

CC3201-1

BASES DE DATOS

OTOÑO 2021

Clase 9: Indexación y  
Optimización de Consultas

Aidan Hogan

[aidhog@gmail.com](mailto:aidhog@gmail.com)

# Las preguntas de hoy

| Planeta  |       |       |      |          |         |      |        |
|----------|-------|-------|------|----------|---------|------|--------|
| nombre   | dist  | radio | grav | días     | años    | temp | anillo |
| Mercurio | 0,39  | 0,38  | 2,8  | 58,646   | 0,241   | 440  | false  |
| Venus    | 0,72  | 0,95  | 8,9  | -243,019 | 0,615   | 730  | false  |
| Tierra   | 1,00  | 1,00  | 9,8  | 0,997    | 1,000   | 288  | false  |
| Marte    | 1,52  | 0,53  | 3,7  | 1,026    | 1,880   | 186  | false  |
| Júpiter  | 5,20  | 10,97 | 22,9 | 0,414    | 11,862  | 152  | true   |
| Saturno  | 9,54  | 9,14  | 9,1  | 0,444    | 29,447  | 134  | true   |
| Urano    | 19,19 | 3,98  | 7,8  | -0,719   | 84,017  | 76   | true   |
| Neptuno  | 30,07 | 3,86  | 11,0 | 0,671    | 164,791 | 53   | true   |

| Satélite  |         |                    |      |
|-----------|---------|--------------------|------|
| nombre    | planeta | descubridor        | año  |
| Luna      | Tierra  | ⊥                  | ⊥    |
| Ganímedes | Júpiter | Galileo Galilei    | 1610 |
| Calisto   | Júpiter | Galileo Galilei    | 1610 |
| Europa    | Júpiter | Galileo Galilei    | 1610 |
| Ío        | Júpiter | Galileo Galilei    | 1610 |
| Titán     | Saturno | Christiaan Huygens | 1655 |
| Tritón    | Neptuno | William Lassell    | 1846 |

| Aterrizaje    |          |      |      |
|---------------|----------|------|------|
| nave          | planeta  | país | año  |
| Messenger     | Mercurio | EEUU | 2015 |
| Venera 3      | Venus    | URRS | 1966 |
| Pioneer       | Venus    | EEUU | 1978 |
| Mars 2 lander | Marte    | URRS | 1971 |
| Viking 1      | Marte    | EEUU | 1976 |
| Beagle 2      | Marte    | ESA  | 2003 |
| Galileo       | Júpiter  | EEUU | 2003 |

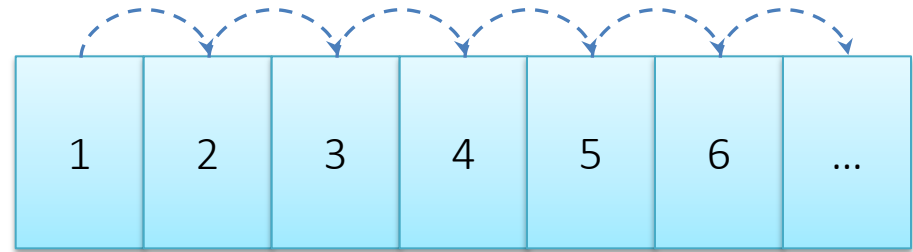
¿Cómo se deberían guardar las tablas en la máquina?

Y ¿cómo se pueden optimizar las consultas sobre estas tablas?

# MEMORIA

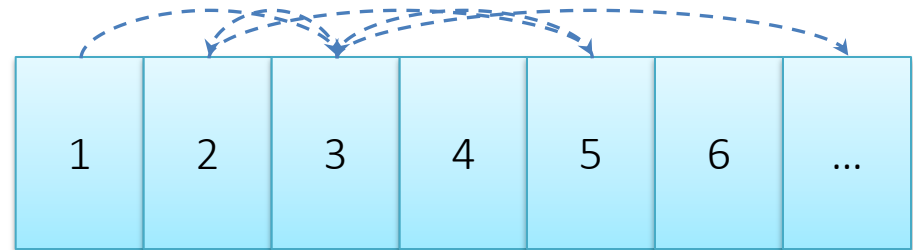
# Acceso secuencial

Memoria:

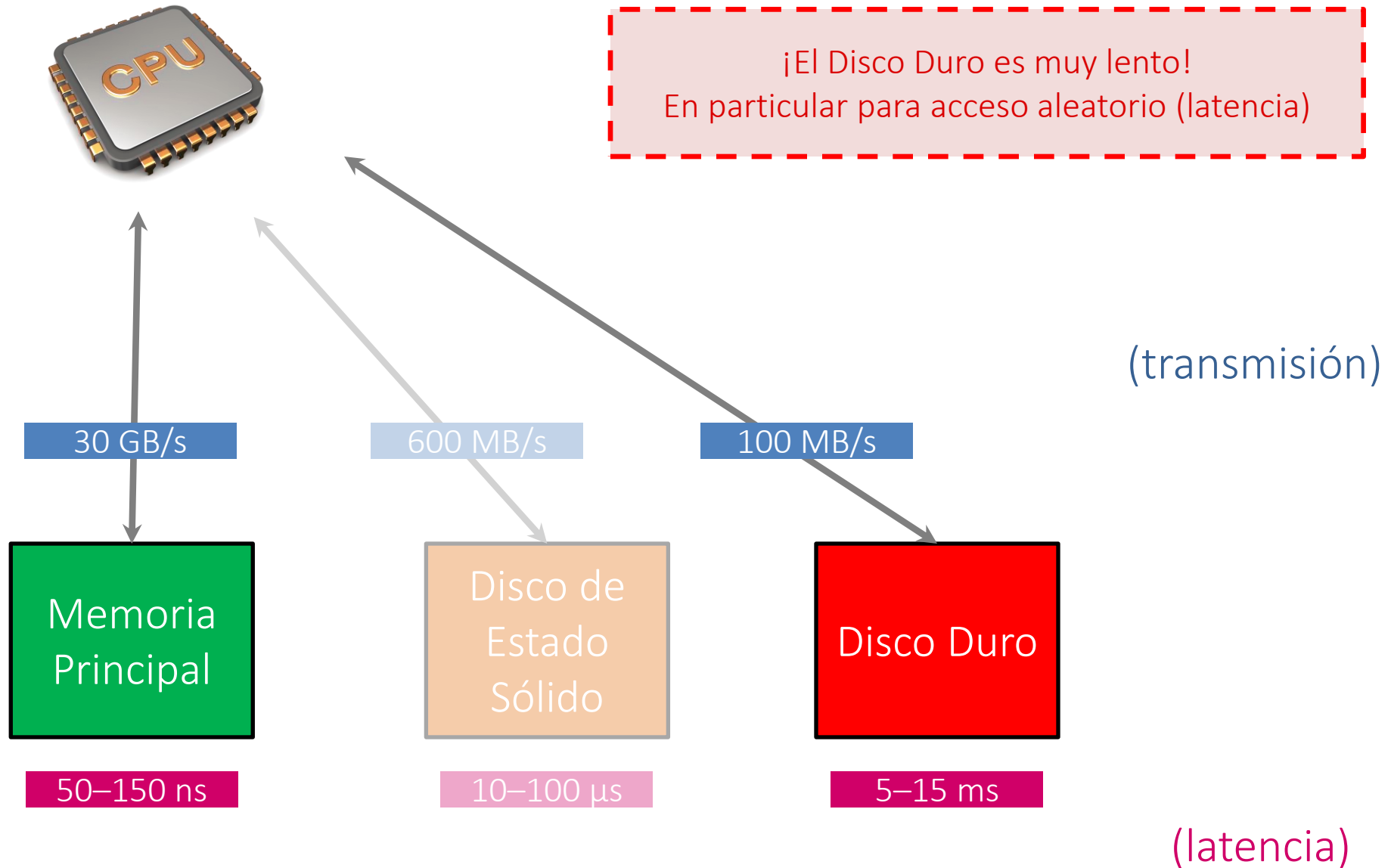


# Acceso “aleatorio”

Memoria:



# Costos de acceder a los datos (estimaciones)



# Costos de acceder a los datos (estimaciones)

*¿Por qué usamos el disco duro entonces?*



Memoria  
Principal

Más rápida ✓

- ✓ Más barato
- ✓ Persistente



Disco Duro

# Alta latencia de los discos duros

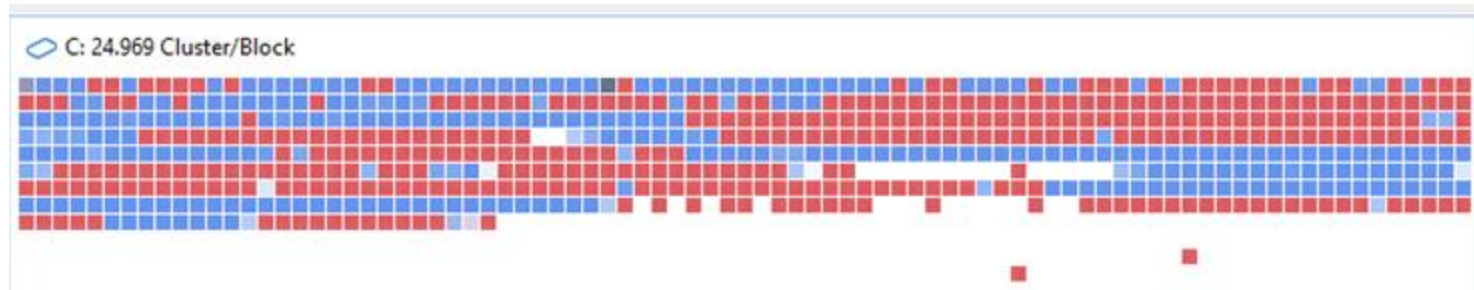
*¿Por qué la latencia de discos duros es tan alta?*



*Tiene un brazo mecánico que tiene que moverse para cambiar el bloque*

# Bloques en el disco duro

| Drive | Name    | Action | Status | Total files | Frag. files | Degree of fragmentation | Size      | Free      | File system | Current file/folder | Rep |
|-------|---------|--------|--------|-------------|-------------|-------------------------|-----------|-----------|-------------|---------------------|-----|
| C:    | System  | Ready  | 0%     | 232.771     | 379         | 0,67%                   | 194,96 GB | 104,42 GB | NTFS        |                     |     |
| D:    |         | Ready  | 0%     | 41          | 0           | 0,00%                   | 10,00 GB  | 9,95 GB   | NTFS        |                     |     |
| E:    | Data II | Ready  | 0%     | 121.351     | 8.398       | 52,06%                  | 172,78 GB | 108,62 GB | NTFS        |                     |     |



*Más eficiente al nivel de hardware.  
Evite espacio no usable.*

# SISTEMA DE COSTOS (CLÁSICO)

# Memoria Principal vs. Memoria Secundaria



Memoria  
Secundaria

- Datos guardados en memoria secundaria
- La lectura se hace por **bloques**
- Un bloque tiene un tamaño de  $B$  tuplas



Memoria  
Principal

- Los datos son llevados a memoria principal
- La memoria tiene una capacidad de  $M$  tuplas

# Sistema de Costos



## Memoria Secundaria

- Datos guardados en memoria secundaria
- La lectura se hace por **bloques**
- Un bloque tiene un tamaño de  $B$  tuplas

### *Sistema de Costos (Clásico):*

*Cuenta los accesos a la memoria secundaria (el disco duro).*

*Cada vez que se lee o se escribe, se suma 1 al costo.*

*Asume que las operaciones en memoria principal toman un tiempo despreciable.*



## Memoria Principal

- Los datos son llevados a memoria principal
- La memoria tiene una capacidad de  $M$  tuplas

# Sistema de Costos

- Un bloque tiene un tamaño de  $B$  tuplas
- La memoria tiene una capacidad de  $M$  tuplas

¿Cuánto cuesta leer  $n$  tuplas desde la memoria secundaria?

$$\lceil \frac{n}{B} \rceil$$

¿Cuántos bloques caben en memoria?

$$\lfloor \frac{M}{B} \rfloor$$

¿Cuántos bloques usa una relación  $R$ ?

$$\lceil \frac{|R|}{B} \rceil$$

# PROCESAMIENTO DE CONSULTAS

# SQL es un lenguaje **declarativo**

Uno dice lo que quiere, no cómo debería ser computado

## (1) Selección/producto:

```
SELECT DISTINCT A.nombre, A.genero
FROM actor A,
     personaje P1, personaje P2
WHERE P1.a_nombre='Tyler, Liv'
      AND P1.p_nombre = P2.p_nombre
      AND P1.p_anho = P2.p_anho
      AND A.nombre = P2.a_nombre
```

## (2) Join explícito:

```
SELECT DISTINCT A.nombre, A.genero
FROM actor A NATURAL JOIN
     ( SELECT DISTINCT P2.a_nombre AS nombre
       FROM personaje P2 NATURAL JOIN
         ( SELECT DISTINCT P1.p_nombre, P1.p_anho
           FROM personaje P1
           WHERE P1.a_nombre='Tyler, Liv'
         ) PLT
     ) CLT
```

## (3) Consulta anidada (FROM):

```
SELECT DISTINCT A.nombre, A.genero FROM
     ( SELECT DISTINCT P2.a_nombre FROM
       ( SELECT DISTINCT P1.p_nombre, P1.p_anho
         FROM personaje P1
         WHERE P1.a_nombre='Tyler, Liv'
       ) PLT, personaje P2
       WHERE PLT.p_nombre = P2.p_nombre
         AND PLT.p_anho = P2.p_anho
     ) CLT, actor A
WHERE CLT.a_nombre = A.nombre
```

## (4) Consulta anidada (WHERE/IN):

```
SELECT DISTINCT A.nombre, A.genero
FROM actor A
WHERE A.nombre IN
     ( SELECT DISTINCT P2.a_nombre
       FROM personaje P2
       WHERE (P2.p_nombre,P2.p_anho) IN
         ( SELECT DISTINCT P1.p_nombre, P1.p_anho
           FROM personaje P1
           WHERE P1.a_nombre='Tyler, Liv'
         )
     )
```

Caja Negra



| nombre                | genero |
|-----------------------|--------|
| Abbott, Jane (II)     | F      |
| Acevedo, Gino (I)     | M      |
| Allpress, Bruce       | M      |
| Appleby, Noel         | M      |
| Appleton, Matt (I)    | M      |
| Astin, Ali            | F      |
| Astin, Sean           | M      |
| Aston, David (I)      | M      |
| Bach, John (I)        | M      |
| Baker, Sala           | M      |
| Bartlett, Timothy     | M      |
| Bean, Sean            | M      |
| Benzon, Jarl          | M      |
| Benzon, Jørn          | M      |
| Beynon-Cole, Victoria | F      |
| Blanchett, Cate       | F      |
| Bloom, Orlando        | M      |
| Boyd, Billy (I)       | M      |
| Boyens, Callum        | M      |
| Britton, Ben (I)      | M      |
| Britton, Ben (VI)     | M      |
| Brophy, Jed           | M      |
| Brophy, Riley         | M      |
| Brophy, Sadwyn        | M      |
| Browning, Alistair    | M      |
| Bryson, Paul (I)      | M      |
| Burnyeat, Luke        | M      |
| Clentworth, Rachel    | F      |
| Comery, Sam           | M      |
| Corrigan, Peter (II)  | M      |
| Crossen, Sabine       | F      |

# SQL es un lenguaje **declarativo**

Uno dice lo que quiere, no cómo debería ser computado

(1) Selección/producto:

*Idealmente*, el motor puede elegir el mejor plan de ejecución independientemente de la expresión particular de la consulta

(2) Join explícito:

El usuario no tiene que preocuparse por la optimización de la consulta

(3) Consulta anidada (FROM):

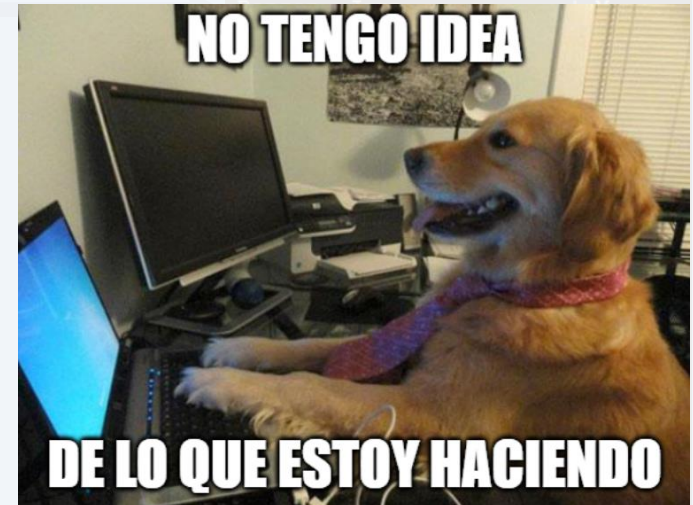
```
SELECT DISTINCT A.nombre, A.genero FROM
( SELECT DISTINCT P2.a_nombre FROM
  ( SELECT DISTINCT P1.p_nombre, P1.p_anho
    FROM personaje P1
    WHERE P1.a_nombre='Tyler, Liv'
  ) PLT, personaje P2
  WHERE PLT.p_nombre = P2.p_nombre
  AND PLT.p_anho = P2.p_anho
) CLT, actor A
WHERE CLT.a_nombre = A.nombre
```

(4) Consulta anidada (WHERE/IN):

```
SELECT DISTINCT A.nombre, A.genero
FROM actor A
WHERE A.nombre IN
( SELECT DISTINCT P2.a_nombre
  FROM personaje P2
  WHERE (P2.p_nombre,P2.p_anho) IN
    ( SELECT DISTINCT P1.p_nombre, P1.p_anho
      FROM personaje P1
      WHERE P1.a_nombre='Tyler, Liv'
    )
  )
```

Caja Negra

| nombre             | genero |
|--------------------|--------|
| Acevedo, Gina (I)  | F      |
| Acevedo, Gina (I)  | M      |
| Acevedo, Gina (I)  | M      |
| Acevedo, Gina (I)  | M      |
| Appleton, Matt (I) | M      |
| Astin, Ali         | F      |
| Astin, Sean        | M      |
| Aston, David (I)   | M      |
| Bach, John (I)     | M      |
| Bach, John (I)     | M      |
| Bach, John (I)     | M      |
| Bean, Sean         | M      |
| Benzon, Jarl       | M      |



BÚSQUEDA

# Búsqueda

- Devolver todas las tuplas de una relación que satisfagan alguna condición

| Planeta       |       |       |      |          |         |      |        |
|---------------|-------|-------|------|----------|---------|------|--------|
| <u>nombre</u> | dist  | radio | grav | días     | años    | temp | anillo |
| Mercurio      | 0,39  | 0,38  | 2,8  | 58,646   | 0,241   | 440  | false  |
| Venus         | 0,72  | 0,95  | 8,9  | -243,019 | 0,615   | 730  | false  |
| Tierra        | 1,00  | 1,00  | 9,8  | 0,997    | 1,000   | 288  | false  |
| Marte         | 1,52  | 0,53  | 3,7  | 1,026    | 1,880   | 186  | false  |
| Júpiter       | 5,20  | 10,97 | 22,9 | 0,414    | 11,862  | 152  | true   |
| Saturno       | 9,54  | 9,14  | 9,1  | 0,444    | 29,447  | 134  | true   |
| Urano         | 19,19 | 3,98  | 7,8  | -0,719   | 84,017  | 76   | true   |
| Neptuno       | 30,07 | 3,86  | 11,0 | 0,671    | 164,791 | 53   | true   |

```
SELECT *  
FROM Planeta  
WHERE nombre = 'Venus'
```

| <u>nombre</u> | dist | radio | grav | días     | años  | temp | anillo |
|---------------|------|-------|------|----------|-------|------|--------|
| Venus         | 0,72 | 0,95  | 8,9  | -243,019 | 0,615 | 730  | false  |

# Búsqueda Secuencial

- Se leen todas las tuplas de la relación  $R$
- Se seleccionan las que cumplan la condición

| Planeta  |       |       |      |          |         |      |        |
|----------|-------|-------|------|----------|---------|------|--------|
| nombre   | dist  | radio | grav | días     | años    | temp | anillo |
| Mercurio | 0,39  | 0,38  | 2,8  | 58,646   | 0,241   | 440  | false  |
| Venus    | 0,72  | 0,95  | 8,9  | -243,019 | 0,615   | 730  | false  |
| Tierra   | 1,00  | 1,00  | 9,8  | 0,997    | 1,000   | 288  | false  |
| Marte    | 1,52  | 0,53  | 3,7  | 1,026    | 1,880   | 186  | false  |
| Júpiter  | 5,20  | 10,97 | 22,9 | 0,414    | 11,862  | 152  | true   |
| Saturno  | 9,54  | 9,14  | 9,1  | 0,444    | 29,447  | 134  | true   |
| Urano    | 19,19 | 3,98  | 7,8  | -0,719   | 84,017  | 76   | true   |
| Neptuno  | 30,07 | 3,86  | 11,0 | 0,671    | 164,791 | 53   | true   |

¿Cuántas tuplas se leen?

$|R|$

¿Cuánto cuesta en términos de bloques?

$\lceil \frac{|R|}{B} \rceil$

¿Cómo podemos optimizar la búsqueda?

...

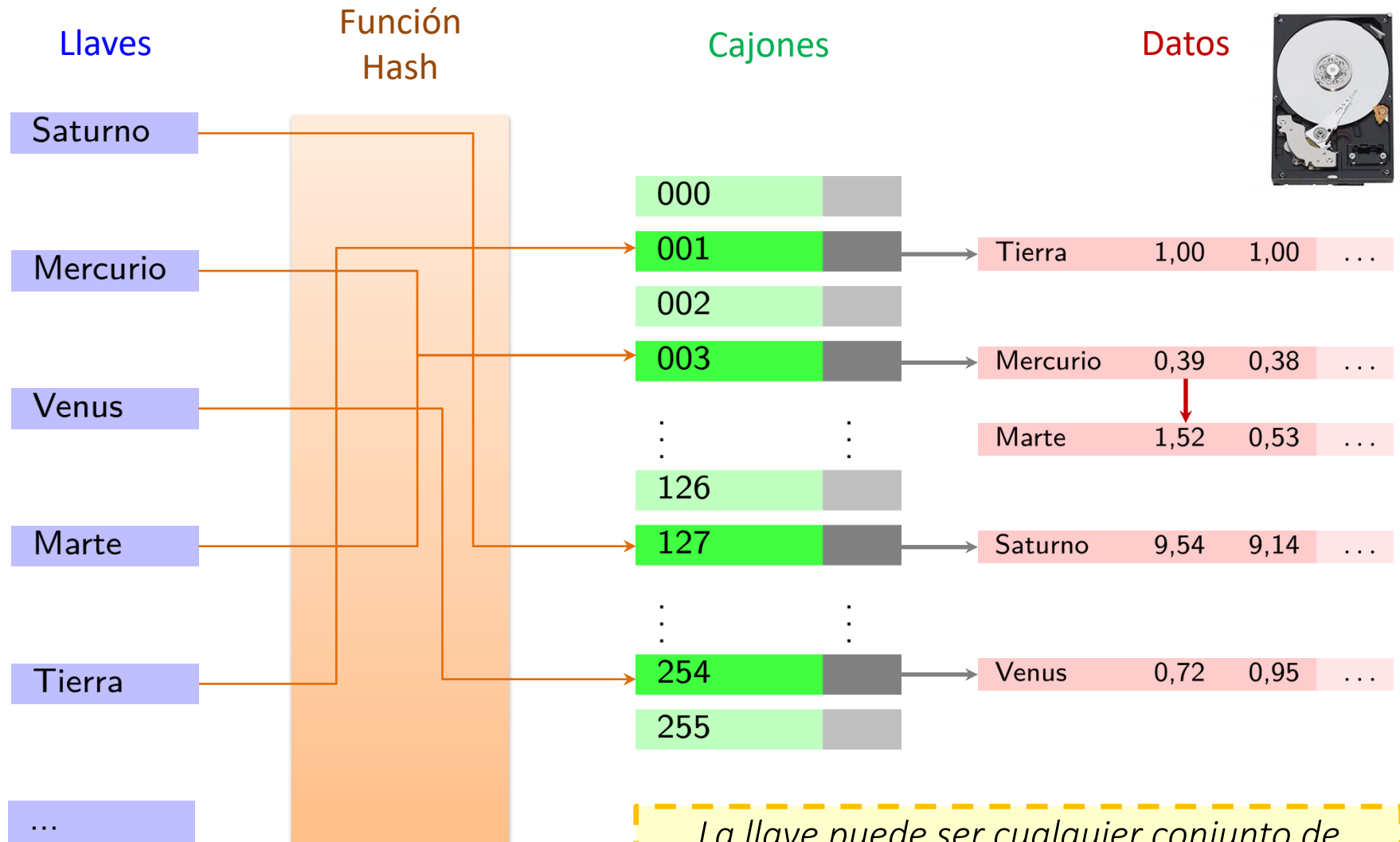
# ÍNDICES

# Índice

- Estructura los datos para facilitar búsquedas

| NamesandNumbers.com |                                                                    |
|---------------------|--------------------------------------------------------------------|
| Wood River Valley   |                                                                    |
| 522-7481            | BATES Paul 118 Willow Rd.....Hailey 788-1206                       |
| 788-3933            | BATES Steve 105 Audubon Pl.....Hailey 788-6222                     |
| 788-9263            | BATES VICKY - INTERIOR MOTIVES PO Box 1820.....Sun Valley 788-5950 |
| 788-9933            | BATHUM Roy 235 Spur Ln.....Ketchum 726-0722                        |
| 578-0595            | BATMAN.....See West Adam                                           |
| 788-8979            | BATT Jeffrey & Camille.....726-7494                                |
| 788-2515            | BATTERSBY Patricia 116 Ritchie Dr.....Ketchum 726-8896             |
| 720-5661            | BAUER Charlotte 621 Northstar Dr.....Hailey 788-4279               |
| 28-7219             | BAUER CHARLOTTE LINDBERG                                           |
| 38-2317             | Radiance Skin Care Studio.....Hailey 578-2214                      |
|                     | BAUER Matt 3340 Woodside Blvd.....Hailey 578-0703                  |
|                     | BAUER Rich.....720-0165                                            |

# Índice Hash



*La llave puede ser cualquier conjunto de atributos de la tabla.*

# Índice Hash

Diagram illustrating the components of a hash index:

- Llaves (Keys)
- Función Hash (Hash Function)
- Cajones (Buckets)
- Datos (Data)

¿El costo de búsqueda con un índice hash?

Caso ideal:  $O(1)$  para devolver una tupla

Peor caso:  $O(\lceil \frac{|R|}{B} \rceil)$

Asumiremos una búsqueda que devuelva una sola tupla.

Para devolver  $k$  tuplas, hay que “sumar”  $\lceil \frac{k}{B} \rceil$  al costo.

Caso ideal:  $O(\max(1, \lceil \frac{k}{B} \rceil))$  para devolver  $k$  tuplas

# Índice Hash

Llaves Función Hash Cajones Datos

¿El costo de búsqueda con un índice hash?

*Caso ideal:*  $O(1)$  para devolver una tupla

*Peor caso:*  $O(\lceil \frac{|R|}{B} \rceil)$

Pero ¿qué tipo de búsqueda?

*¡Hemos asumido que se sabe el valor exacto de la llave!*

```
SELECT *  
FROM Planeta  
WHERE nombre = 'Venus'
```

# Índice Hash

Llaves      Función Hash      Cajones      Datos

¿El costo de búsqueda con un índice hash?

Caso ideal:  $O(1)$  para devolver una tupla

Peor caso:  $O(\lceil \frac{|R|}{B} \rceil)$

¿Qué pasa con las búsquedas por rango?

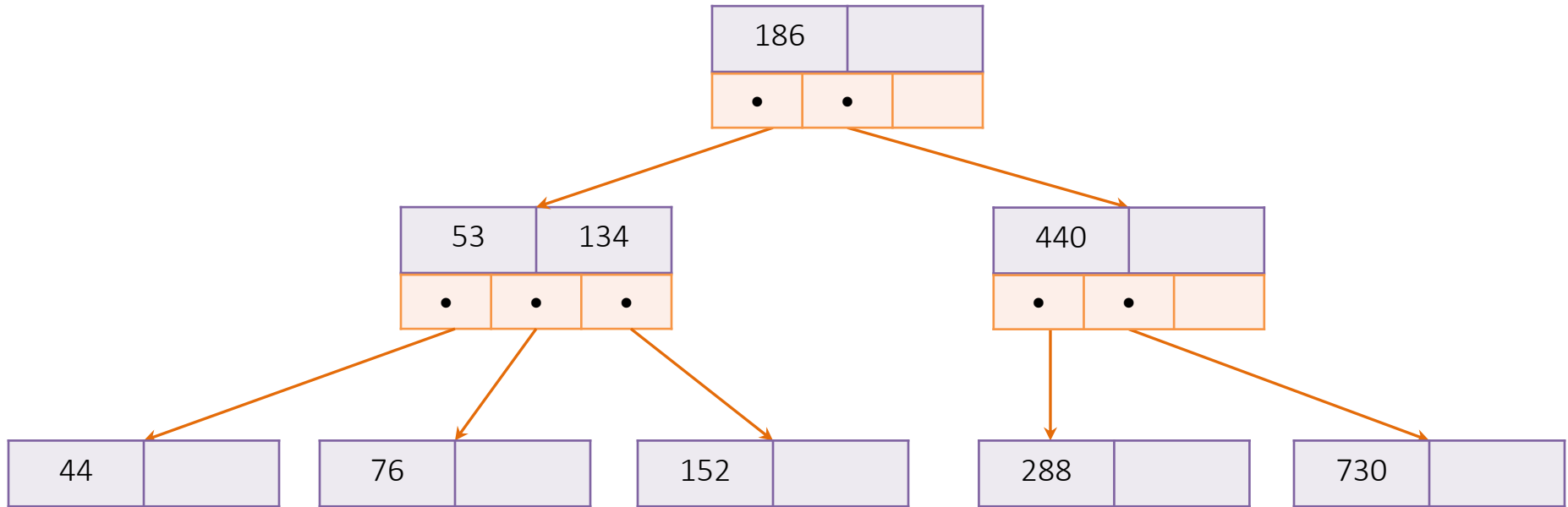
```
SELECT *  
FROM Planeta  
WHERE temp > 100  
AND temp < 200
```

Buscar  $n$  valores (ideal):  $O(n)$

Leer todo:  $O(\lceil \frac{|R|}{B} \rceil)$

# Índice Árbol B

| Planeta | nombre   | dist  | radio | grav | días     | años    | temp | anillo |
|---------|----------|-------|-------|------|----------|---------|------|--------|
|         | Mercurio | 0,39  | 0,38  | 2,8  | 58,646   | 0,241   | 440  | false  |
|         | Venus    | 0,72  | 0,95  | 8,9  | -243,019 | 0,615   | 730  | false  |
|         | Tierra   | 1,00  | 1,00  | 9,8  | 0,997    | 1,000   | 288  | false  |
|         | Marte    | 1,52  | 0,53  | 3,7  | 1,026    | 1,880   | 186  | false  |
|         | Júpiter  | 5,20  | 10,97 | 22,9 | 0,414    | 11,862  | 152  | true   |
|         | Saturno  | 9,54  | 9,14  | 9,1  | 0,444    | 29,447  | 134  | true   |
|         | Urano    | 19,19 | 3,98  | 7,8  | -0,719   | 84,017  | 76   | true   |
|         | Neptuno  | 30,07 | 3,86  | 11,0 | 0,671    | 164,791 | 53   | true   |
|         | Pluto    | 39,48 | 0,19  | 0,1  | 6,387    | 248,000 | 44   | false  |



*Árbol B: Árbol ordenado, balanceado*

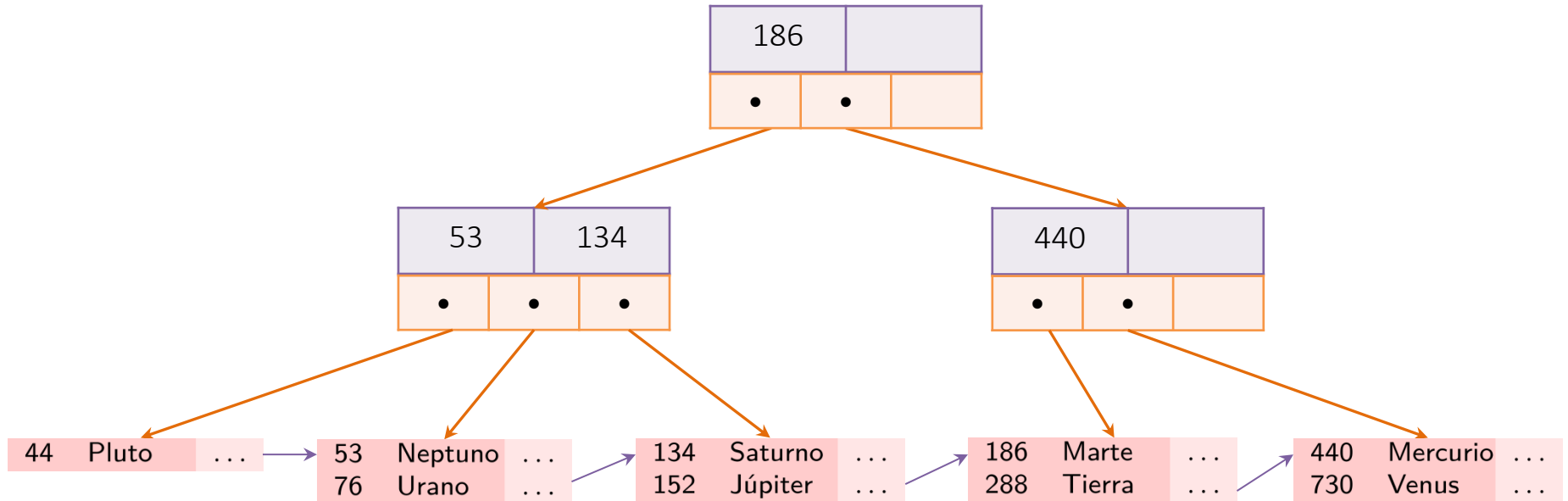
***a-b** Árbol B: Los nodos internos tienen entre  $a$  y  $b$  hijos ( $a \geq \lceil \frac{b}{2} \rceil$ )*

¿Este caso?

2-3 Árbol B

# Índice Árbol B+

| Planeta | nombre   | dist  | radio | grav | días     | años    | temp | anillo |
|---------|----------|-------|-------|------|----------|---------|------|--------|
|         | Mercurio | 0,39  | 0,38  | 2,8  | 58,646   | 0,241   | 440  | false  |
|         | Venus    | 0,72  | 0,95  | 8,9  | -243,019 | 0,615   | 730  | false  |
|         | Tierra   | 1,00  | 1,00  | 9,8  | 0,997    | 1,000   | 288  | false  |
|         | Marte    | 1,52  | 0,53  | 3,7  | 1,026    | 1,880   | 186  | false  |
|         | Júpiter  | 5,20  | 10,97 | 22,9 | 0,414    | 11,862  | 152  | true   |
|         | Saturno  | 9,54  | 9,14  | 9,1  | 0,444    | 29,447  | 134  | true   |
|         | Urano    | 19,19 | 3,98  | 7,8  | -0,719   | 84,017  | 76   | true   |
|         | Neptuno  | 30,07 | 3,86  | 11,0 | 0,671    | 164,791 | 53   | true   |
|         | Pluto    | 39,48 | 0,19  | 0,1  | 6,387    | 248,000 | 44   | false  |



*Árbol B+:* Árbol B donde se guardan las tuplas en las hojas del árbol.

Las hojas guardan todas las llaves y sus valores.

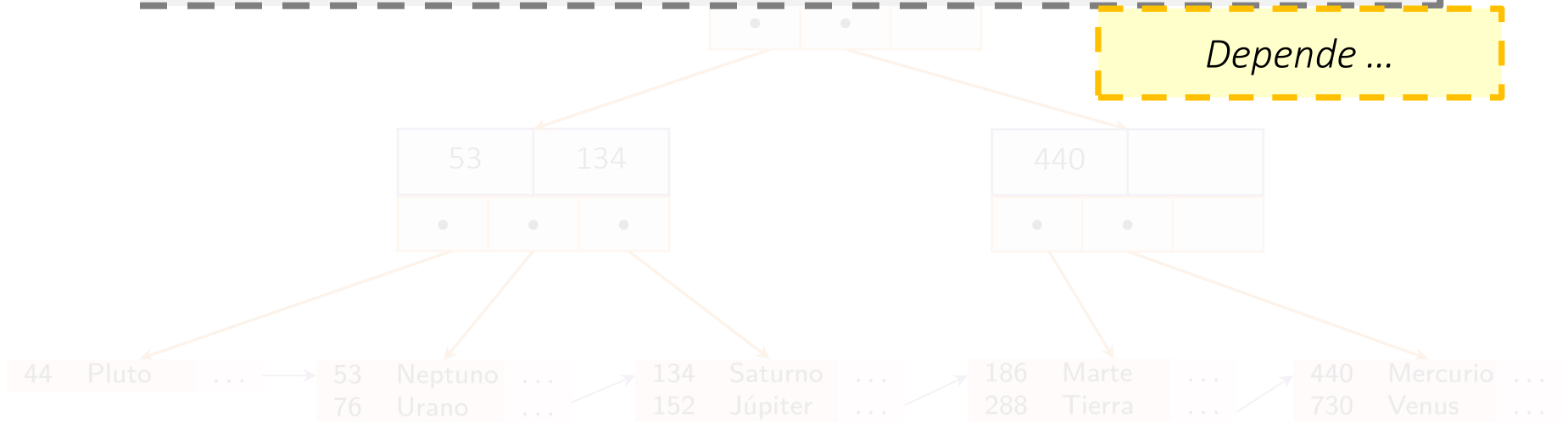
Se conectan las hojas para poder hacer búsquedas por rango más eficientes.

# Índice Árbol B+

| Planeta  |       |       |      |          |         |      |        |
|----------|-------|-------|------|----------|---------|------|--------|
| nombre   | dist  | radio | grav | días     | años    | temp | anillo |
| Mercurio | 0,39  | 0,38  | 2,8  | 58,646   | 0,241   | 440  | false  |
| Venus    | 0,72  | 0,95  | 8,9  | -243,019 | 0,615   | 730  | false  |
| Tierra   | 1,00  | 1,00  | 9,8  | 0,997    | 1,000   | 288  | false  |
| Marte    | 1,52  | 0,53  | 3,7  | 1,026    | 1,880   | 186  | false  |
| Júpiter  | 5,20  | 10,97 | 22,9 | 0,414    | 11,862  | 152  | true   |
| Saturno  | 9,54  | 9,14  | 9,1  | 0,444    | 29,447  | 134  | true   |
| Urano    | 19,19 | 3,98  | 7,8  | -0,719   | 84,017  | 76   | true   |
| Neptuno  | 30,07 | 3,86  | 11,0 | 0,671    | 164,791 | 53   | true   |
| Pluto    | 39,48 | 0,19  | 0,1  | 6,387    | 248,000 | 44   | false  |

¿El costo de búsqueda con un árbol B+?

Depende ...

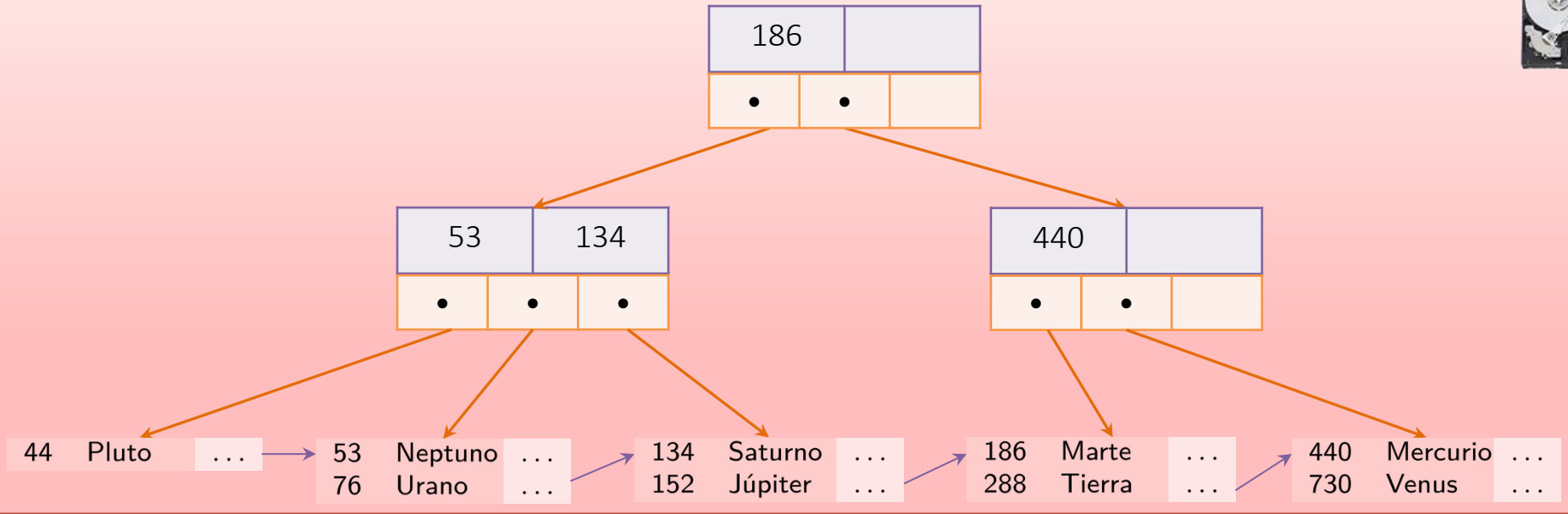


*Árbol B+: Árbol B donde se guardan las tuplas en las hojas del árbol.*

*Las hojas guardan todas las llaves y sus valores.*

*Se conectan las hojas para poder hacer búsquedas por rango más eficientes.*

# Índice Árbol B+

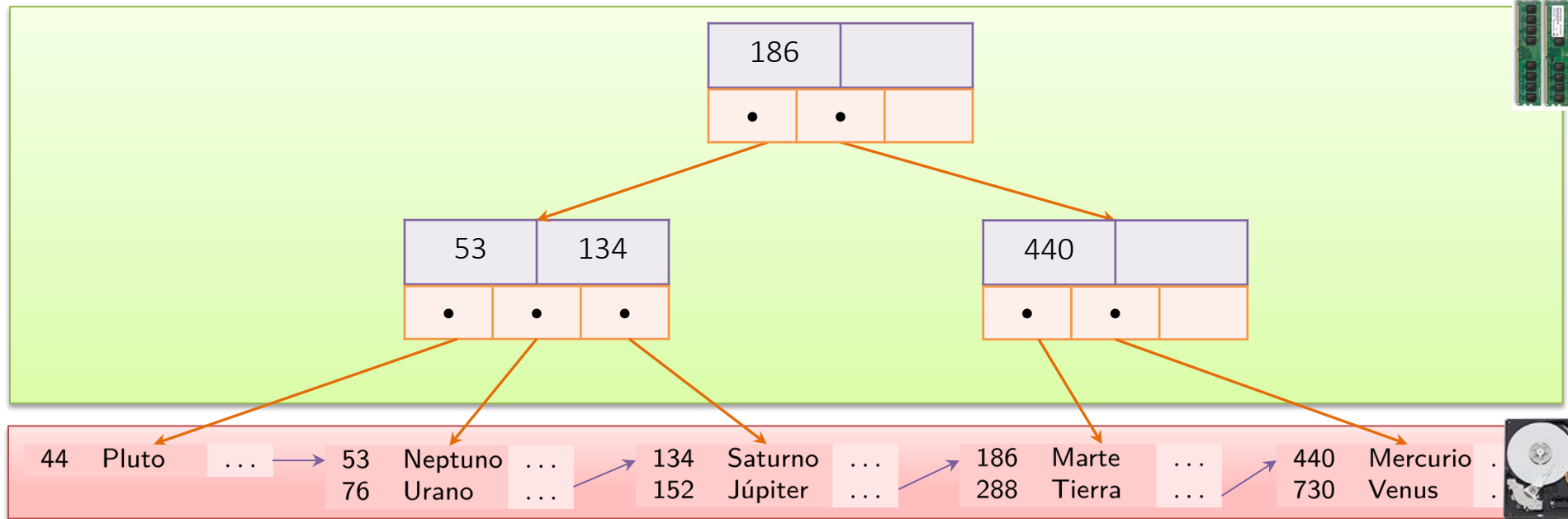


¿El costo en términos de bloques leídos de memoria secundaria?

*Si se guarda cada nodo como un bloque en memoria secundaria:*

$O(\log_b(\lceil \frac{|R|}{B} \rceil))$  para devolver una tupla

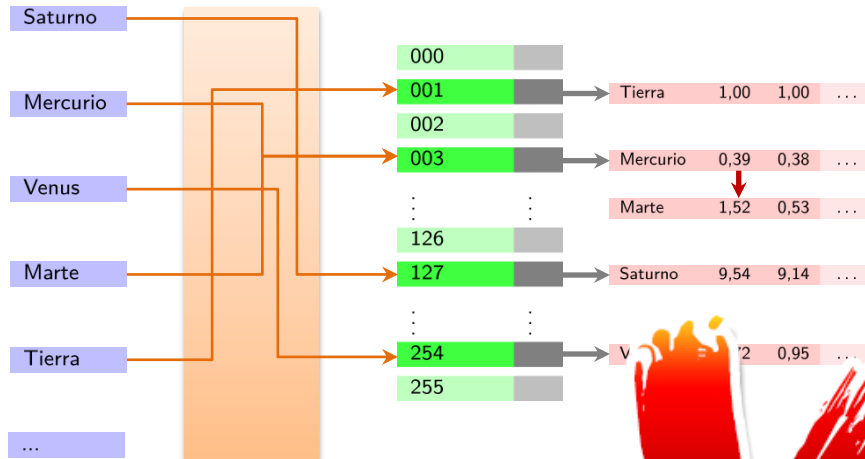
# Índice Árbol B+



¿El costo en términos de bloques leídos de memoria secundaria?

*Si se cachean la raíz y los nodos internos en memoria principal:  
 $O(1)$  para devolver una tupla*

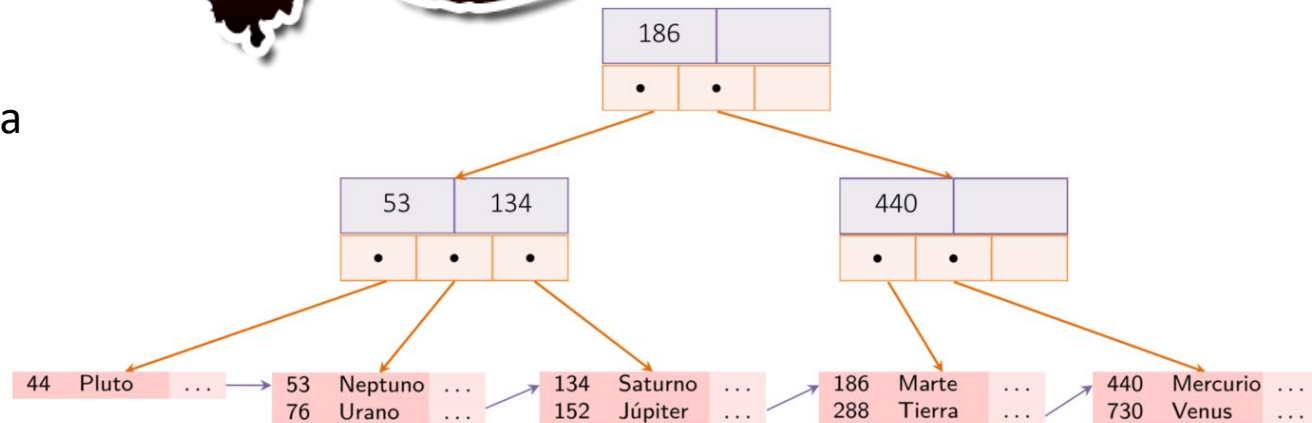
# Índices: Hash vs. Árbol B+



Levemente más eficiente para búsquedas exactas asumiendo una función de hash ideal

*Árbol B+ es más popular*

Mucho más eficiente para búsquedas por rango



# Crear Índices: SQL

- Por defecto, se crea un índice para la llave primaria de la tabla
- Para crear/borrar índices sobre otros atributos:

```
CREATE INDEX nombre ON tabla (atr1,atr2) -- btree por defecto
```

```
CREATE INDEX nombre ON tabla USING hash (atr1,atr2)
```

```
DROP INDEX nombre
```

JOINS

# Reunir tablas: (EQUI) JOIN

| Planeta  |       |       |      |          |         |      |        |
|----------|-------|-------|------|----------|---------|------|--------|
| nombre   | dist  | radio | grav | días     | años    | temp | anillo |
| Mercurio | 0,39  | 0,38  | 2,8  | 58,646   | 0,241   | 440  | false  |
| Venus    | 0,72  | 0,95  | 8,9  | -243,019 | 0,615   | 730  | false  |
| Tierra   | 1,00  | 1,00  | 9,8  | 0,997    | 1,000   | 288  | false  |
| Marte    | 1,52  | 0,53  | 3,7  | 1,026    | 1,880   | 186  | false  |
| Júpiter  | 5,20  | 10,97 | 22,9 | 0,414    | 11,862  | 152  | true   |
| Saturno  | 9,54  | 9,14  | 9,1  | 0,444    | 29,447  | 134  | true   |
| Urano    | 19,19 | 3,98  | 7,8  | -0,719   | 84,017  | 76   | true   |
| Neptuno  | 30,07 | 3,86  | 11,0 | 0,671    | 164,791 | 53   | true   |

| Aterrizaje    |          |      |      |
|---------------|----------|------|------|
| nave          | planeta  | país | año  |
| Messenger     | Mercurio | EEUU | 2015 |
| Venera 3      | Venus    | URRS | 1966 |
| Pioneer       | Venus    | EEUU | 1978 |
| Mars 2 lander | Marte    | URRS | 1971 |
| Viking 1      | Marte    | EEUU | 1976 |
| Beagle 2      | Marte    | ESA  | 2003 |
| Galileo       | Júpiter  | EEUU | 2003 |

```
SELECT nave, nombre, dist, año
FROM Planeta, Aterrizaje
WHERE nombre = planeta
```

¿Cómo deberíamos ejecutar el join?

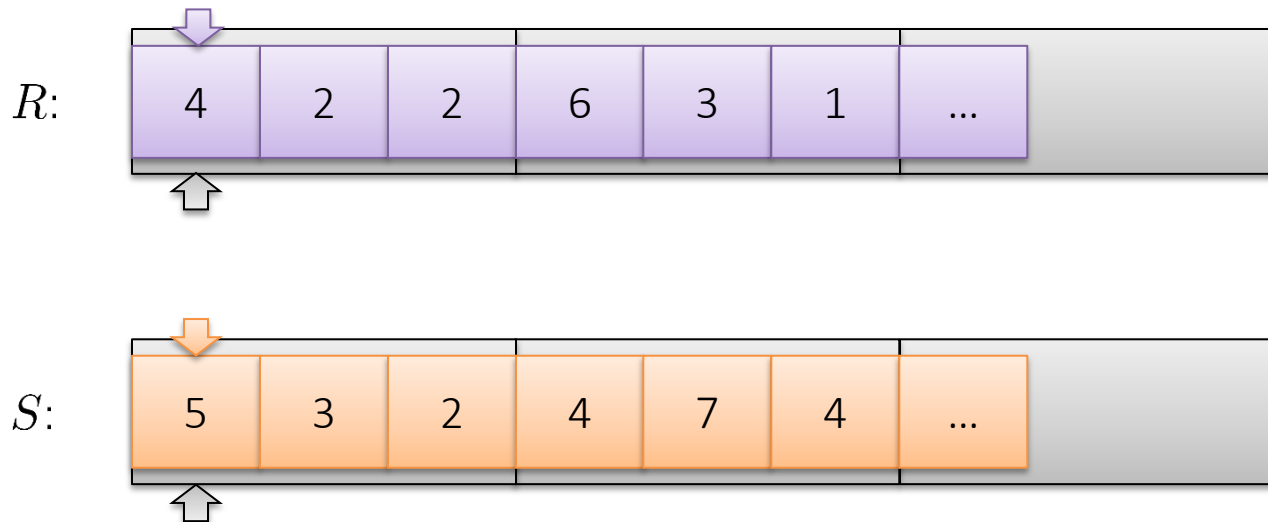
| nave          | nombre   | dist | año  |
|---------------|----------|------|------|
| Messenger     | Mercurio | 0,39 | 2015 |
| Venera 3      | Venus    | 0,72 | 1966 |
| Pioneer       | Venus    | 0,72 | 1978 |
| Mars 2 lander | Marte    | 1,52 | 1971 |
| Viking 1      | Marte    | 1,52 | 1976 |
| Beagle 2      | Marte    | 1,52 | 2003 |
| Galileo       | Júpiter  | 5,20 | 2003 |

# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$

$R \bowtie S$

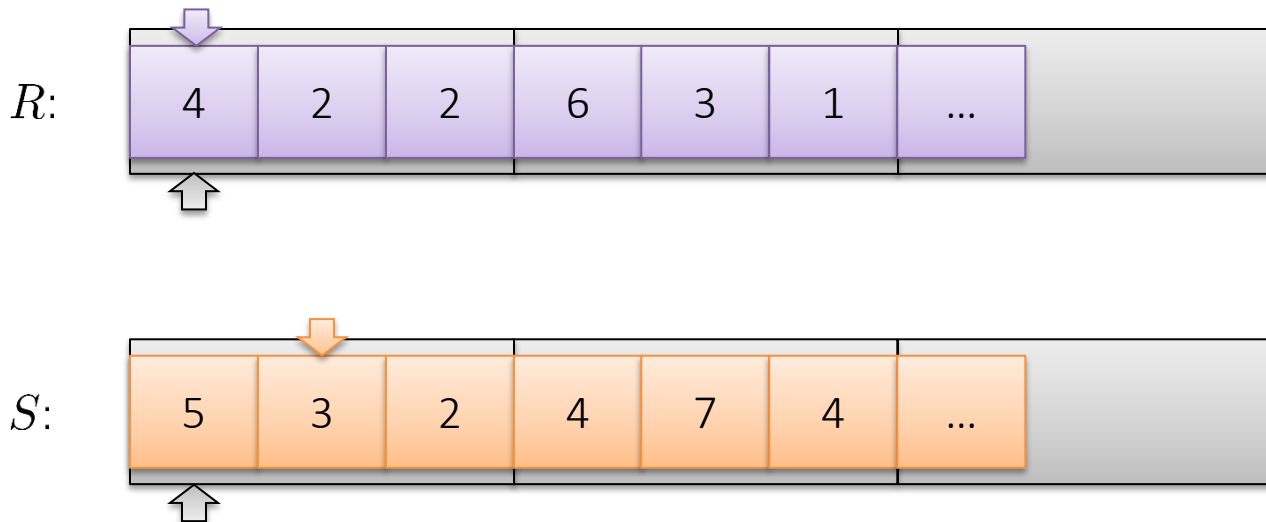


# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$

$R \bowtie S$

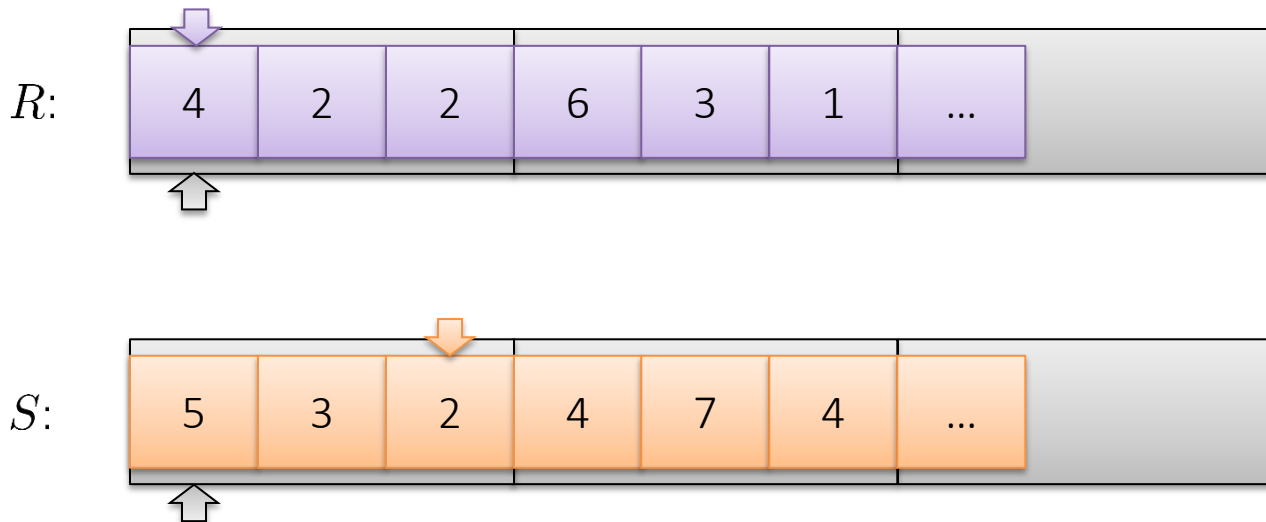


# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$

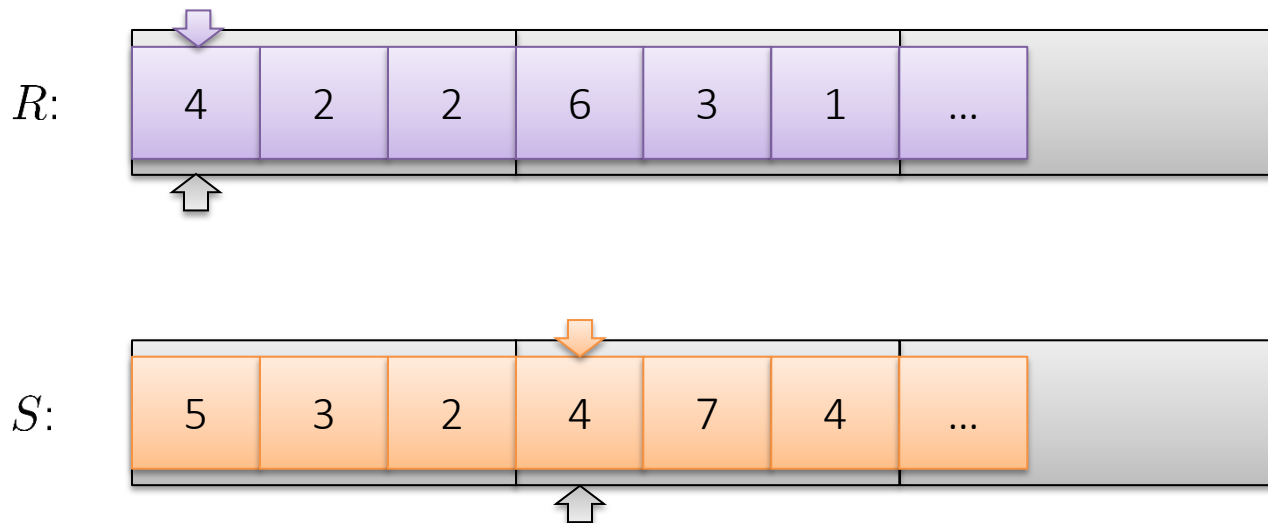
$R \bowtie S$



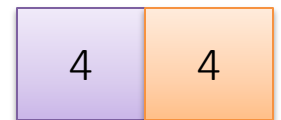
# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$



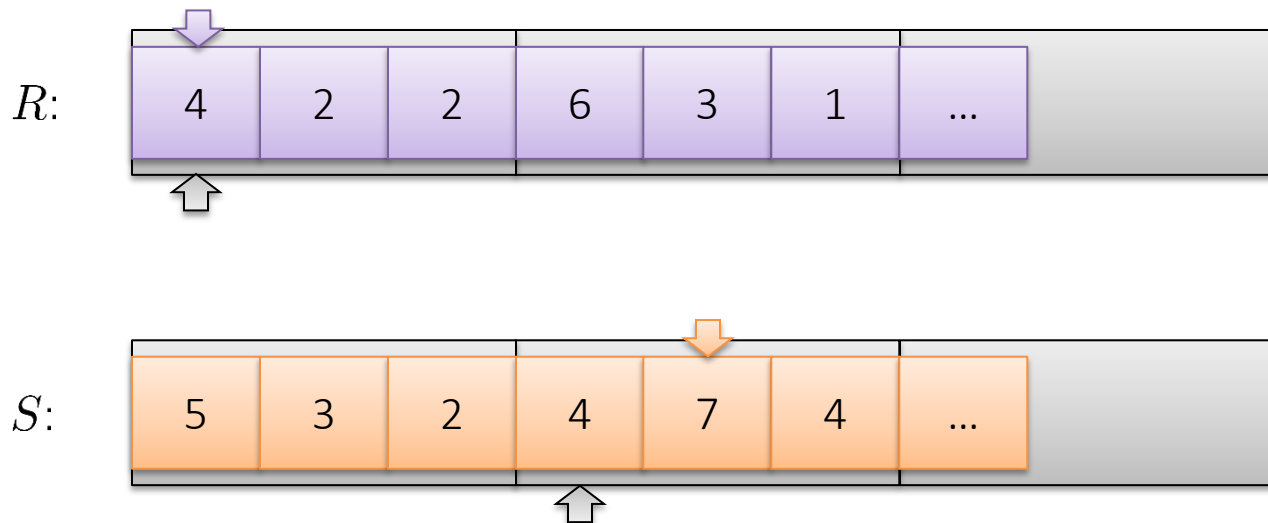
$R \bowtie S$



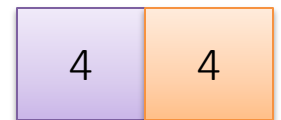
# Loop anidado (sin índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$



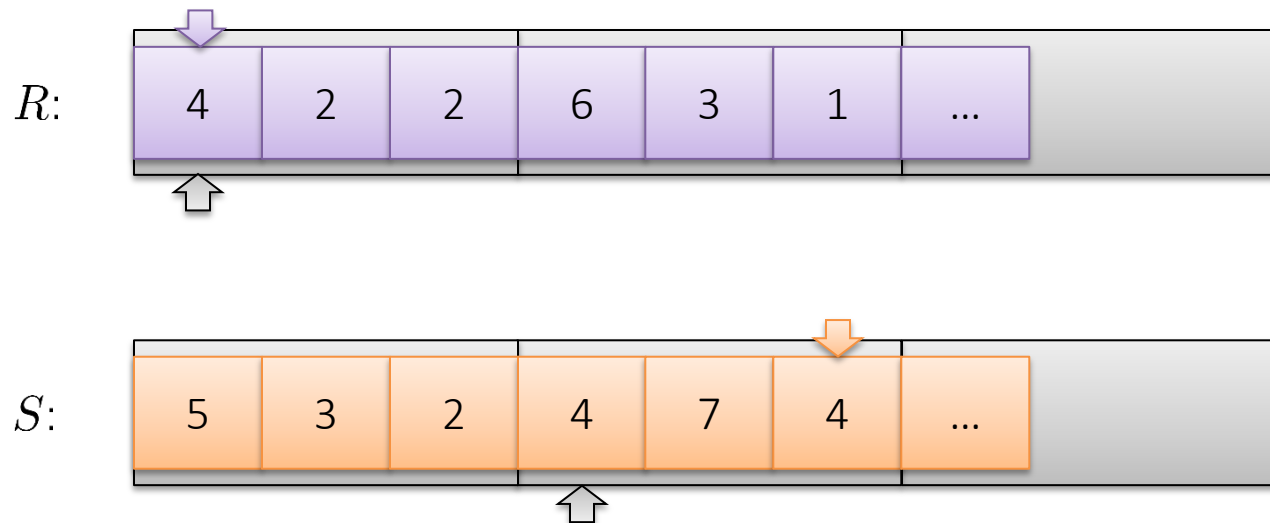
$$R \bowtie S$$



# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$



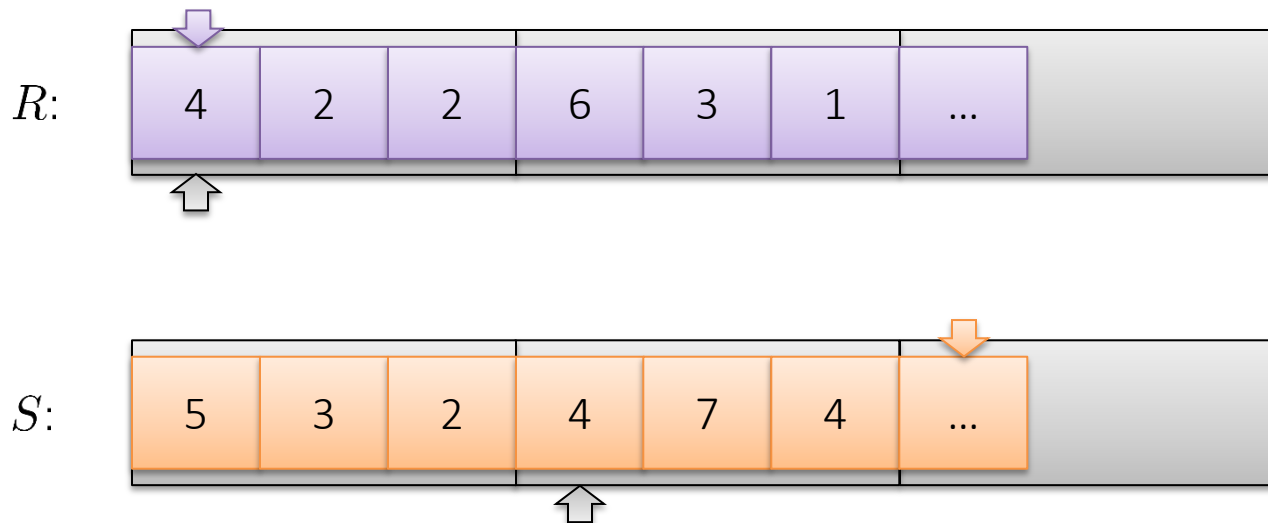
$R \bowtie S$

|   |   |
|---|---|
| 4 | 4 |
| 4 | 4 |

# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$



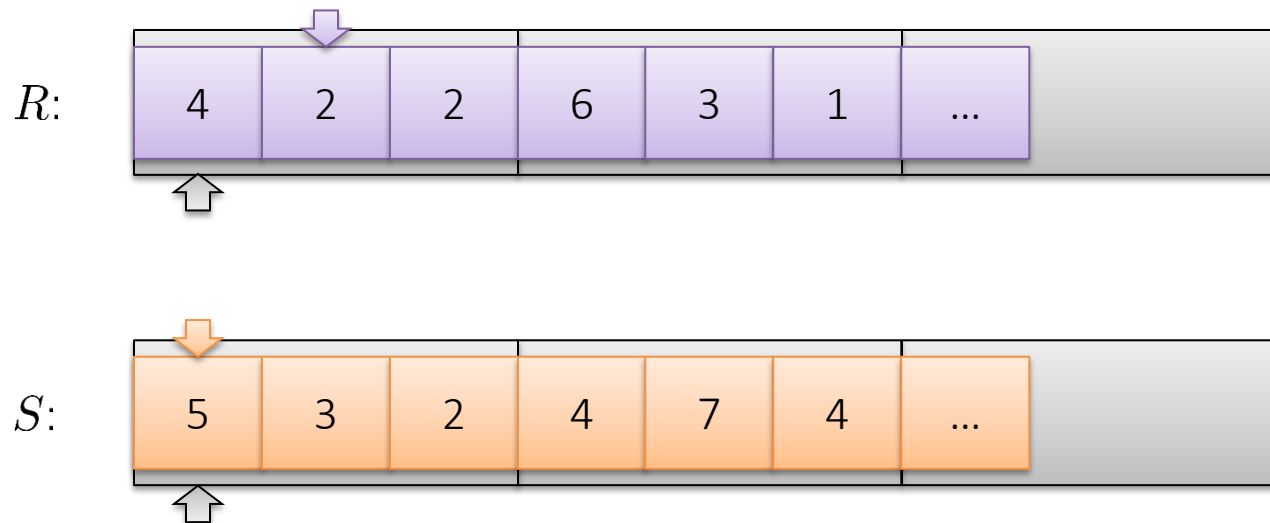
$R \bowtie S$

|   |   |
|---|---|
| 4 | 4 |
| 4 | 4 |

# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$



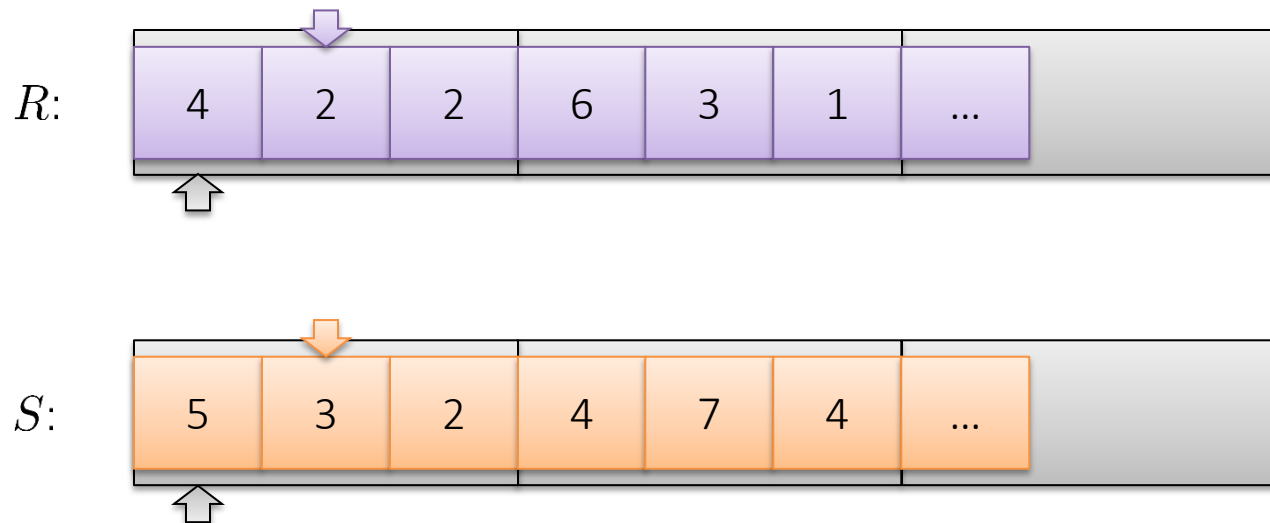
$R \bowtie S$

|   |   |
|---|---|
| 4 | 4 |
| 4 | 4 |

# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$



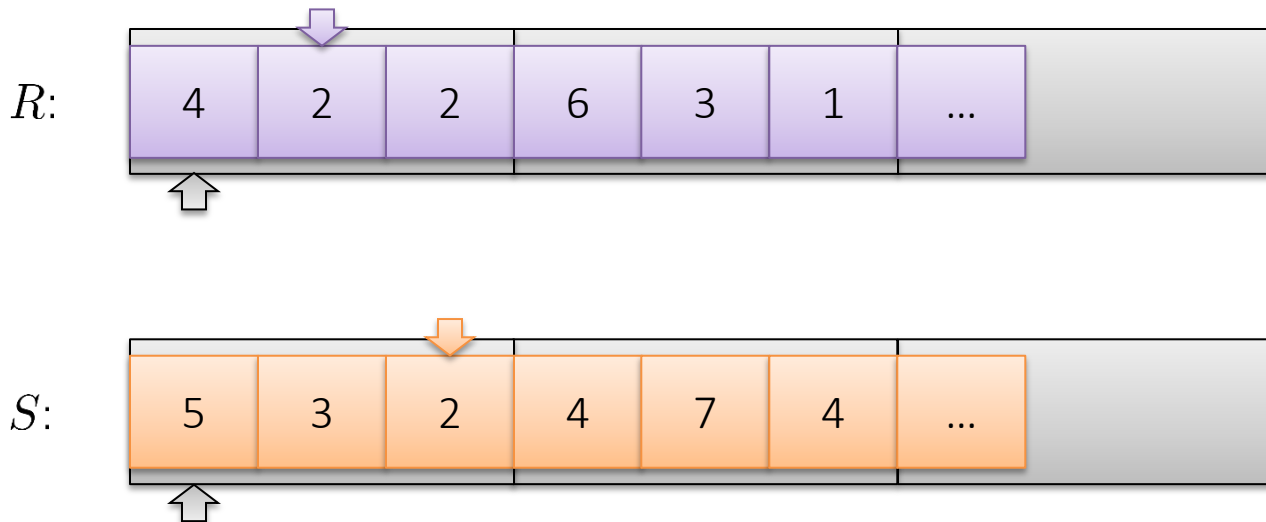
$R \bowtie S$

|   |   |
|---|---|
| 4 | 4 |
| 4 | 4 |

# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$



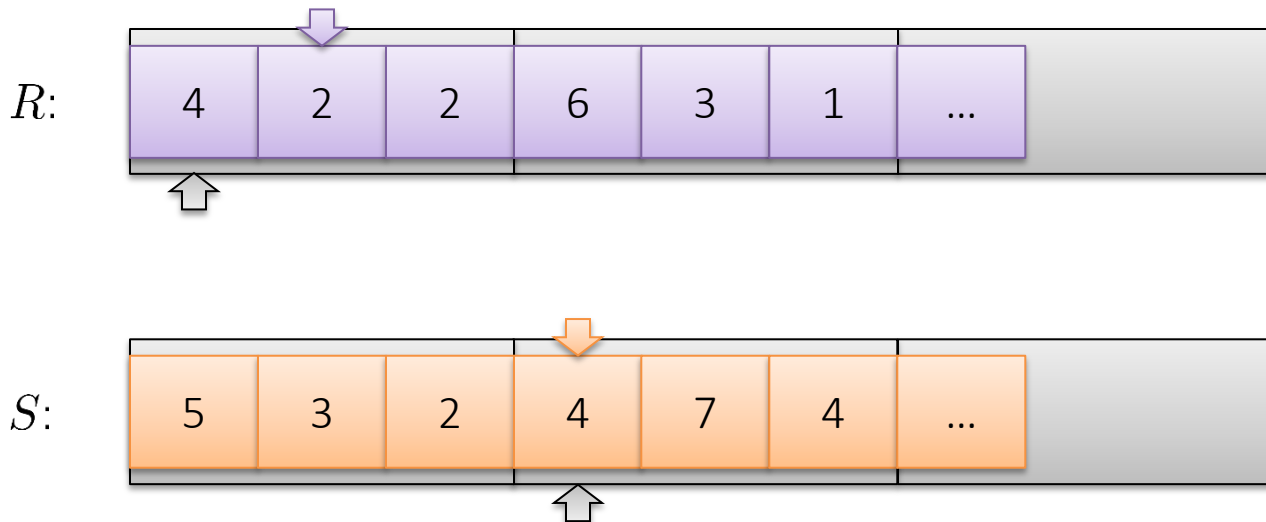
$R \bowtie S$

|   |   |
|---|---|
| 4 | 4 |
| 4 | 4 |
| 2 | 2 |

# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$



$R \bowtie S$

|   |   |
|---|---|
| 4 | 4 |
| 4 | 4 |
| 2 | 2 |

...

# Loop anidado (sin índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Para cada tupla  $s \in S$ 
    - Si  $r$  y  $s$  satisfacen el join:  
escribir  $\{r\} \times \{s\}$

¿Costo?

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot \lceil \frac{|S|}{B} \rceil$$

¿Memoria?

$2B$  tuplas

¿Elegir  $R$  y  $S$ ?

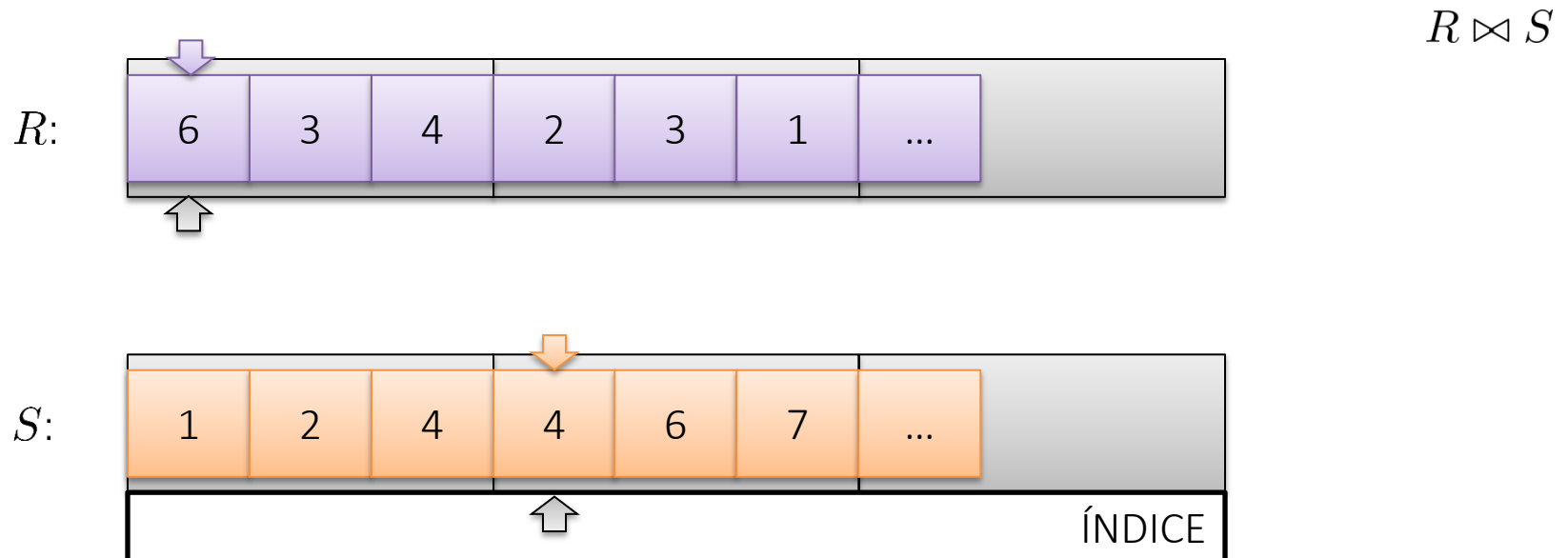
$|R| < |S|$  (para ahorrar tiempo)

¿Podemos optimizar esta forma de hacer joins?

# Loop anidado (con índices)

$R \bowtie S$

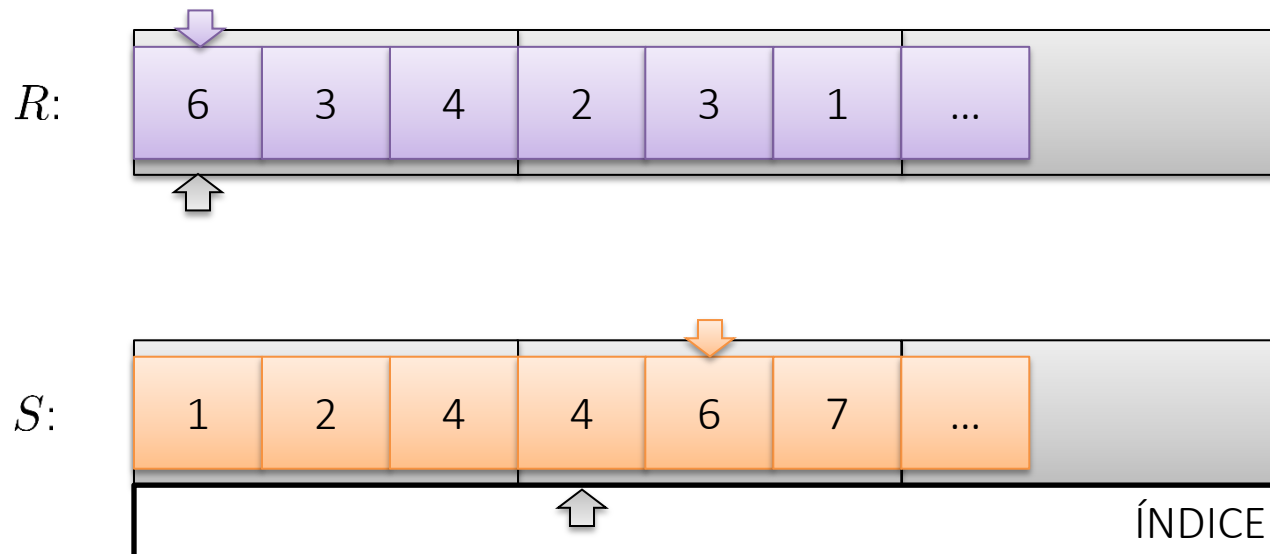
- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



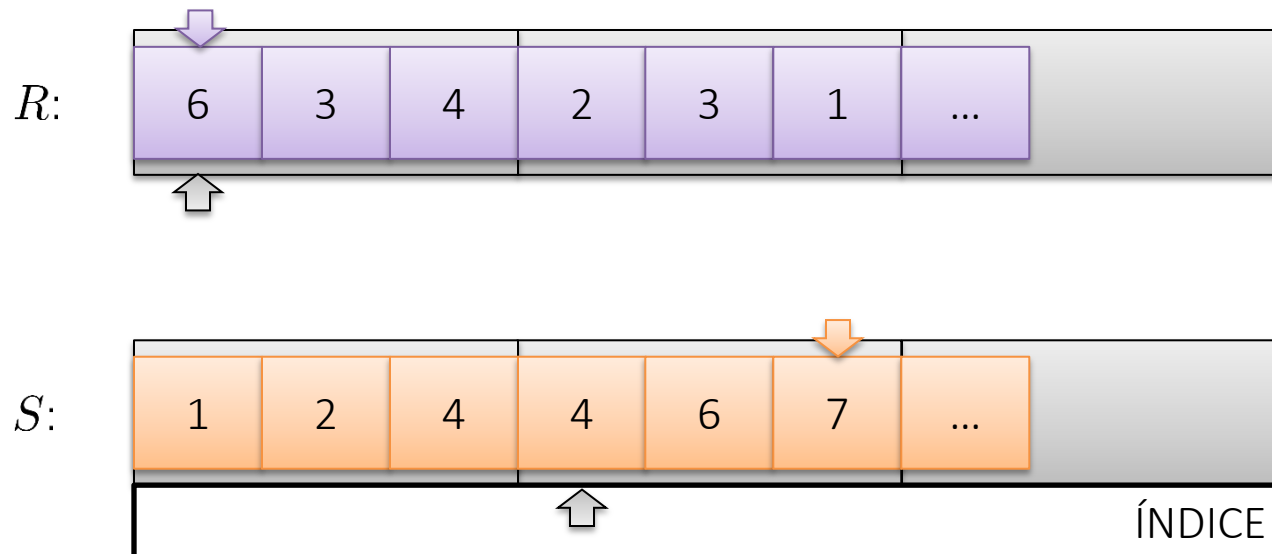
$$R \bowtie S$$



# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



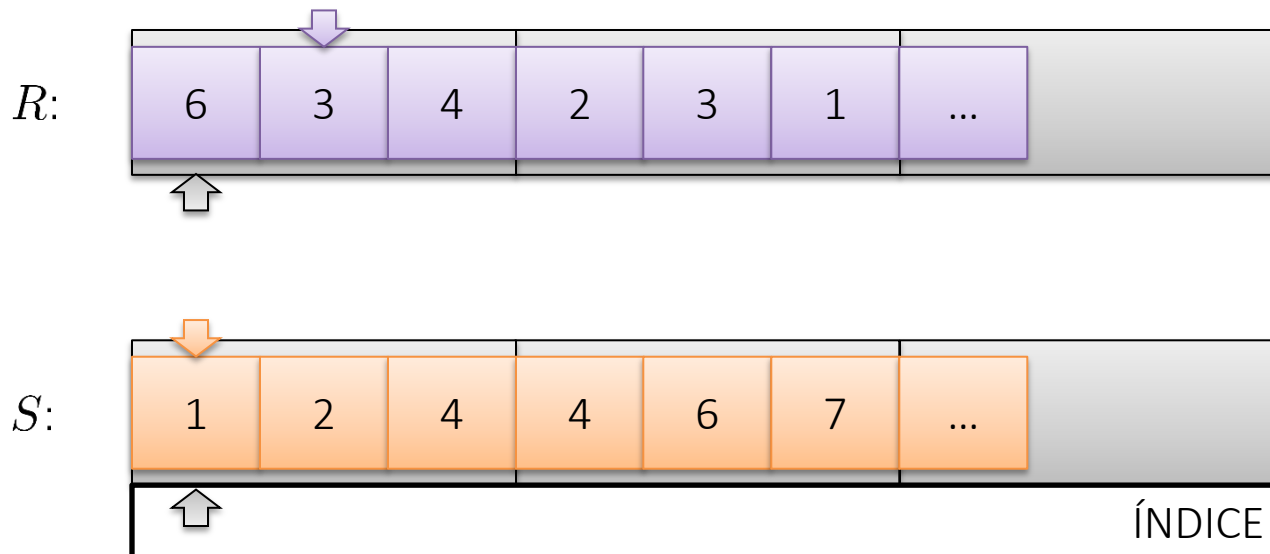
$$R \bowtie S$$

|   |   |
|---|---|
| 6 | 6 |
|---|---|

# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



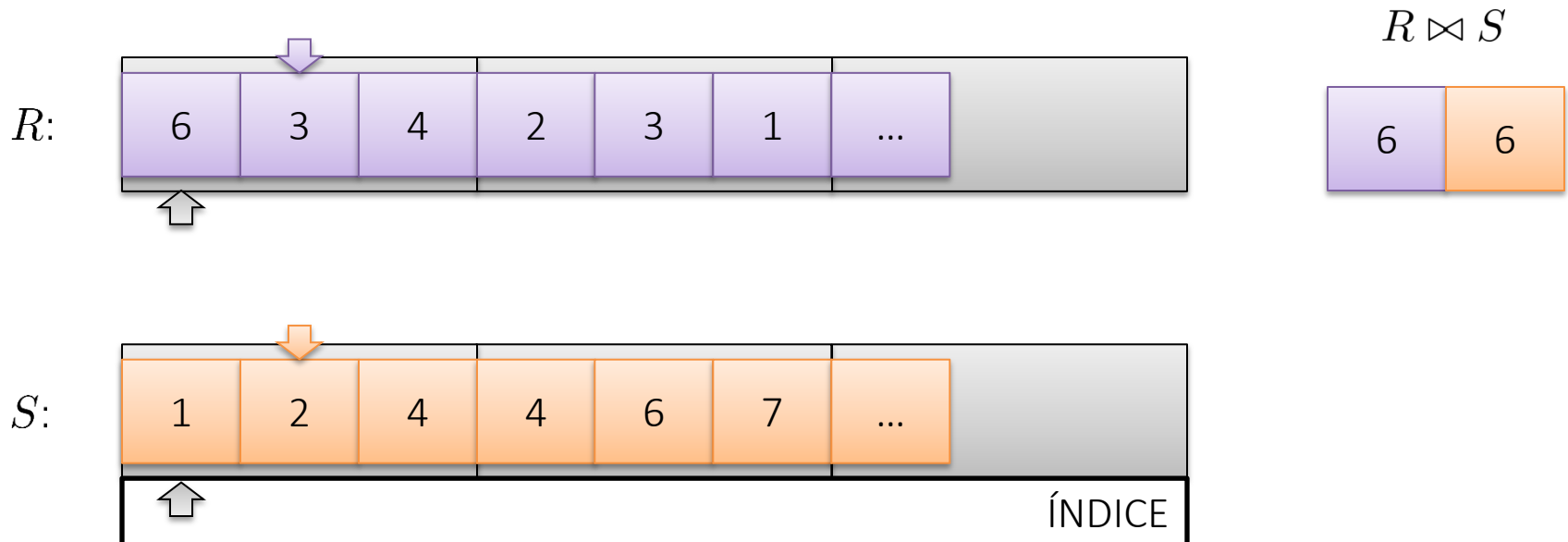
$$R \bowtie S$$

|   |   |
|---|---|
| 6 | 6 |
|---|---|

# Loop anidado (con índices)

$$R \bowtie S$$

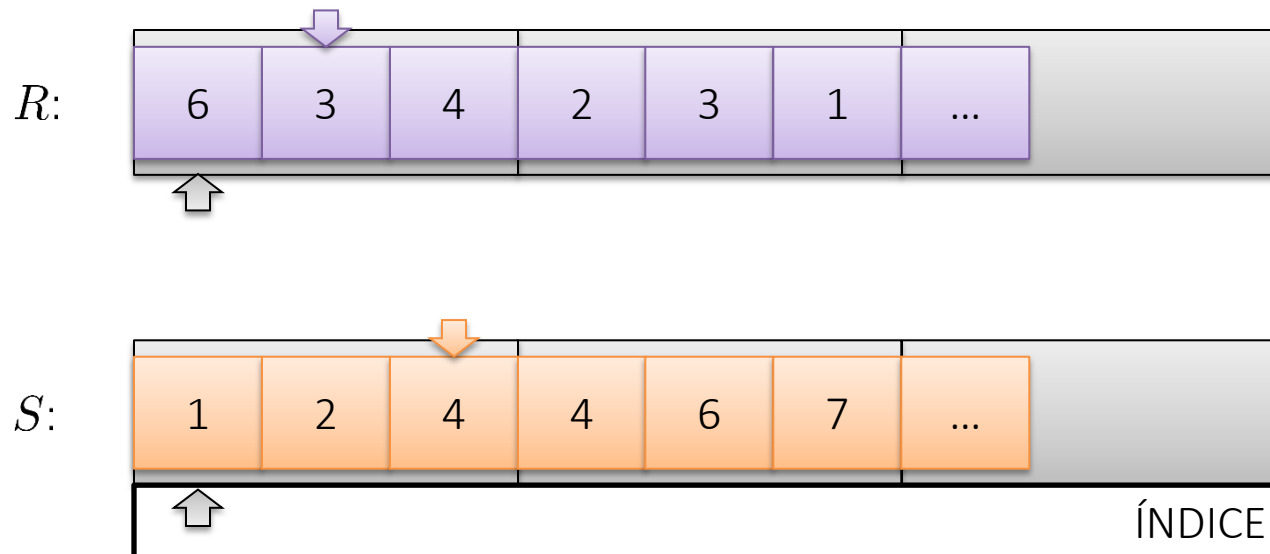
- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



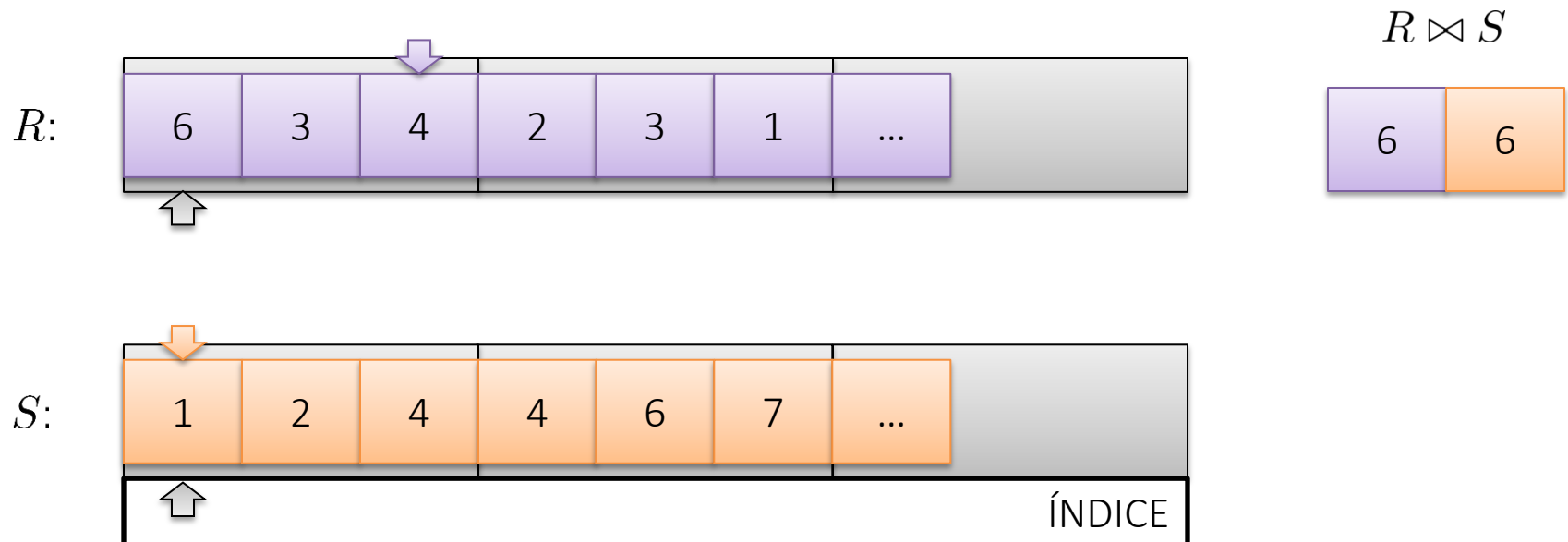
$$R \bowtie S$$

|   |   |
|---|---|
| 6 | 6 |
|---|---|

# Loop anidado (con índices)

$$R \bowtie S$$

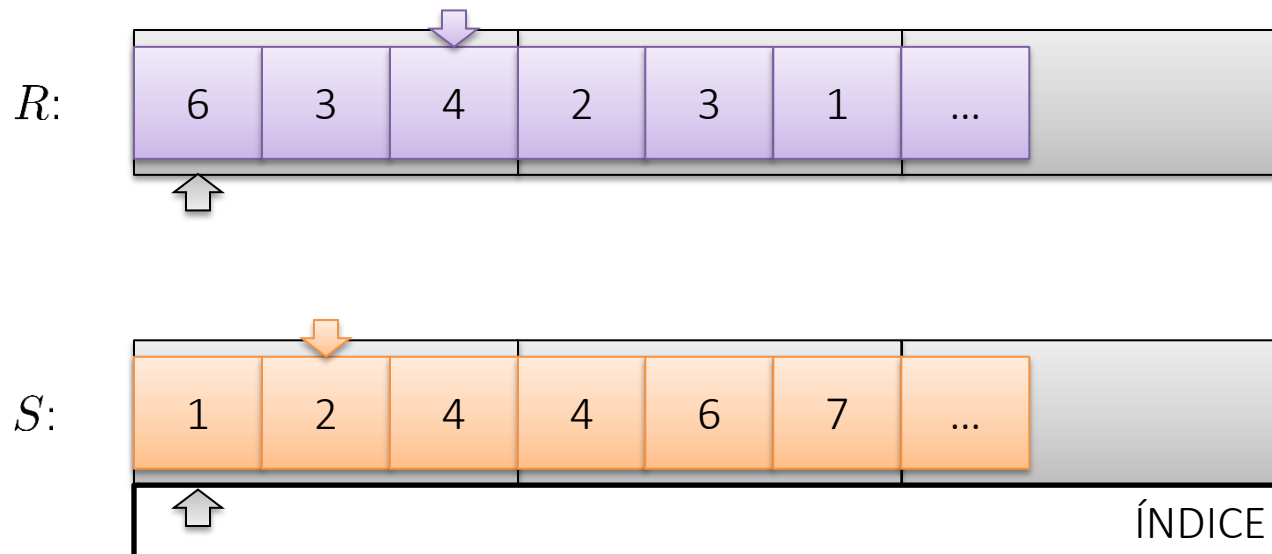
- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



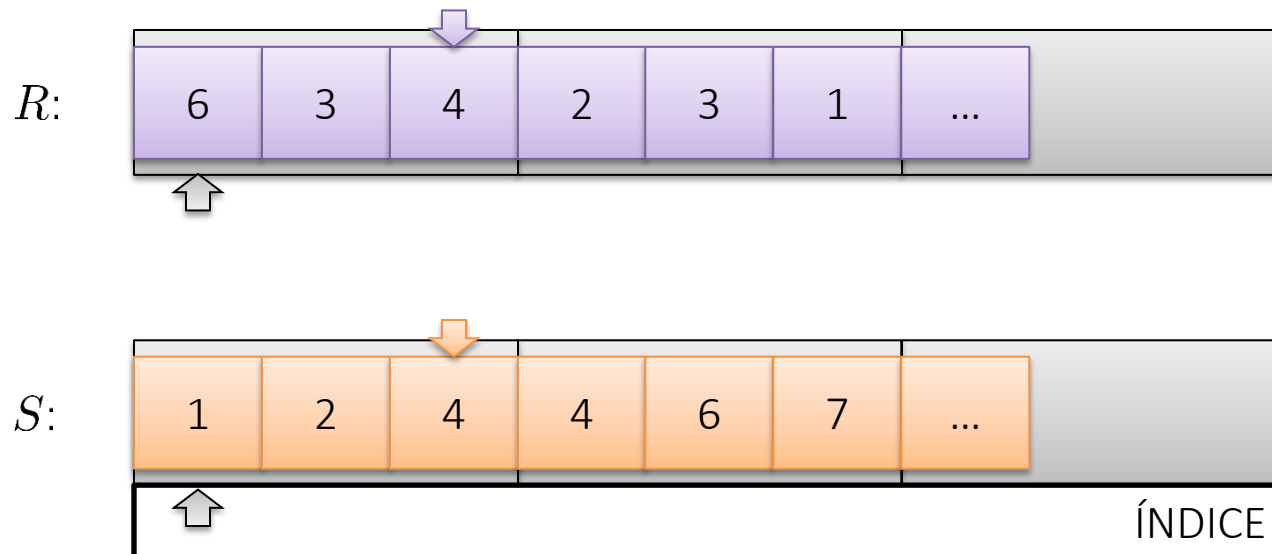
$$R \bowtie S$$



# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



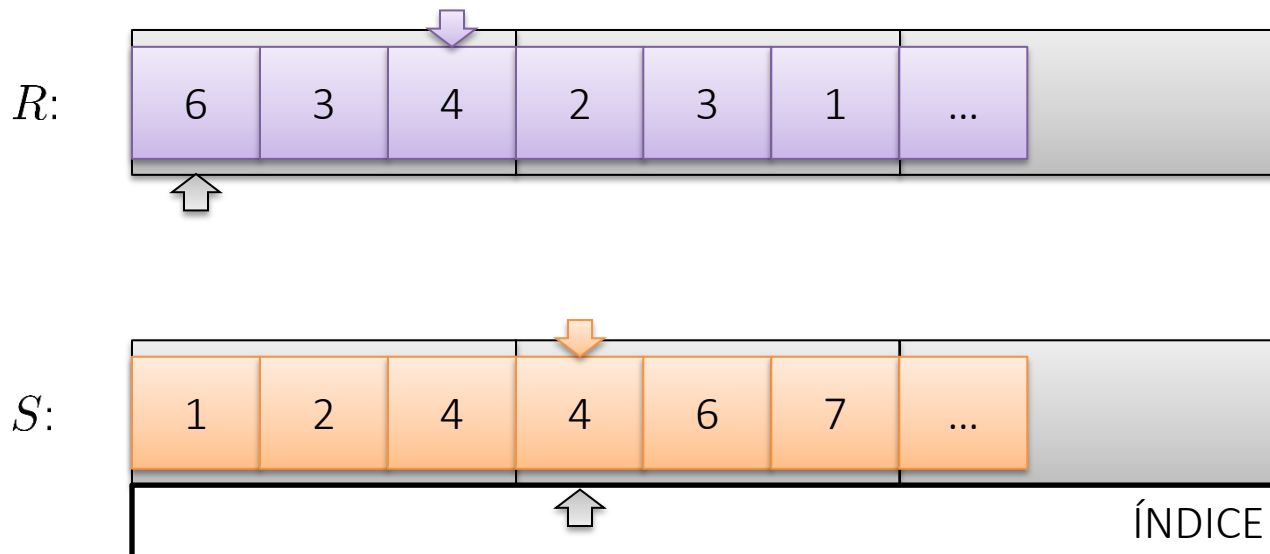
$$R \bowtie S$$

|   |   |
|---|---|
| 6 | 6 |
| 4 | 4 |

# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



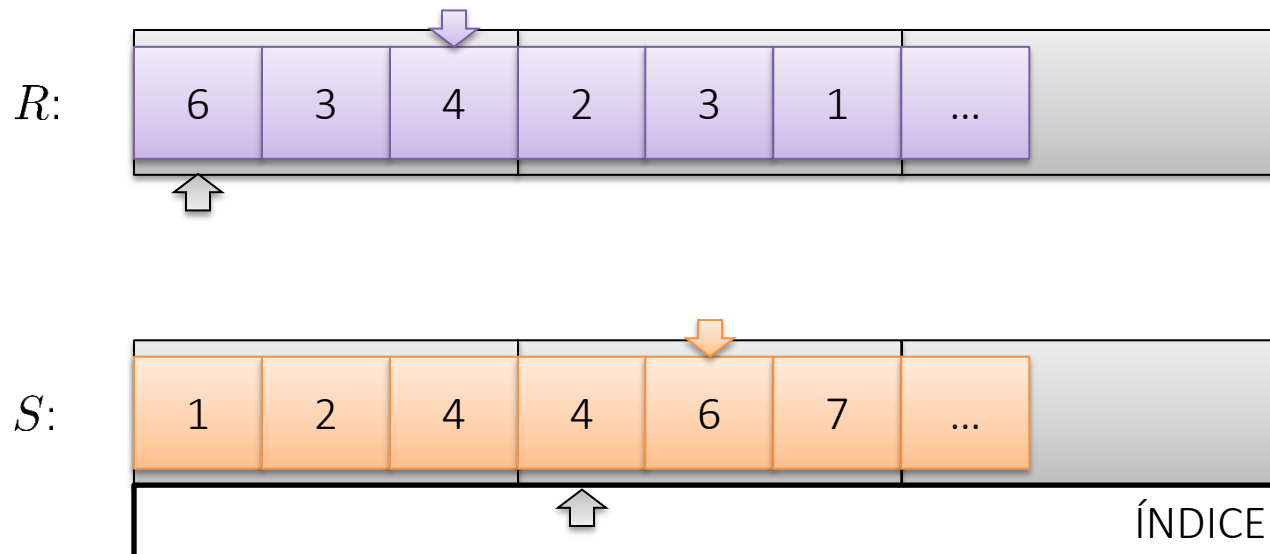
$$R \bowtie S$$

|   |   |
|---|---|
| 6 | 6 |
| 4 | 4 |
| 4 | 4 |

# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



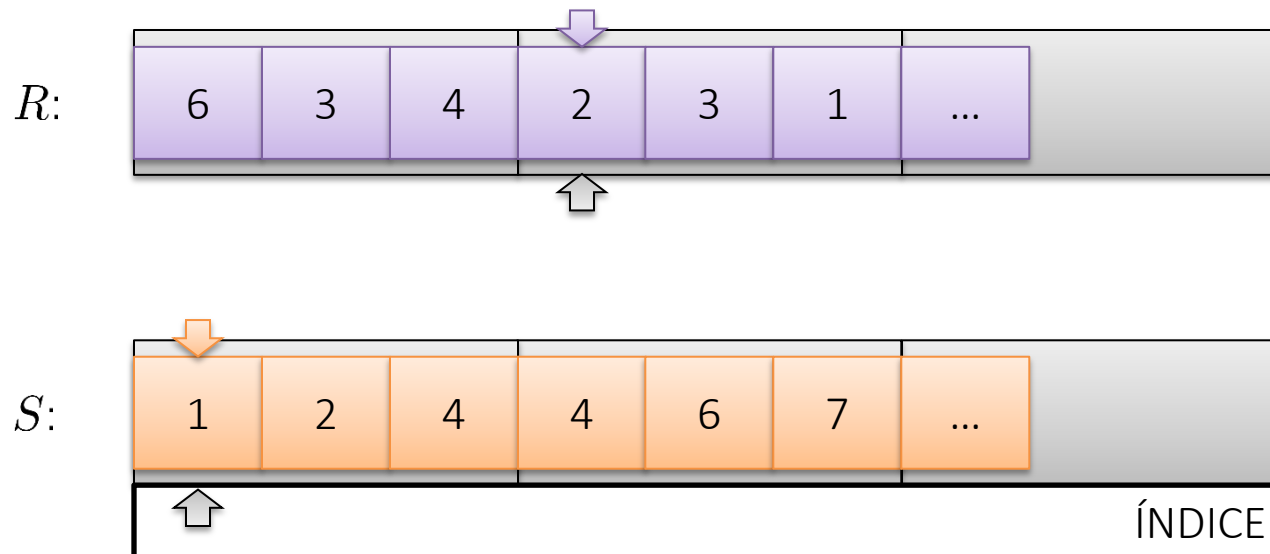
$$R \bowtie S$$

|   |   |
|---|---|
| 6 | 6 |
| 4 | 4 |
| 4 | 4 |

# Loop anidado (con índices)

$R \bowtie S$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



$R \bowtie S$

|   |   |
|---|---|
| 6 | 6 |
| 4 | 4 |
| 4 | 4 |

...

# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$

¿Costo?

$\mathcal{B}(S)$ : costo de buscar en  $S$

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot \mathcal{B}(S)$$

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot \lceil \frac{|S|}{B} \rceil \text{ peor caso}$$

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot O(\log_b(\lceil \frac{|S|}{B} \rceil)) \text{ mejor caso: Árbol } B+ \text{ (disco)}$$

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot O(1) \text{ mejor caso: Hash / Árbol } B+ \text{ (mem. p.)}$$

El “**mejor caso**” aquí asume que, para cada tupla  $r$ , las tuplas  $s_1, \dots, s_k$  que sean compatibles con  $r$  estén en un número constante de bloques.

# Loop anidado (con índices)

$$R \bowtie S$$

- Para cada tupla  $r \in R$ 
  - Buscar  $s \in S$  en el índice tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$

¿Costo?

$\mathcal{B}(S)$ : costo de buscar en  $S$

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot \mathcal{B}(S)$$

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot \lceil \frac{|S|}{B} \rceil \text{ peor caso}$$

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot O(\log_b(\lceil \frac{|S|}{B} \rceil)) \text{ mejor caso: Árbol } B+ \text{ (disco)}$$

$$\lceil \frac{|R|}{B} \rceil + |R| \cdot O(1) \text{ mejor caso: Hash / Árbol } B+ \text{ (mem. p.)}$$

¿Memoria?

$2B$  tuplas

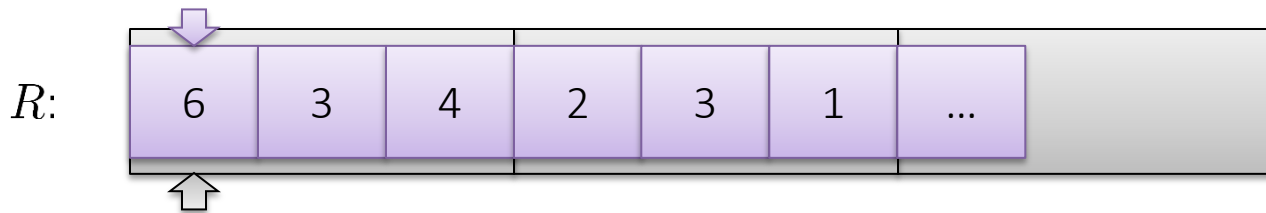
¿Elegir  $R$  y  $S$ ?

$|R| < |S|$  (para ahorrar tiempo)

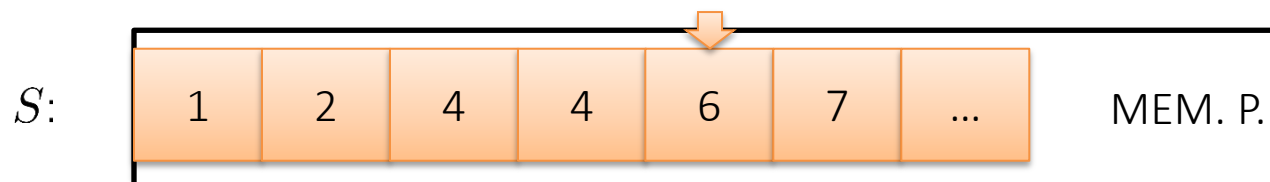
# Hash-join

$$R \bowtie S$$

- Guardar  $S$  en memoria principal
- Para cada tupla  $r \in R$ 
  - Buscar  $s$  en memoria principal tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



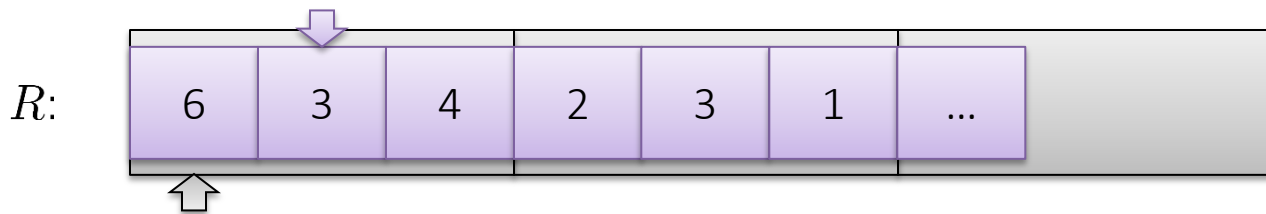
$$R \bowtie S$$



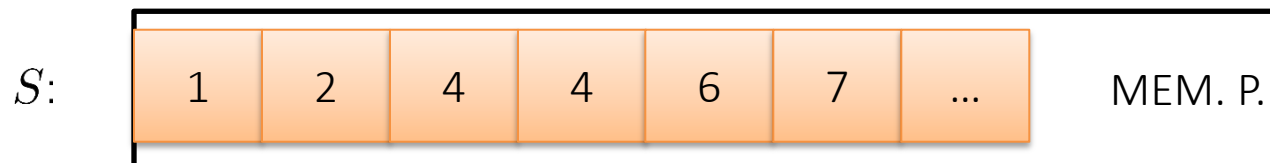
# Hash-join

$$R \bowtie S$$

- Guardar  $S$  en memoria principal
- Para cada tupla  $r \in R$ 
  - Buscar  $s$  en memoria principal tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



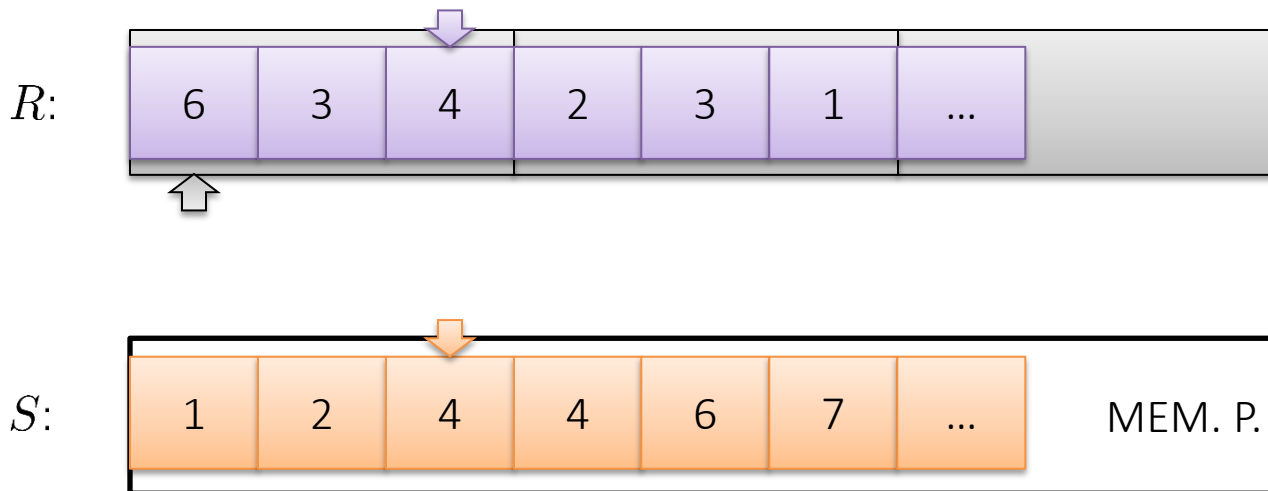
$$R \bowtie S$$



# Hash-join

$$R \bowtie S$$

- Guardar  $S$  en memoria principal
- Para cada tupla  $r \in R$ 
  - Buscar  $s$  en memoria principal tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



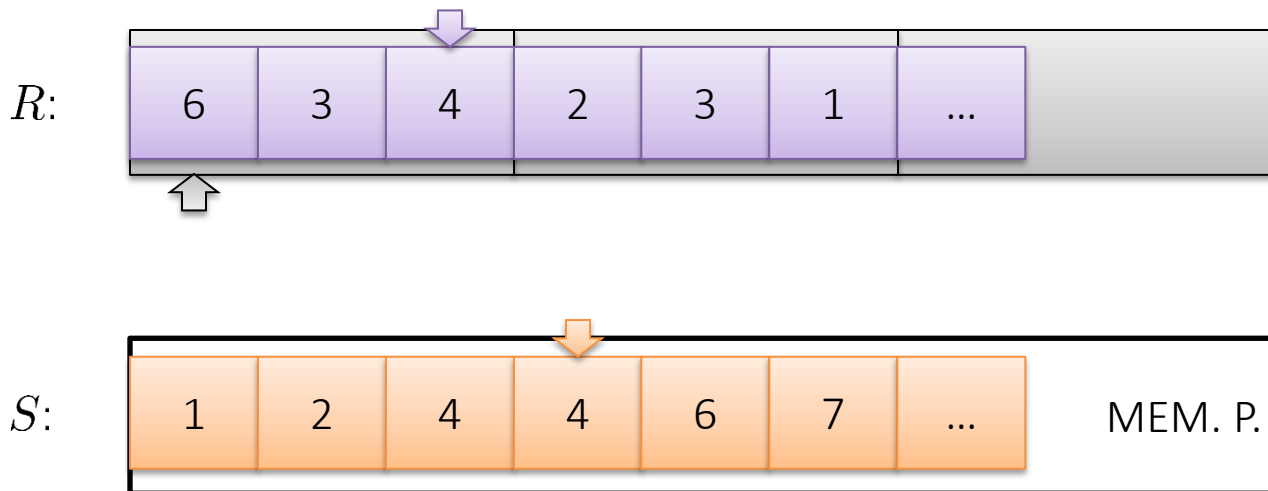
$$R \bowtie S$$

|   |   |
|---|---|
| 6 | 6 |
| 4 | 4 |

# Hash-join

$$R \bowtie S$$

- Guardar  $S$  en memoria principal
- Para cada tupla  $r \in R$ 
  - Buscar  $s$  en memoria principal tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$



$$R \bowtie S$$

|   |   |
|---|---|
| 6 | 6 |
| 4 | 4 |
| 4 | 4 |

...

# Hash-join

$$R \bowtie S$$

- Guardar  $S$  en memoria principal
- Para cada tupla  $r \in R$ 
  - Buscar  $s$  en memoria principal tal que  $r$  y  $s$  satisfagan el join:  
escribir  $\{r\} \times \{s\}$

¿Costo?

$$\lceil \frac{|R|}{B} \rceil + \lceil \frac{|S|}{B} \rceil$$

¿Memoria?

$$|S| + B \text{ tuplas}$$

¿Elegir  $R$  y  $S$ ?

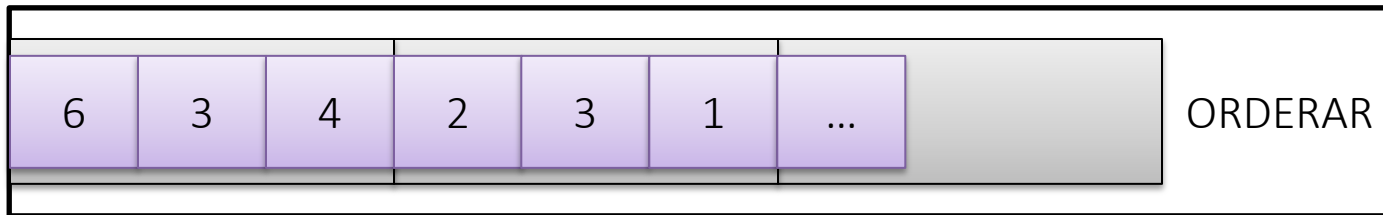
$$|S| < |R| \text{ (para ahorrar memoria)}$$

# Sort-merge-join

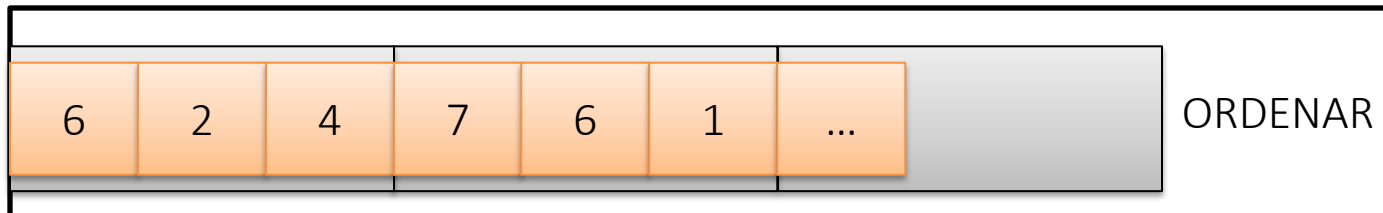
$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$

$R$ :



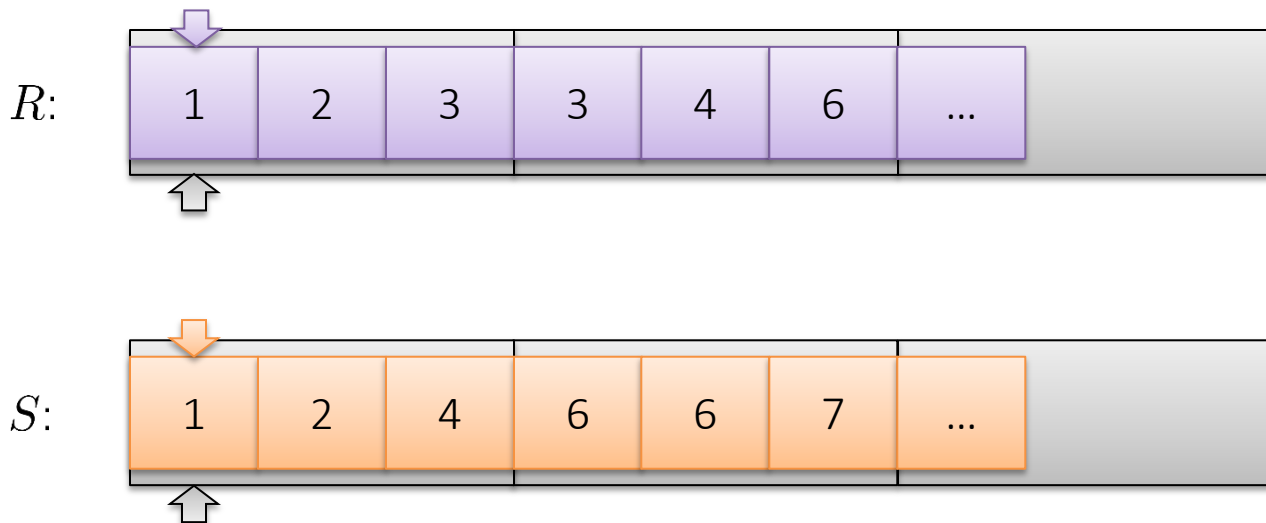
$S$ :



# Sort-merge-join

$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$



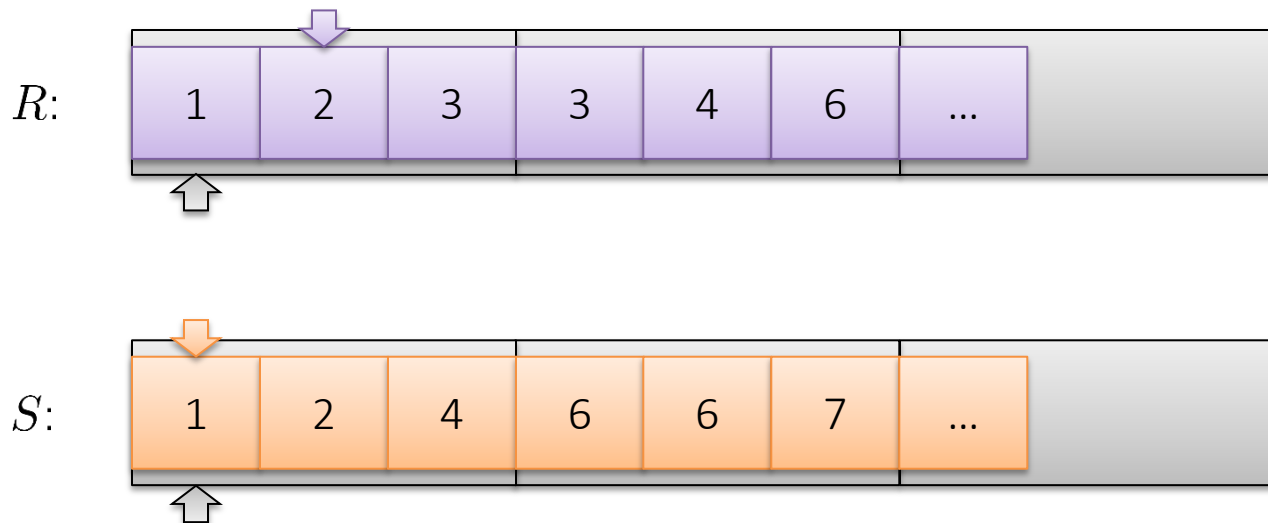
$$R \bowtie S$$



# Sort-merge-join

$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$



$$R \bowtie S$$

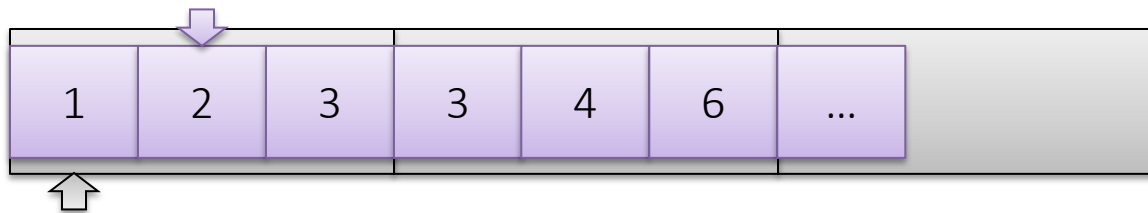
|   |   |
|---|---|
| 1 | 1 |
|---|---|

# Sort-merge-join

$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$

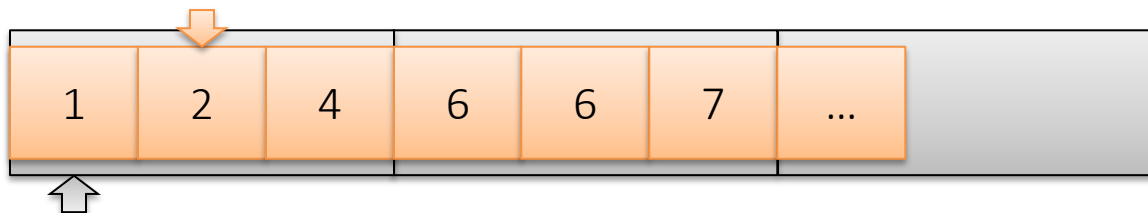
$R$ :



$$R \bowtie S$$

|   |   |
|---|---|
| 1 | 1 |
| 2 | 2 |

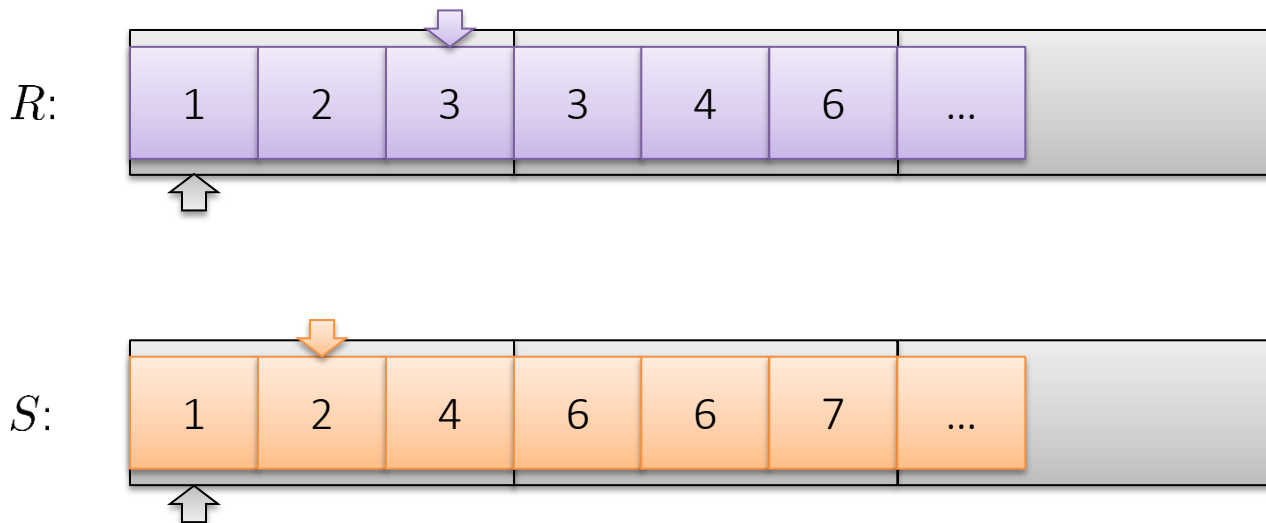
$S$ :



# Sort-merge-join

$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$



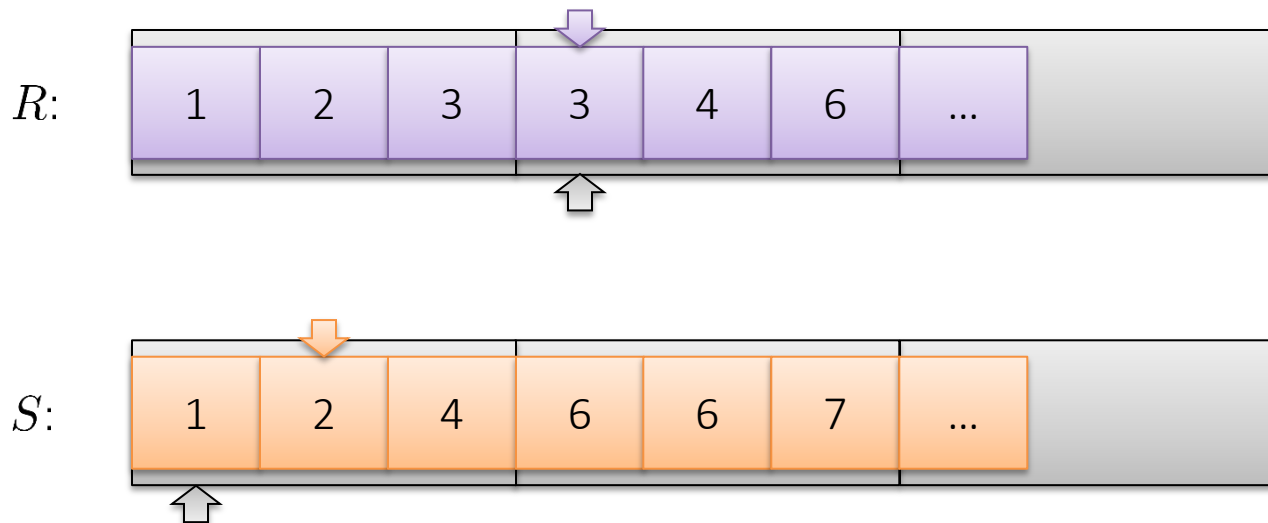
$$R \bowtie S$$

|   |   |
|---|---|
| 1 | 1 |
| 2 | 2 |

# Sort-merge-join

$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$



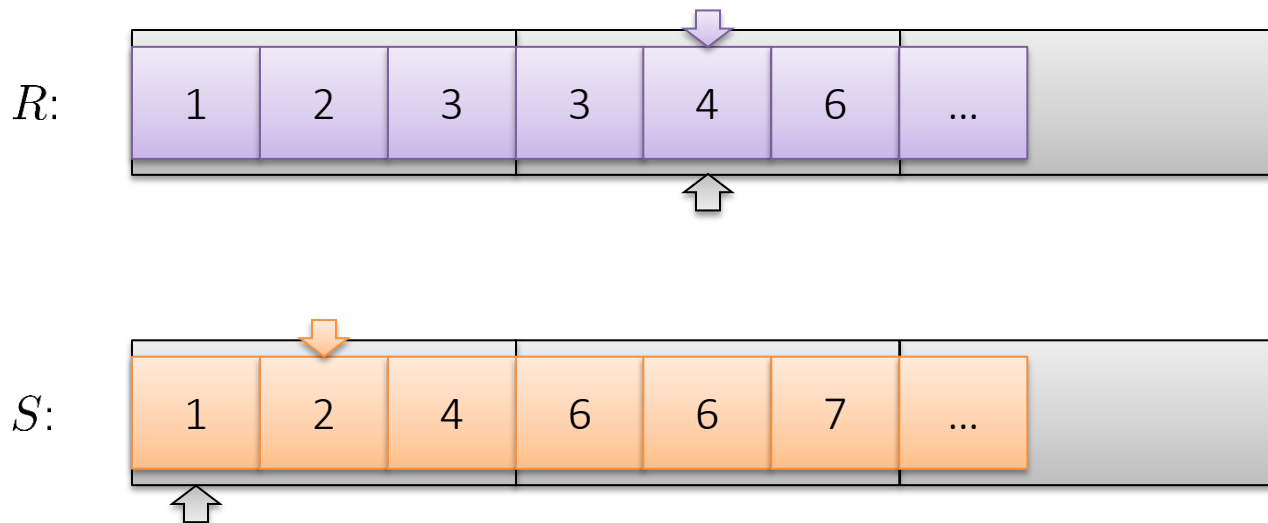
$$R \bowtie S$$

|   |   |
|---|---|
| 1 | 1 |
| 2 | 2 |

# Sort-merge-join

$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$



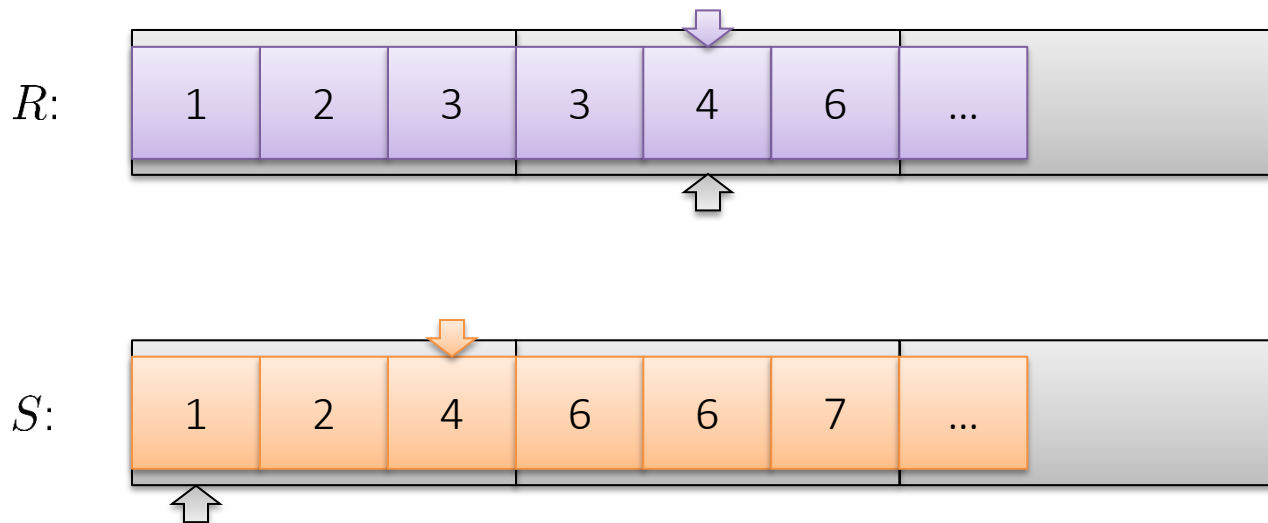
$$R \bowtie S$$

|   |   |
|---|---|
| 1 | 1 |
| 2 | 2 |

# Sort-merge-join

$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$



$$R \bowtie S$$

|   |   |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 4 | 4 |

...

# Sort-merge-join

$$R \bowtie S$$

- Ordenar  $R$  y  $S$  por los atributos del join
- Aplicar un merge-sort y para cada tupla  $r$  y  $s$  que satisfagan el join: escribir  $\{r\} \times \{s\}$

¿Costo?

$$\mathcal{O} + \lceil \frac{|R|}{B} \rceil + \lceil \frac{|S|}{B} \rceil$$

$\mathcal{O}$ : costo de ordenamiento

¿Memoria?

$2B$  tuplas (una vez que estén ordenadas)

¿Elegir  $R$  y  $S$ ?

No importa

Puede ser que las relaciones ya estén ordenadas por los atributos del join, en cual caso ¡es una buena opción!

# Joins: Comparación

## ¿Cuál es mejor?

1. Loop anidado (sin índice)
2. Loop anidado (con índice)
3. Hash-join
4. Sort-merge-join

- Loop anidado (sin índice):
  - Nunca es bueno (pero a veces es lo mejor)
- Loop anidado (con índice):
  - Cuando el índice esté disponible y:
    - Pocas tuplas en  $R$  y pocas tuplas en  $S$  satisfagan el join
- *Hash-join*
  - Cuando  $S$  quepa en memoria y:
    - Muchas tuplas en  $R$  satisfagan el join
- *Sort-merge-join*
  - Cuando  $R$  y  $S$  ya estén ordenadas por los atributos del join y:
    - Muchas tuplas en  $R$  y  $S$  satisfagan el join

# Ver la planificación: EXPLAIN/ANALYZE

`EXPLAIN SELECT ...` *-- mostrar la planificación sin ejecutar la consulta*

`ANALYZE SELECT ...` *-- ejecutar la consulta y mostrar analisis*

`EXPLAIN ANALYZE SELECT ...` *-- combinar ambos*

```
EXPLAIN ANALYZE SELECT * FROM lab7.personaje100
WHERE p_nombre='Whiplash' AND p_anho=2014;
```

```
Seq Scan on personaje100
  (cost=0.00..53967.89 rows=2 width=47)
  (actual time=6.683..402.203 rows=54 loops=1)
    Filter: ((p_nombre)::text = 'Whiplash'::text AND (p_anho = 2014))
    Rows Removed by Filter: 2170472
Planning time: 0.111 ms
Execution time: 402.273 ms
```

# Ver la planificación: EXPLAIN/ANALYZE

`EXPLAIN SELECT ...` -- *mostrar la planificación sin ejecutar la consulta*

`ANALYZE SELECT ...` -- *ejecutar la consulta y mostrar analisis*

`EXPLAIN ANALYZE SELECT ...` -- *combinar ambos*

```
EXPLAIN ANALYZE SELECT * FROM lab7i.personaje100
  WHERE p_nombre='Whiplash' AND p_anho=2014;
```

```
Index Scan using personaje100_pnombreanho on personaje100
  (cost=0.43..12.47 rows=2 width=47)
  (actual time=0.123..0.540 rows=54 loops=1)
    Index Cond: (((p_nombre)::text = 'Whiplash'::text) AND (p_anho = 2014))
Planning time: 0.143 ms
Execution time: 0.589 ms
```

# INDEXACIÓN Y OPTIMIZACIÓN

# SQL es un lenguaje **declarativo**

Uno dice lo que quiere, no cómo debería ser computado

(1) Selección/producto:

*Idealmente*, el motor puede elegir el mejor plan de ejecución independientemente de la expresión particular de la consulta

(2) Join explícito:

El usuario no tiene que preocuparse por la optimización de la consulta

(3) Consulta anidada (FROM):

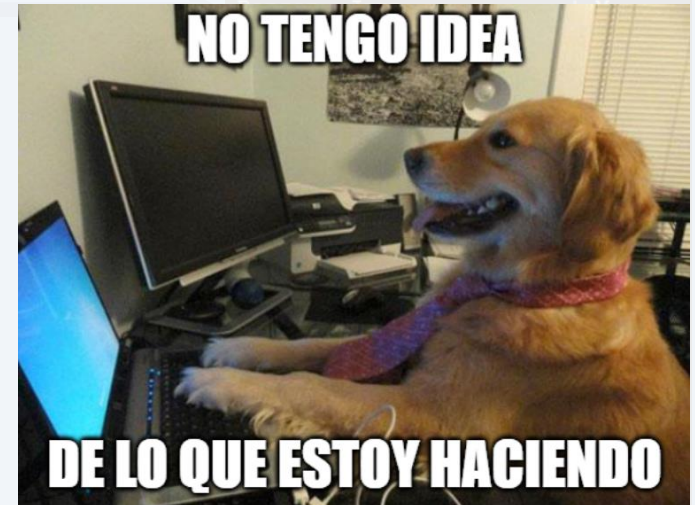
```
SELECT DISTINCT A.nombre, A.genero FROM
( SELECT DISTINCT P2.a_nombre FROM
  ( SELECT DISTINCT P1.p_nombre, P1.p_anho
    FROM personaje P1
    WHERE P1.a_nombre='Tyler, Liv'
  ) PLT, personaje P2
  WHERE PLT.p_nombre = P2.p_nombre
  AND PLT.p_anho = P2.p_anho
) CLT, actor A
WHERE CLT.a_nombre = A.nombre
```

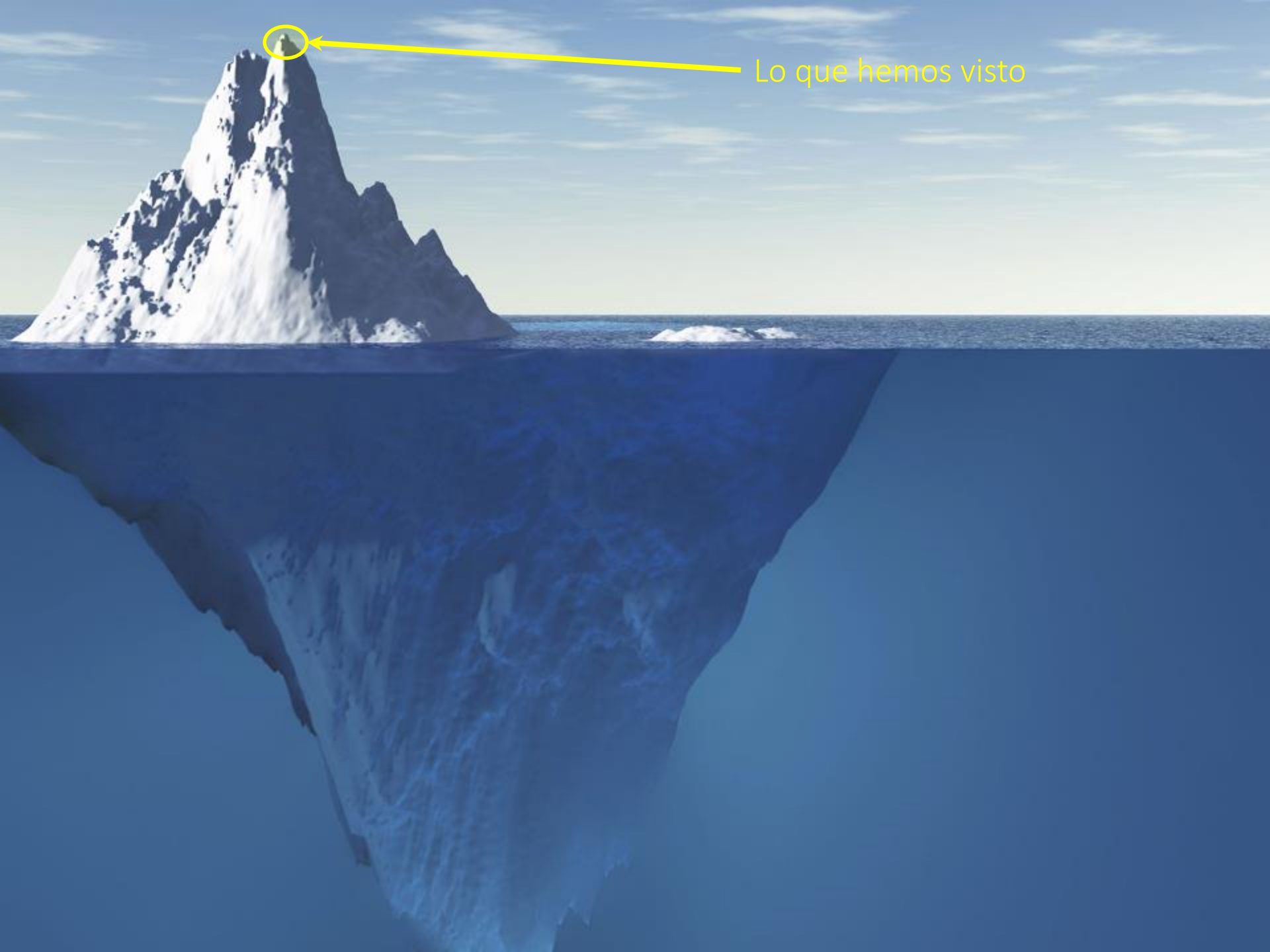
(4) Consulta anidada (WHERE/IN):

```
SELECT DISTINCT A.nombre, A.genero
FROM actor A
WHERE A.nombre IN
( SELECT DISTINCT P2.a_nombre
  FROM personaje P2
  WHERE (P2.p_nombre,P2.p_anho) IN
    ( SELECT DISTINCT P1.p_nombre, P1.p_anho
      FROM personaje P1
      WHERE P1.a_nombre='Tyler, Liv'
    )
  )
```

Caja Negra

| nombre             | genero |
|--------------------|--------|
| Acevedo, Gina (I)  | F      |
| Acevedo, Gina (I)  | M      |
| Acevedo, Gina (I)  | M      |
| Acevedo, Gina (I)  | M      |
| Appleton, Matt (I) | M      |
| Astin, Ali         | F      |
| Astin, Sean        | M      |
| Aston, David (I)   | M      |
| Bach, John (I)     | M      |
| Bach, John (I)     | M      |
| Bach, John (I)     | M      |
| Bach, John (I)     | M      |
| Benzon, Jarl       | M      |





Lo que hemos visto

# ¿Preguntas?

