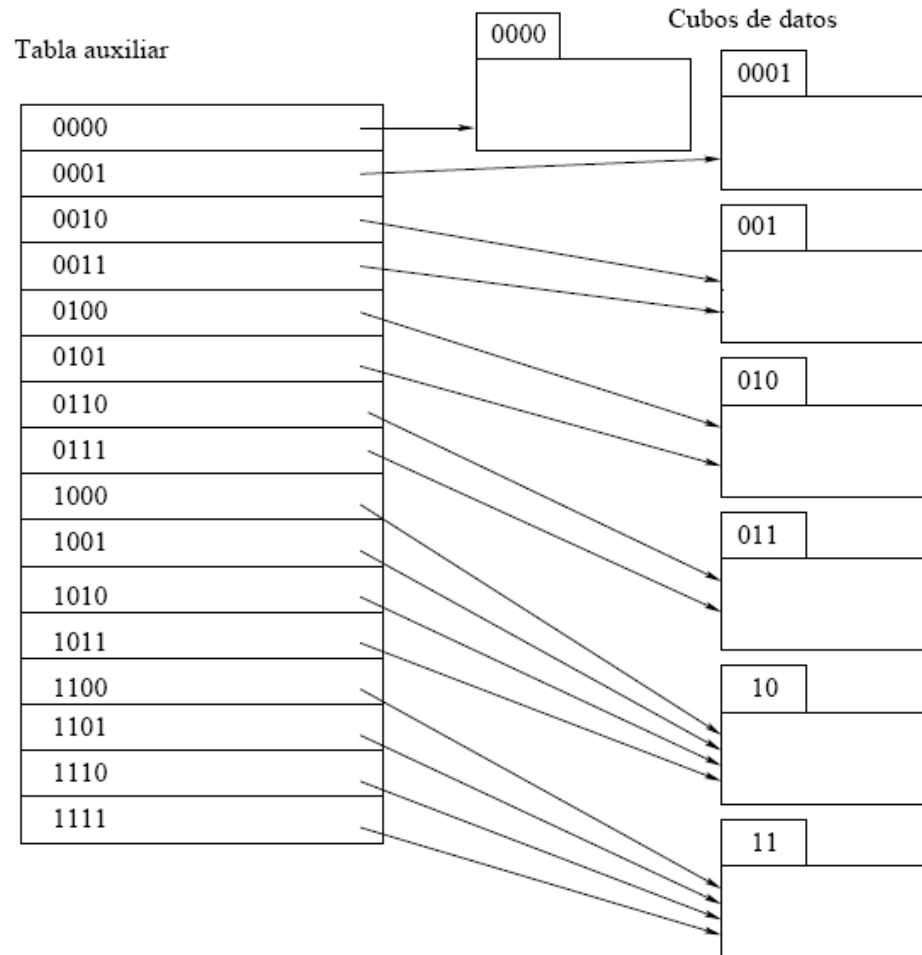
- $k$  = clave para direccionar
- $k' = h(k)$  valor entero entre 0 y  $M$
- $n$  = número de bits que tiene  $k'$  en binario
- $d \leq n$ , los  $d$  primeros (o últimos) dígitos de  $k'$  seleccionan el cubo donde está el registro y se llaman pseudollave.
- $b < d \leq n$ , inicialmente el archivo tiene  $2^b$  cubos distintos, como máximo tendrá  $2^d$ . Si son necesarios más, hay que aumentar  $d$  (si se puede).

# Hashing dinámico

- Algoritmo

- Se considera una tabla índice en memoria con  $2^d$  filas.
- En la primera columna de esta tabla se sitúan todas las posibles sucesiones de  $d$  dígitos binarios :
  - $d$  es la “profundidad global” de la tabla
- Todos los cubos tienen en principio “profundidad local” igual a  $b$ :
  - En principio, todas las entradas cuyos  $b$  primeros\* dígitos son iguales apuntan al mismo cubo.
    - Allí se almacenan los registros cuyo valor de  $k'$  tiene esos  $b$  primeros dígitos.
- Cuando se llena un cubo se divide en 2, poniendo en uno de ellos los registros con el dígito  $b+1$  de  $k'$  a 0 y en otro los que lo tienen igual a 1.
  - Este proceso aumenta la profundidad local de los cubos generados, que está limitada por la profundidad global de la tabla. Si se alcanza el límite, antes de dividir habría que aumentar la profundidad global (si es posible).

# Hashing dinámico



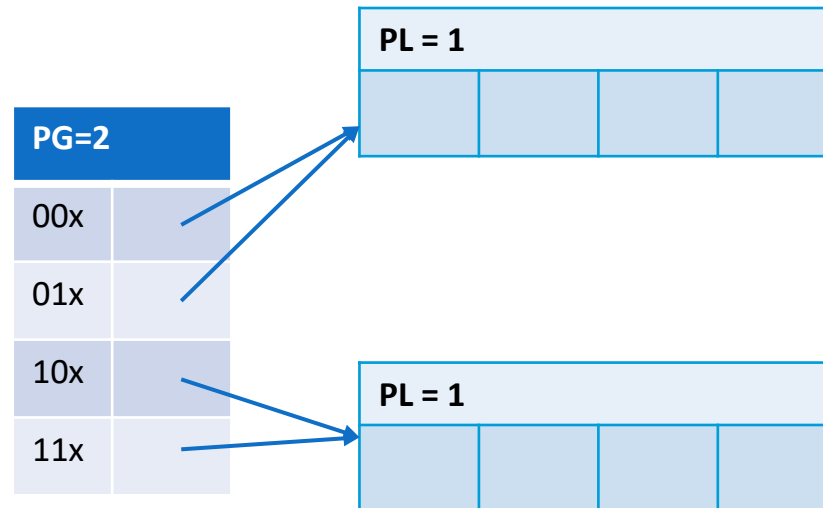


# Hashing dinámico

- Ejemplo

- Inicialmente:

- Tabla auxiliar de 2 bits de profundidad global (PG=d).
    - $b=1$ ; Por tanto 2 cubos con profundidad local (PL) 1.
    - 4 registros por cubo.
    - $h(k)=k \bmod 8$
    - Tomamos los d primeros bits para direccionar.

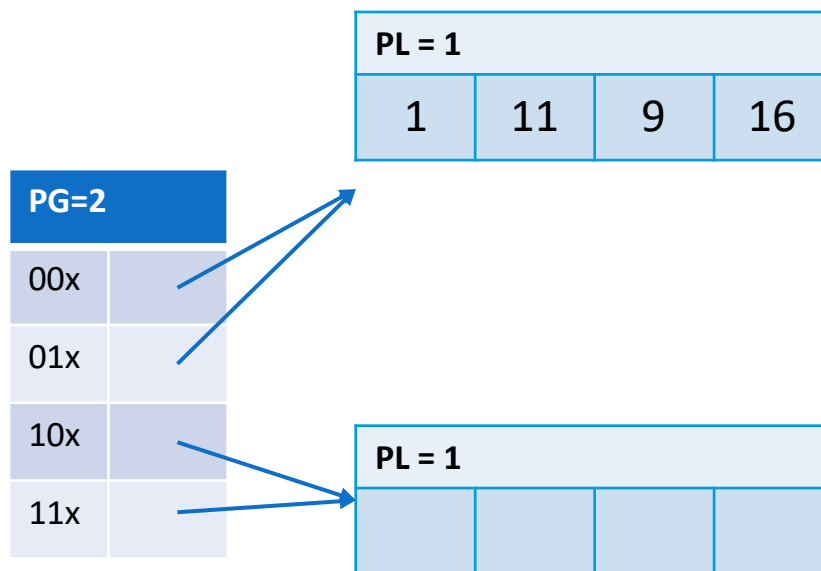


# Hashing dinámico

- Insertamos: 1, 11, 9, 16, 3, 32, 15, 4, 2, 19, 7, 14, 17, 27



	clave	H(k)	Binario
➡	1	1	001
➡	11	3	011
➡	9	1	001
➡	16	0	000
	3	3	011
	32	0	000
	15	7	111
	4	4	100
	2	2	010
	19	3	011
	7	7	111
	14	6	110
	17	1	001
	27	3	011

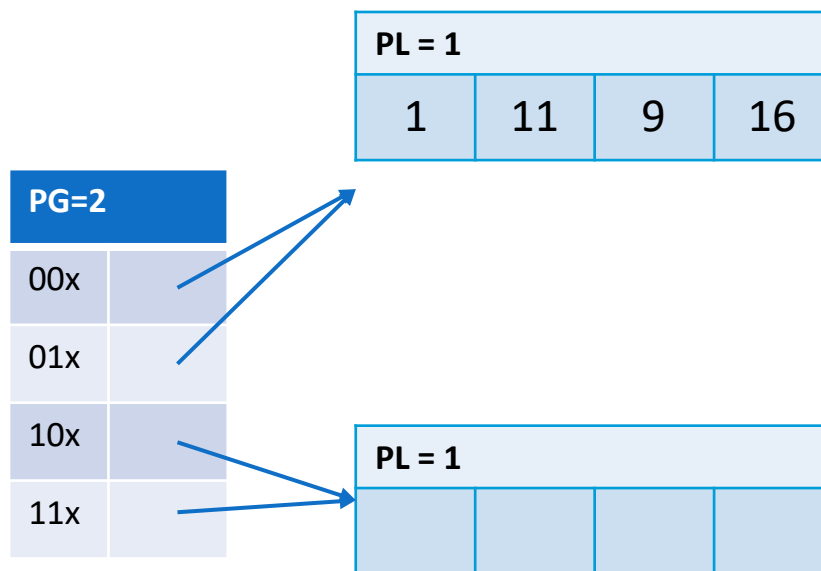


# Hashing dinámico

- Insertamos: 1, 11, 9, 16, 3, 32, 15, 4, 2, 19, 7, 14, 17, 27



clave	H(k)	Binario
1	1	001
11	3	011
9	1	001
16	0	000
3	3	011
32	0	000
15	7	111
4	4	100
2	2	010
19	3	011
7	7	111
14	6	110
17	1	001
27	3	011

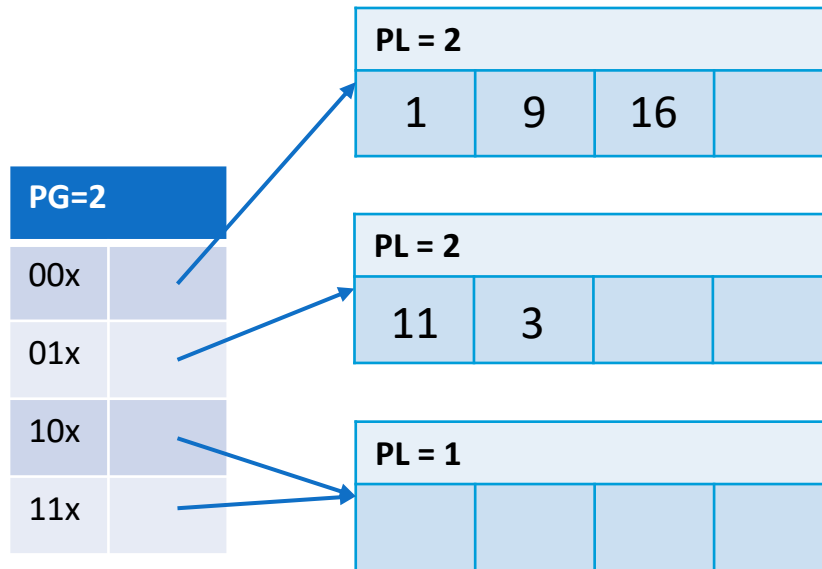


# Hashing dinámico

- Insertamos: 1, 11, 9, 16, 3, 32, 15, 4, 2, 19, 7, 14, 17, 27



clave	H(k)	Binario
1	1	001
11	3	011
9	1	001
16	0	000
3	3	011
32	0	000
15	7	111
4	4	100
2	2	010
19	3	011
7	7	111
14	6	110
17	1	001
27	3	011

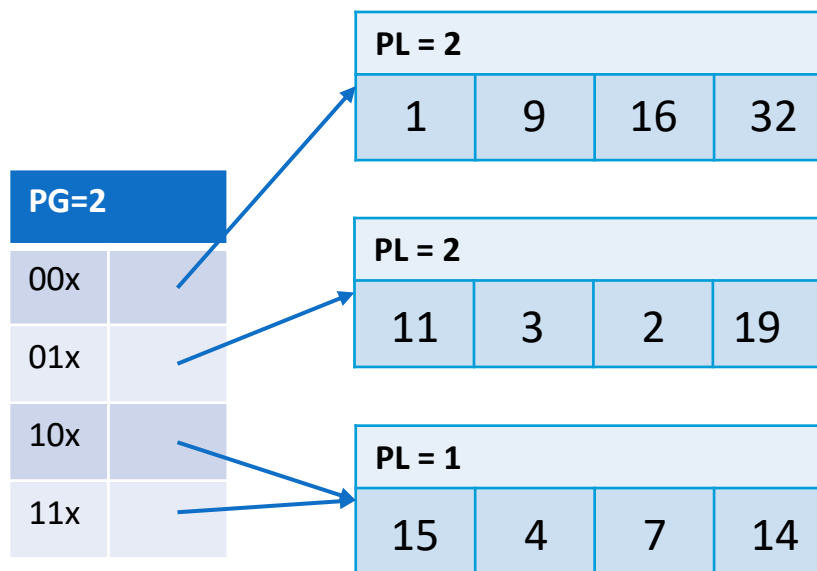


# Hashing dinámico

- Insertamos: 1, 11, 9, 16, 3, 32, 15, 4, 2, 19, 7, 14, 17, 27



clave	H(k)	Binario
1	1	001
11	3	011
9	1	001
16	0	000
3	3	011
32	0	000
15	7	111
4	4	100
2	2	010
19	3	011
7	7	111
14	6	110
17	1	001
27	3	011

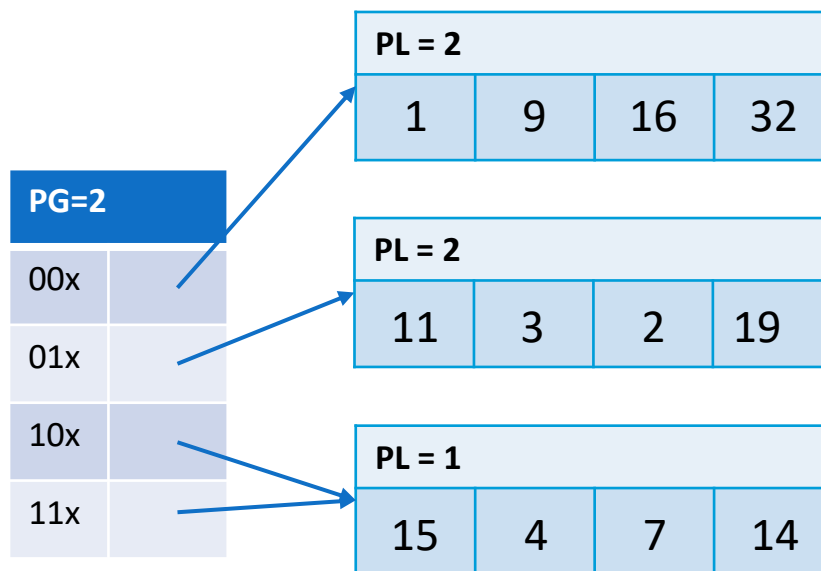


# Hashing dinámico

- Insertamos: 1, 11, 9, 16, 3, 32, 15, 4, 2, 19, 7, 14, 17, 27



clave	H(k)	Binario
1	1	001
11	3	011
9	1	001
16	0	000
3	3	011
32	0	000
15	7	111
4	4	100
2	2	010
19	3	011
7	7	111
14	6	110
17	1	001
27	3	011

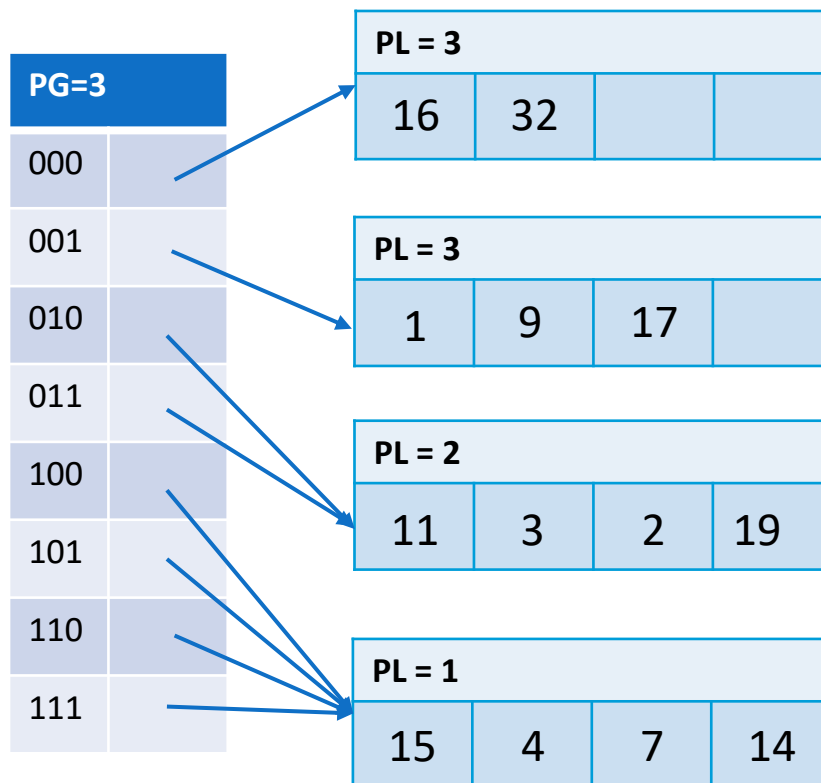


# Hashing dinámico

- Insertamos: 1, 11, 9, 16, 3, 32, 15, 4, 2, 19, 7, 14, 17, 27



clave	H(k)	Binario
1	1	001
11	3	011
9	1	001
16	0	000
3	3	011
32	0	000
15	7	111
4	4	100
2	2	010
19	3	011
7	7	111
14	6	110
17	1	001
27	3	011

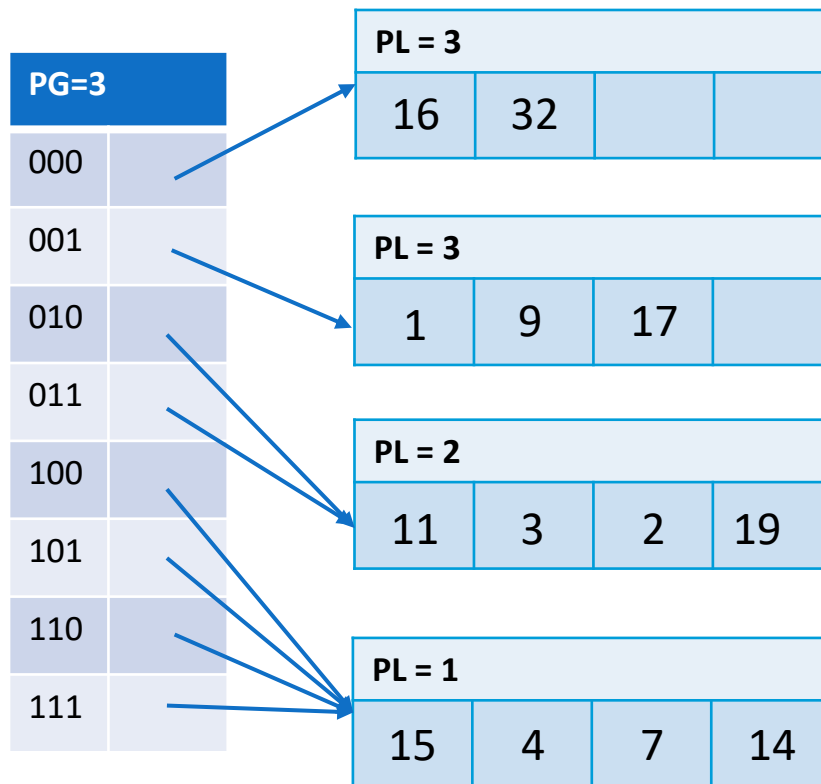


# Hashing dinámico

- Insertamos: 1, 11, 9, 16, 3, 32, 15, 4, 2, 19, 7, 14, 17, 27



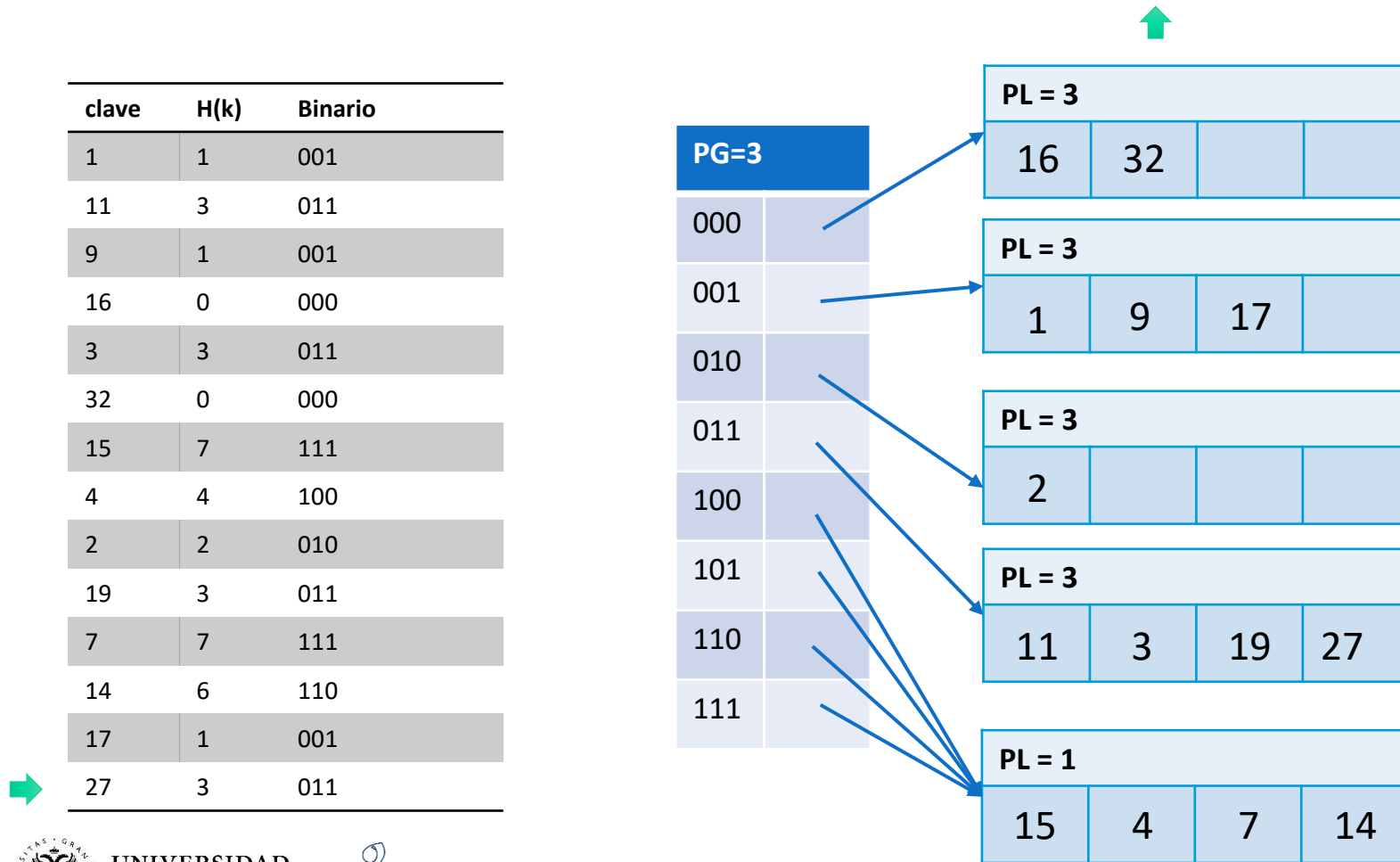
clave	H(k)	Binario
1	1	001
11	3	011
9	1	001
16	0	000
3	3	011
32	0	000
15	7	111
4	4	100
2	2	010
19	3	011
7	7	111
14	6	110
17	1	001
27	3	011





# Hashing dinámico

- Insertamos: 1, 11, 9, 16, 3, 32, 15, 4, 2, 19, 7, 14, 17, 27



# Hashing dinámico

- El hashing dinámico supera los problemas clásicos del acceso directo.
- También tiene sus inconvenientes:
  - Utilizar una tabla índice adicional (nuevos accesos a disco si no cabe en memoria).
  - El tamaño de la tabla depende de “d”.

# Contenidos

- Acceso directo
- Hashing básico
- Hashing dinámico

# Imágenes

- Imágenes tomadas de [Pixabay](#)
  - Portada
    - Imagen de [Manfred Steger](#)  
<https://pixabay.com/es/vectors/pixelchen-conferencia-profesor-3976299/>
  - Cabecera
    - Imágenes de [Gerd Altmann](#)  
<https://pixabay.com/es/illustrations/digitalizaci%C3%B3n-placa-de-circuito-6892340/>