

# Algorithmes de Jointure

**JOIN( F1 , F2 , condition )**

*Toutes les concaténations des enregistrements de F1 avec tous les enregistrements de F2 pour lesquelles la condition donnée est vérifiée*

## **3 approches :**

Jointure par Boucles Imbriquées

Jointure par Tri-Fusion

Jointure par Hachage

## **Objectif :**

Minimiser le nombre d'opérations d'E/S

# Algorithme de Jointure par Boucles Imbriquées

( Nested-Loop Join Algorithm )

Joindre sur une condition **C** quelconque 2 fichiers **F1** et **F2** formés resp. de **N1** et **N2** blocs en utilisant **M** buffers en **MC**

## Principe :

Chaque enregistrement de F1  
doit être mis en correspondance  
avec tous les enregistrements de F2  
pour vérifier la condition **C**

## Fichier F1

a01  
a02  
a03

a04  
a05  
a06

a07  
a08  
a09

a10  
a11  
a12

## Fichier F2

b01  
b02  
b03

b04  
b05  
b06

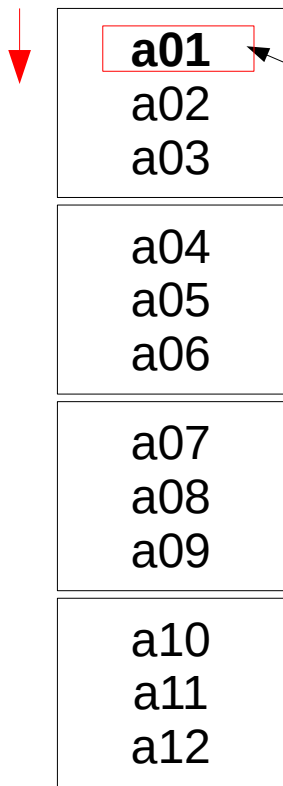
b07  
b08  
b09

b10  
b11  
b12

## Fichier Résultat

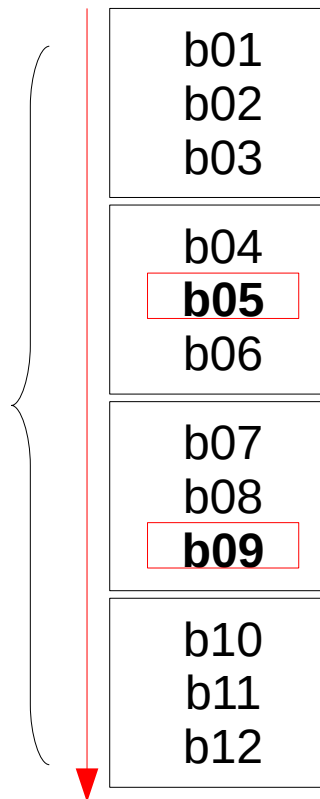


Fichier F1

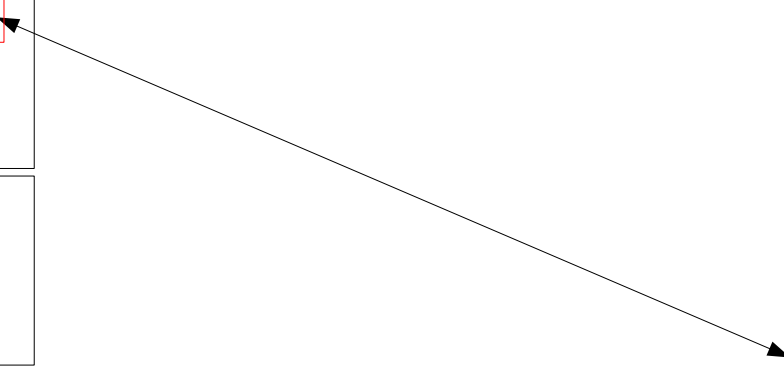


<b>a01</b> a02 a03
a04 a05 a06
a07 a08 a09
a10 a11 a12

Fichier F2



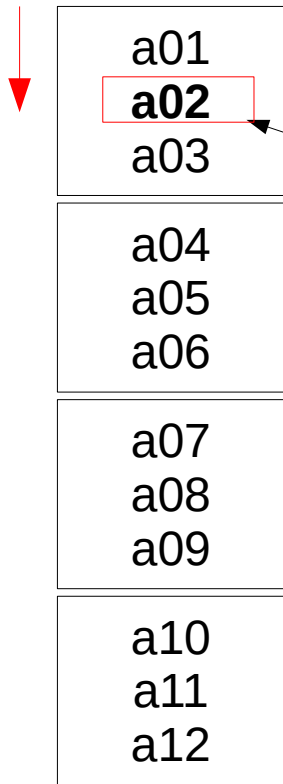
b01 b02 b03
b04 <b>b05</b> b06
b07 b08 <b>b09</b>
b10 b11 b12



Fichier Résultat

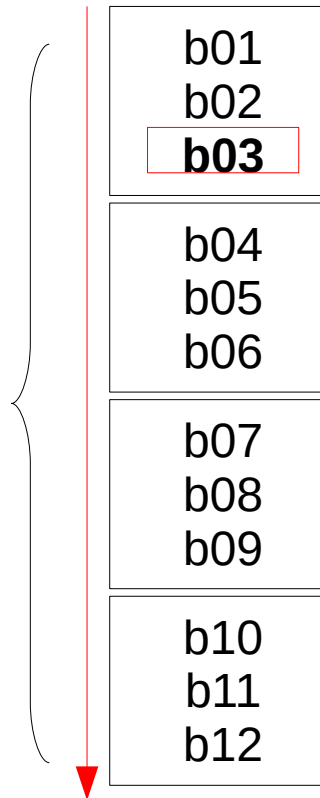
<b>a01,b05</b> <b>a01,b09</b>
----------------------------------

## Fichier F1

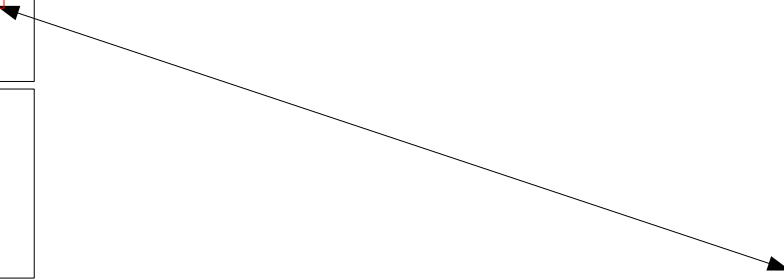


a01 <b>a02</b> a03
a04 a05 a06
a07 a08 a09
a10 a11 a12

## Fichier F2



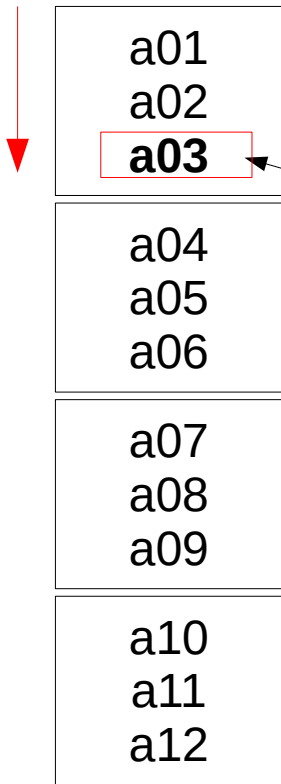
b01 b02 <b>b03</b>
b04 b05 b06
b07 b08 b09
b10 b11 b12



## Fichier Résultat

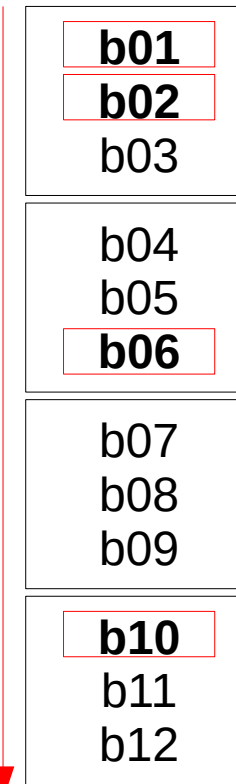
a01,b05
a01,b09
<b>a02,b03</b>

## Fichier F1

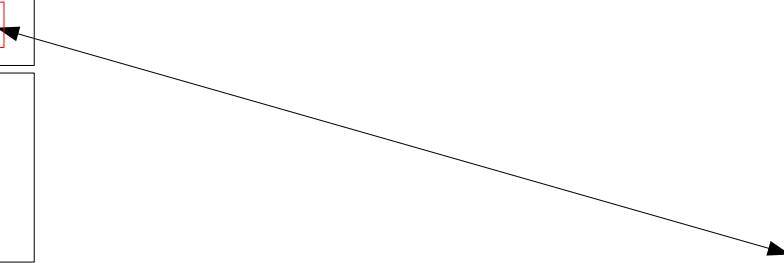


a01 a02 <b>a03</b>
a04 a05 a06
a07 a08 a09
a10 a11 a12

## Fichier F2



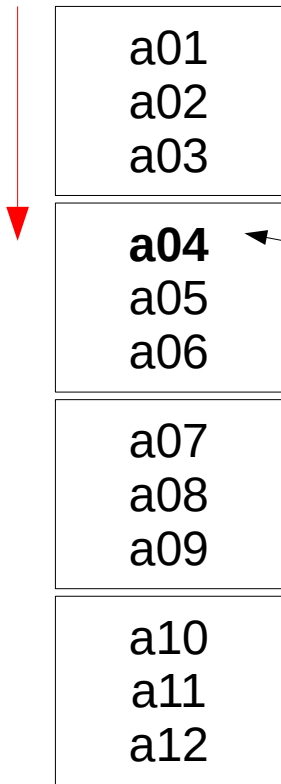
<b>b01</b> <b>b02</b> b03
b04 b05 <b>b06</b>
b07 b08 b09
<b>b10</b> b11 b12



## Fichier Résultat

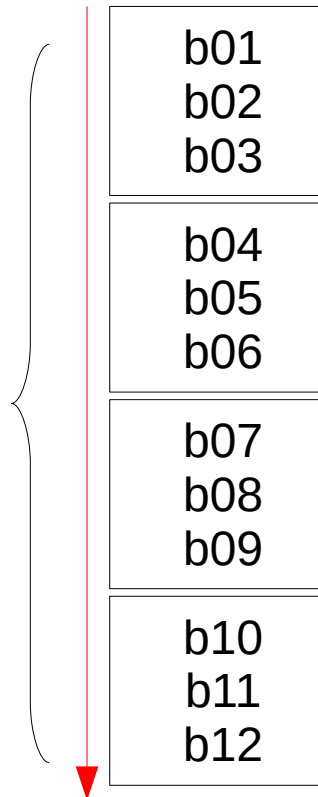
a01,b05 a01,b09 a02,b03	<b>a03,b01</b> <b>a03,b02</b> <b>a03,b06</b>	<b>a03,b10</b>
-------------------------------	--	----------------

## Fichier F1



a01 a02 a03
<b>a04</b> a05 a06
a07 a08 a09
a10 a11 a12

## Fichier F2

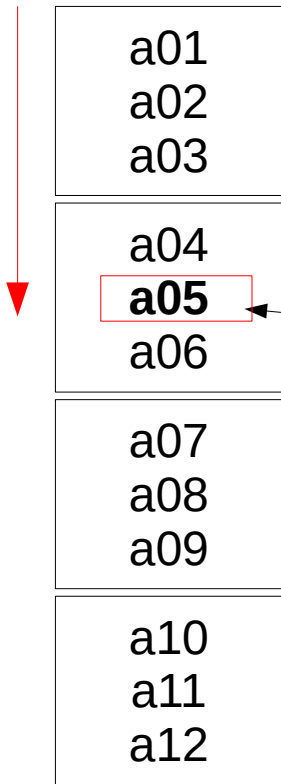


b01 b02 b03
b04 b05 b06
b07 b08 b09
b10 b11 b12

## Fichier Résultat

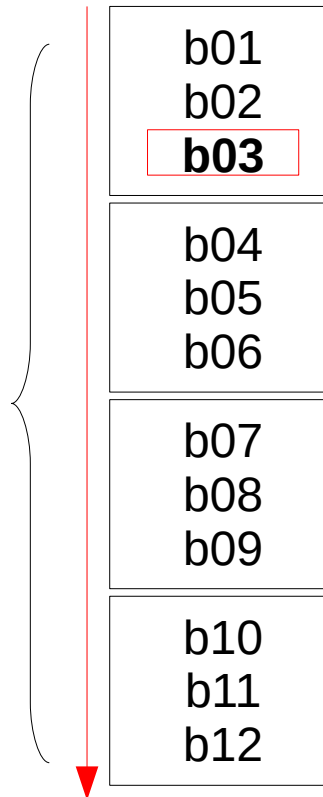
a01,b05 a01,b09 a02,b03	a03,b01 a03,b02 a03,b06	a03,b10
-------------------------------	-------------------------------	---------

## Fichier F1



a01
a02
a03
a04
<b>a05</b>
a06
a07
a08
a09
a10
a11
a12

## Fichier F2



b01
b02
<b>b03</b>
b04
b05
b06
b07
b08
b09
b10
b11
b12



## Fichier Résultat

a01,b05	a03,b01	a03,b10
a01,b09	a03,b02	<b>a05,b03</b>
a02,b03	a03,b06	



## Fichier F1

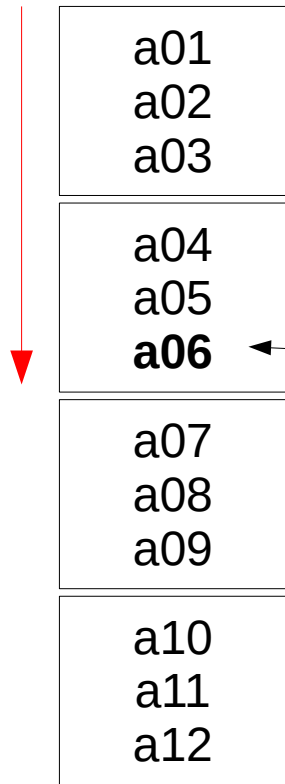


Diagram of Fichier F1: A vertical stack of four boxes. The first box contains 'a01', 'a02', 'a03'. The second box contains 'a04', 'a05', and 'a06' (bolded). The third box contains 'a07', 'a08', 'a09'. The fourth box contains 'a10', 'a11', 'a12'. A red arrow points down the left side. A black arrow points from the right to the bolded 'a06'.

a01 a02 a03
a04 a05 <b>a06</b>
a07 a08 a09
a10 a11 a12

## Fichier F2

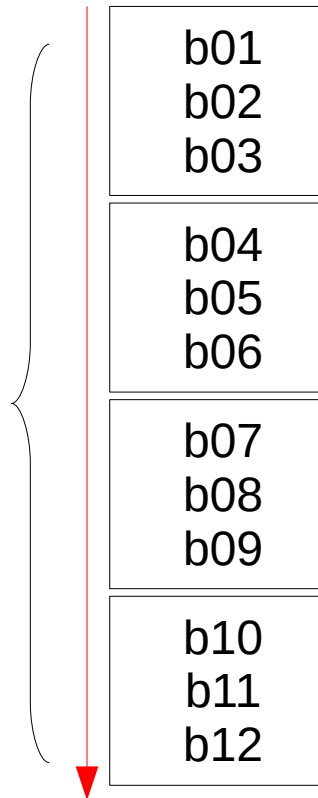


Diagram of Fichier F2: A vertical stack of four boxes. The first box contains 'b01', 'b02', 'b03'. The second box contains 'b04', 'b05', 'b06'. The third box contains 'b07', 'b08', 'b09'. The fourth box contains 'b10', 'b11', 'b12'. A red arrow points down the left side. A bracket is on the left side, spanning the first three boxes.

b01 b02 b03
b04 b05 b06
b07 b08 b09
b10 b11 b12

## Fichier Résultat

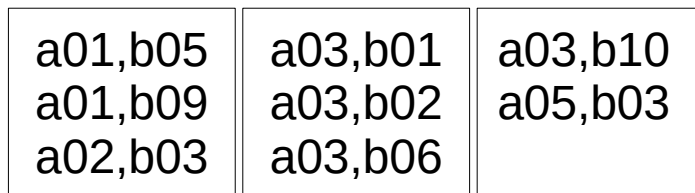
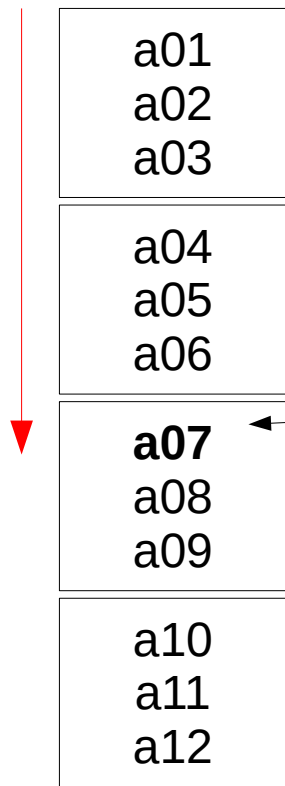


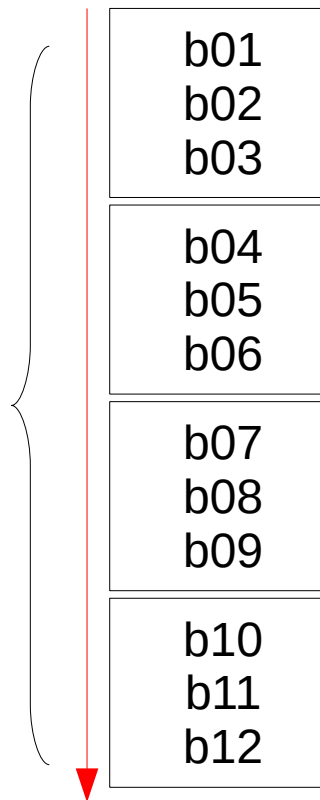
Diagram of Fichier Résultat: Three boxes containing joined data. The first box contains 'a01,b05', 'a01,b09', 'a02,b03'. The second box contains 'a03,b01', 'a03,b02', 'a03,b06'. The third box contains 'a03,b10', 'a05,b03'.

a01,b05 a01,b09 a02,b03	a03,b01 a03,b02 a03,b06	a03,b10 a05,b03
-------------------------------	-------------------------------	--------------------

## Fichier F1



## Fichier F2



## Fichier Résultat

a01,b05 a01,b09 a02,b03	a03,b01 a03,b02 a03,b06	a03,b10 a05,b03
-------------------------------	-------------------------------	--------------------

## Fichier F1

a01
a02
a03
a04
a05
a06
a07
<b>a08</b>
a09
a10
a11
a12

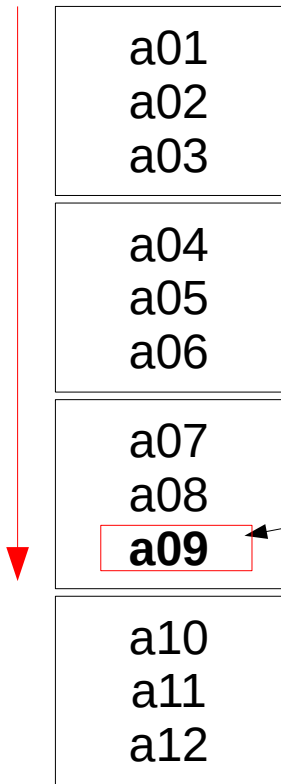
## Fichier F2

<b>b01</b>
b02
b03
b04
<b>b05</b>
b06
b07
b08
b09
b10
b11
b12

## Fichier Résultat

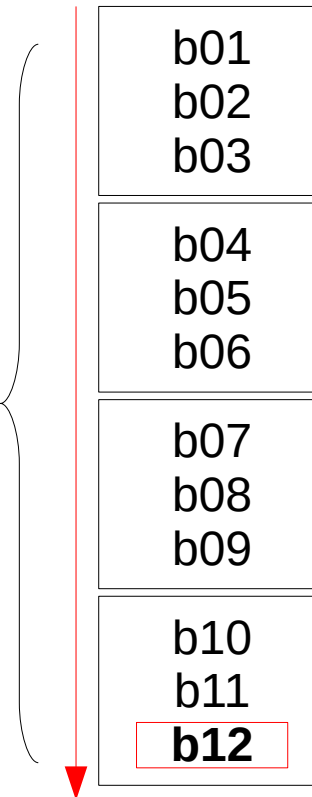
a01,b05	a03,b01	a03,b10	<b>a08,b05</b>
a01,b09	a03,b02	a05,b03	
a02,b03	a03,b06	<b>a08,b01</b>	

## Fichier F1



a01 a02 a03
a04 a05 a06
a07 a08 <b>a09</b>
a10 a11 a12

## Fichier F2



b01 b02 b03
b04 b05 b06
b07 b08 b09
b10 b11 <b>b12</b>

## Fichier Résultat

a01,b05 a01,b09 a02,b03	a03,b01 a03,b02 a03,b06	a03,b10 a05,b03 a08,b01	a08,b05 <b>a09,b12</b>
-------------------------------	-------------------------------	-------------------------------	---------------------------

## Fichier F1

a01 a02 a03
a04 a05 a06
a07 a08 a09
<b>a10</b> a11 a12

## Fichier F2

b01 b02 b03
b04 b05 b06
b07 b08 b09
b10 <b>b11</b> b12

## Fichier Résultat

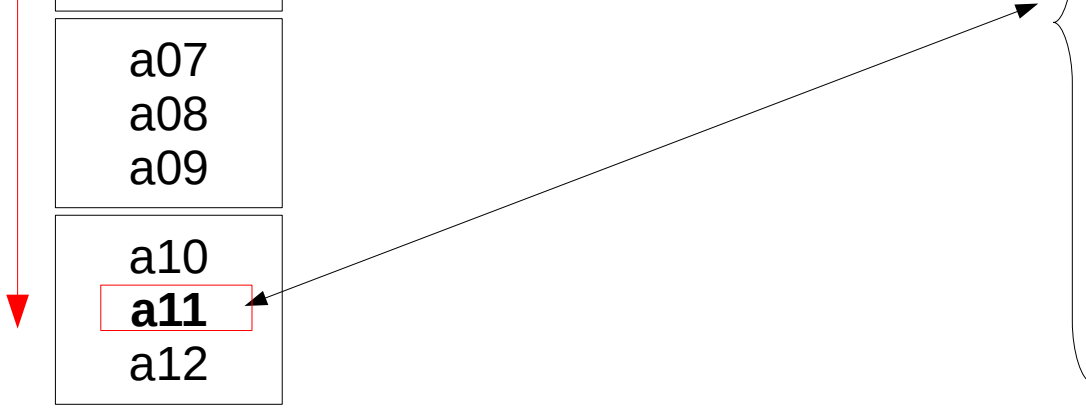
a01,b05 a01,b09 a02,b03	a03,b01 a03,b02 a03,b06	a03,b10 a05,b03 a08,b01	a08,b05 a09,b12 <b>a10,b11</b>
-------------------------------	-------------------------------	-------------------------------	--------------------------------------

## Fichier F1

a01
a02
a03
a04
a05
a06
a07
a08
a09
a10
<b>a11</b>
a12

## Fichier F2

b01
<b>b02</b>
<b>b03</b>
b04
b05
b06
b07
b08
b09
b10
b11
b12



## Fichier Résultat

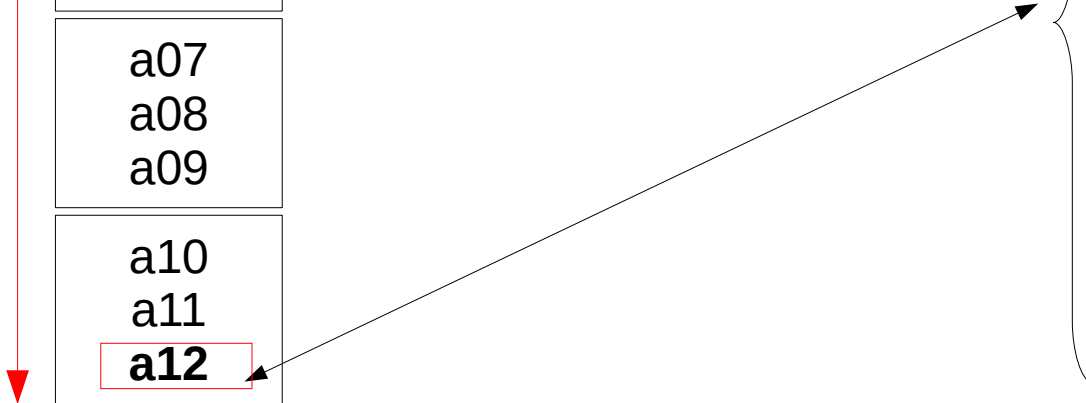
a01,b05	a03,b01	a03,b10	a08,b05	<b>a11,b02</b>
a01,b09	a03,b02	a05,b03	a09,b12	<b>a11,b03</b>
a02,b03	a03,b06	a08,b01	a10,b11	

## Fichier F1

a01
a02
a03
a04
a05
a06
a07
a08
a09
a10
a11
<b>a12</b>

## Fichier F2

b01
b02
<b>b03</b>
<b>b04</b>
b05
b06
b07
b08
<b>b09</b>
b10
<b>b11</b>
<b>b12</b>



## Fichier Résultat

a01,b05	a03,b01	a03,b10	a08,b05	a11,b02	<b>a12,b04</b>	<b>a12,b12</b>
a01,b09	a03,b02	a05,b03	a09,b12	a11,b03	<b>a12,b09</b>	
a02,b03	a03,b06	a08,b01	a10,b11	<b>a12,b03</b>	<b>a12,b11</b>	

## Algorithme : Boucles\_Imbriquées (version naïve)

$i_3 \leftarrow 1 ; j_3 \leftarrow 1$

**Pour**  $i_1 = 1 \dots N1$

*// Boucle Externe : Parcours de F1 ...*

*LireDir( F1,  $i_1$ , buf1 )*

**Pour**  $j_1 = 1 \dots \text{buf1.NB}$

**Pour**  $i_2 = 1 \dots N2$

*// Boucle Interne : Parcours de F2 ...*

*LireDir( F2,  $i_2$ , buf2 )*

**Pour**  $j_2 = 1 \dots \text{buf2.NB}$

$e1 \leftarrow \text{buf1.Tab}[j_1]$

$e2 \leftarrow \text{buf2.Tab}[j_2]$

**SI** (  $C(e1, e2)$  ) *// C : condition de jointure*

$\text{buf3.Tab}[j_3] \leftarrow \langle e1 : e2 \rangle ; j_3++$

**SI** (  $j_3 > b'$  )

$\text{buf3.NB} = b ; \text{EcrireDir}( F3, i_3, \text{buf3} ) ; j_3 \leftarrow 1 ; i_3++$

**FSI**

**FSI**

**FP** *//  $j_2$*

**FP** *//  $i_2$*

*--- Fin de la Boucle Interne (F2) ---*

**FP** *//  $j_1$*

**FP** *//  $i_1$*

*--- Fin de la Boucle Externe (F1) ---*



## Algorithme : Boucles\_Imbriquées (version naïve)

$i_3 \leftarrow 1 ; j_3 \leftarrow 1$

**Pour**  $i_1 = 1 \dots N1$  *// Boucle Externe : Parcours de F1 ...*

LireDir( F1,  $i_1$ , buf1 )

**Pour**  $j_1 = 1 \dots \text{buf1.NB}$

**Pour**  $i_2 = 1 \dots N2$  *// Boucle Interne : Parcours de F2 ...*

LireDir( F2,  $i_2$ , buf2 )

**Pour**  $j_2 = 1 \dots \text{buf2.NB}$

... SI (  $C(e1, e2)$  ) ... EcrireDir(F3,  $i_3$ , buf3) ... FSI ...

**FP** //  $j_2$

**FP** //  $i_2$  *--- Fin de la Boucle Interne (F2) ---*

**FP** //  $j_1$

**FP** //  $i_1$  *--- Fin de la Boucle Externe (F1) ---*

F1 est parcouru **une seule fois** ( $N1$  lectures)

F2 est parcouru **autant de fois que d'enregistrements dans F1** ( $b_1 * N1 * N2$  lectures)

**Coût total (en lectures) :  $N1 + b_1 * N1 * N2$**

Le coût en écritures dépend de la condition **C** (entre 0 et  $b_1 * b_2 * N1 * N2 / b'$ )

(  $b_j$  : capacité d'un bloc du fichier d'entrée  $F_j$  ,  $b'$  : capacité d'un bloc du fichier de sortie )

# Algorithme : Boucles\_Imbriquées (version améliorée)

$i_3 \leftarrow 1 ; j_3 \leftarrow 1$

**Pour**  $i_1 = 1 \dots N1$

// Boucle Externe : Parcours de F1 ...

LireDir( F1,  $i_1$ , buf1 )

**Pour**  $j_1 = 1 \dots \text{buf1.NB}$

**Pour**  $i_2 = 1 \dots N2$

// Boucle Interne : Parcours de F2 ...

LireDir( F2,  $i_2$ , buf2 )

**Pour**  $j_2 = 1 \dots \text{buf2.NB}$

$e1 \leftarrow \text{buf1.Tab}[j_1]$

$e2 \leftarrow \text{buf2.Tab}[j_2]$

**SI** (  $C(e1, e2)$  ) // C : condition de jointure

$\text{buf3.Tab}[j_3] \leftarrow \langle e1 : e2 \rangle ; j_3++$

**SI** (  $j_3 > b'$  )

$\text{buf3.NB} = b ; \text{EcrireDir}( F3, i_3, \text{buf3} ) ; j_3 \leftarrow 1 ; i_3++$

**FSI**

**FSI**

**FP** //  $j_2$

**FP** //  $i_2$

--- Fin de la Boucle Interne (F2) ---

**FP** //  $j_1$

**FP** //  $i_1$

--- Fin de la Boucle Externe (F1) ---

# Algorithme : Boucles\_Imbriquées (version améliorée)

$i_3 \leftarrow 1 ; j_3 \leftarrow 1$

**Pour**  $i_1 = 1 \dots N1$

*// Boucle Externe : Parcours de F1 ...*

*LireDir( F1,  $i_1$ , buf1 )*

**Pour**  $i_2 = 1 \dots N2$

*// Boucle Interne : Parcours de F2 ...*

*LireDir( F2,  $i_2$ , buf2 )*

**Pour**  $j_1 = 1 \dots \text{buf1.NB}$

**Pour**  $j_2 = 1 \dots \text{buf2.NB}$

$e1 \leftarrow \text{buf1.Tab}[j_1]$

$e2 \leftarrow \text{buf2.Tab}[j_2]$

**SI** (  $C( e1, e2 )$  ) *// C : condition de jointure*

$\text{buf3.Tab}[j_3] \leftarrow \langle e1 : e2 \rangle ; j_3++$

**SI** (  $j_3 > b'$  )

$\text{buf3.NB} = b ; \text{EcrireDir}( F3, i_3, \text{buf3} ) ; j_3 \leftarrow 1 ; i_3++$

**FSI**

**FSI**

**FP** *//  $j_2$*

**FP** *//  $j_1$*

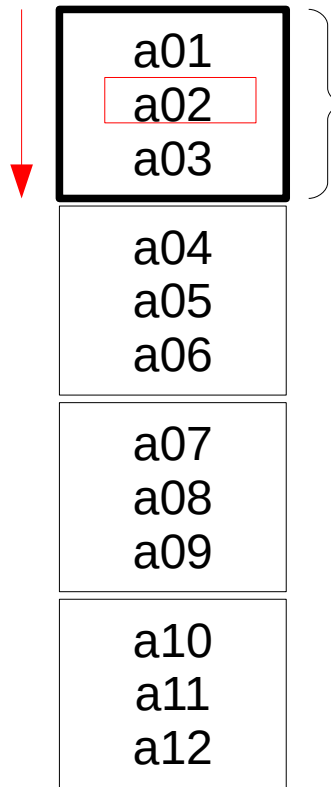
**FP** *//  $i_2$*

*--- Fin de la Boucle Interne (F2) ---*

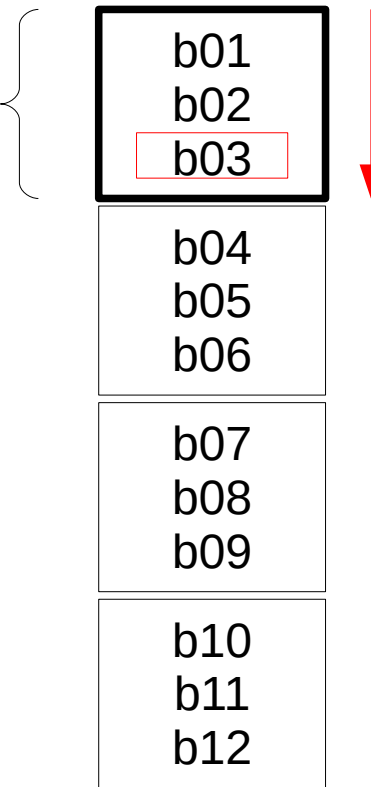
**FP** *//  $i_1$*

*--- Fin de la Boucle Externe (F1) ---*

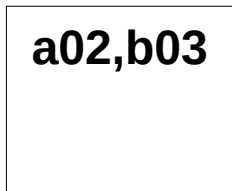
Fichier F1



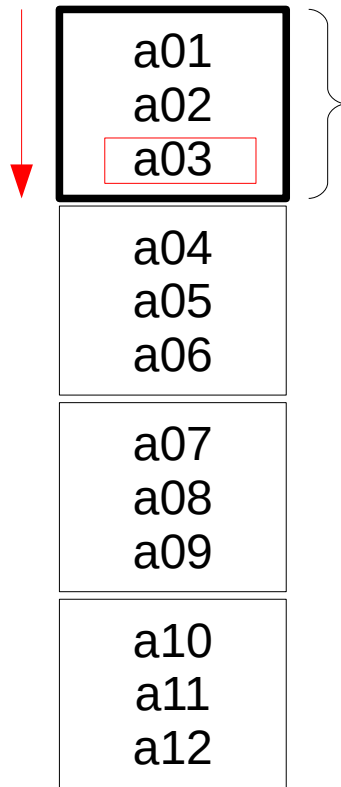
Fichier F2



Fichier Résultat

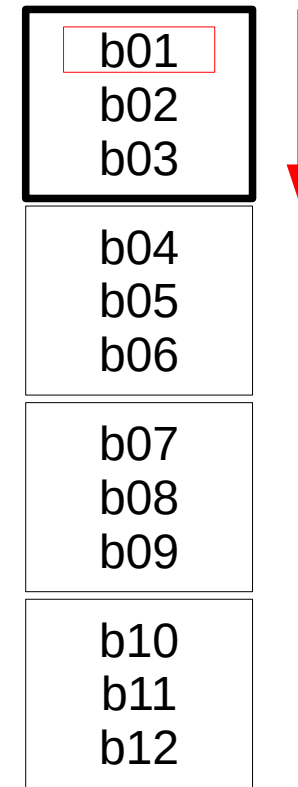


Fichier F1



a01
a02
a03
a04
a05
a06
a07
a08
a09
a10
a11
a12

Fichier F2



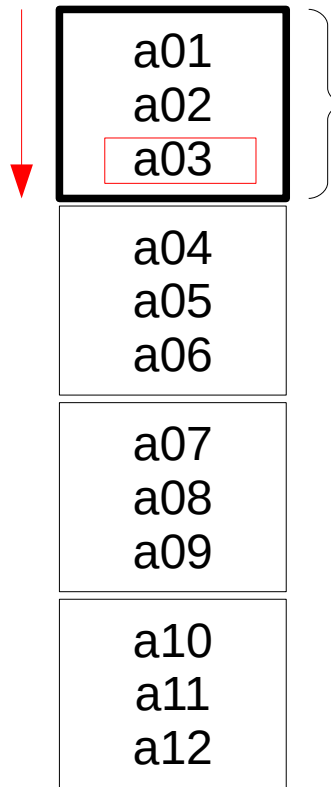
b01
b02
b03
b04
b05
b06
b07
b08
b09
b10
b11
b12



Fichier Résultat

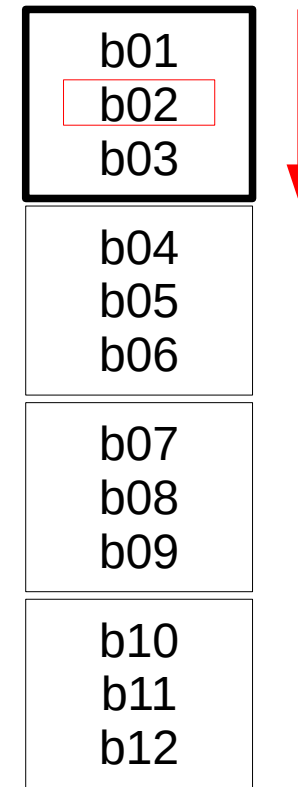
a02,b03
<b>a03,b01</b>

Fichier F1



a01
a02
a03
a04
a05
a06
a07
a08
a09
a10
a11
a12

Fichier F2



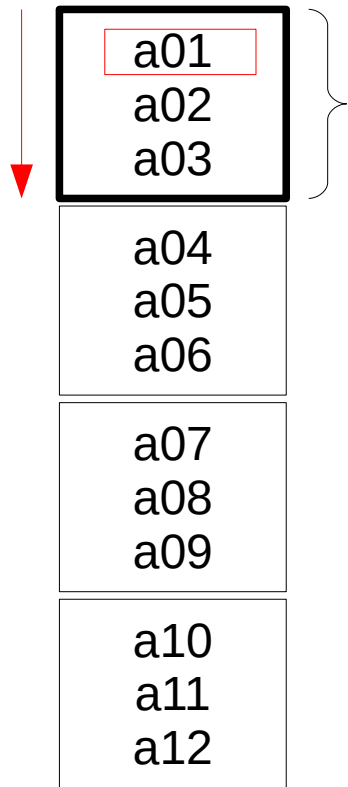
b01
b02
b03
b04
b05
b06
b07
b08
b09
b10
b11
b12



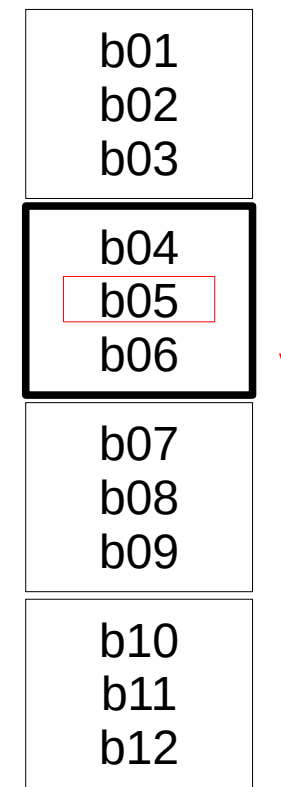
Fichier Résultat

a02,b03
a03,b01
<b>a03,b02</b>

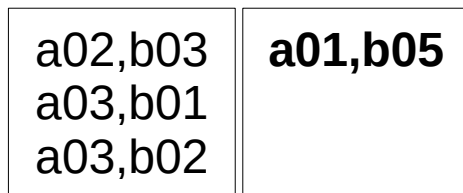
Fichier F1



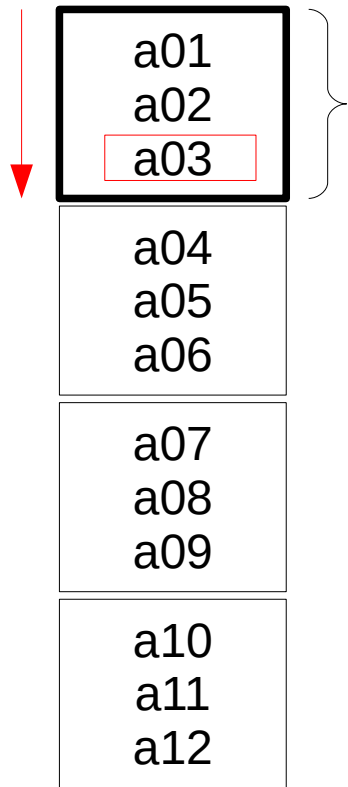
Fichier F2



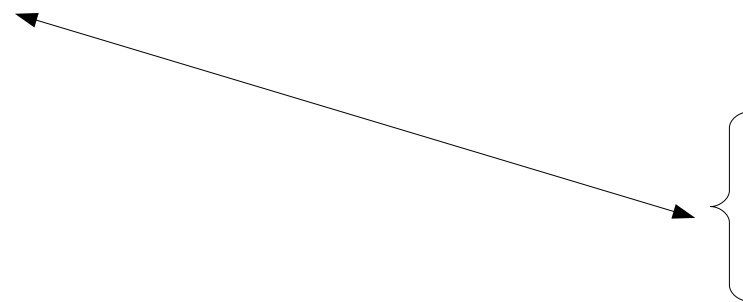
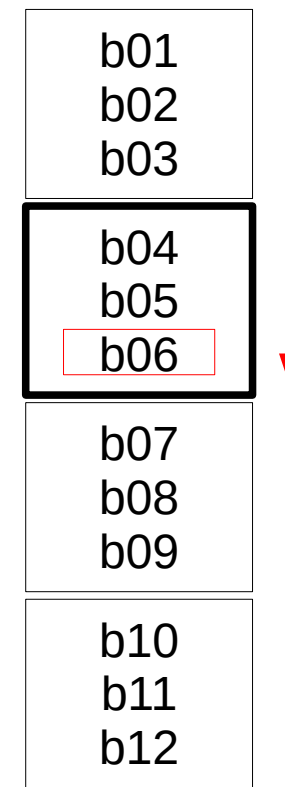
Fichier Résultat



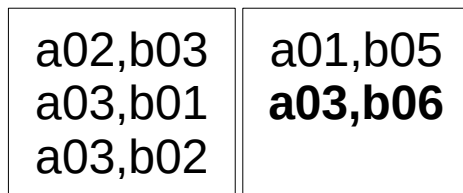
Fichier F1



Fichier F2

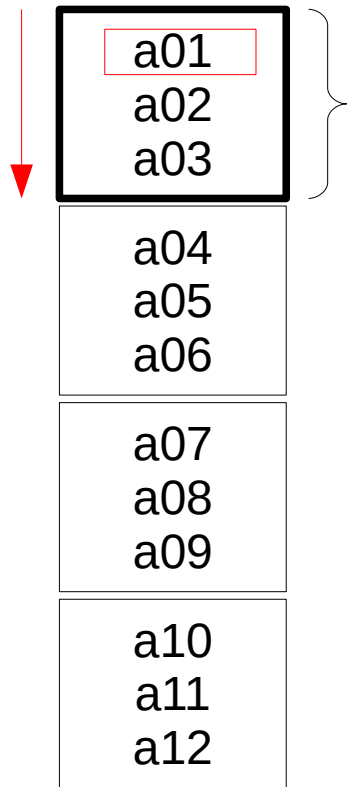


Fichier Résultat

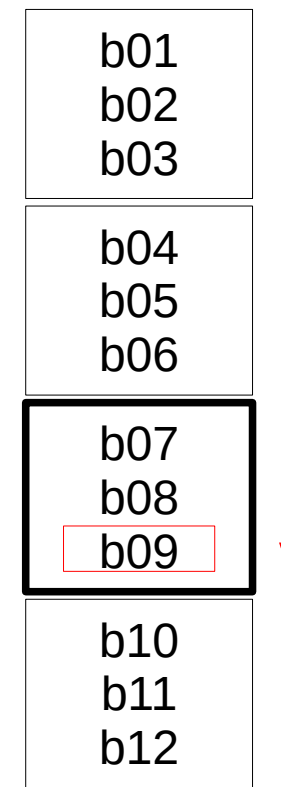




## Fichier F1



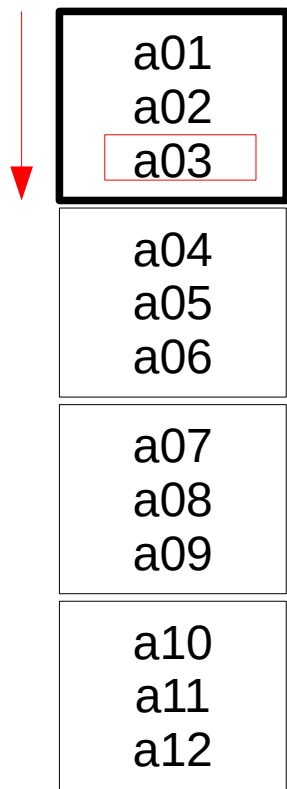
## Fichier F2



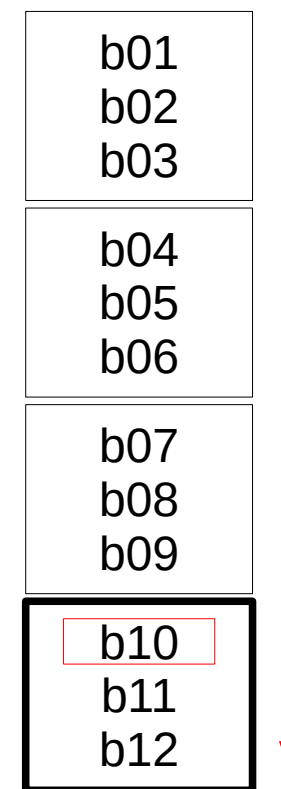
## Fichier Résultat

a02,b03	a01,b05
a03,b01	a03,b06
a03,b02	<b>a01,b09</b>

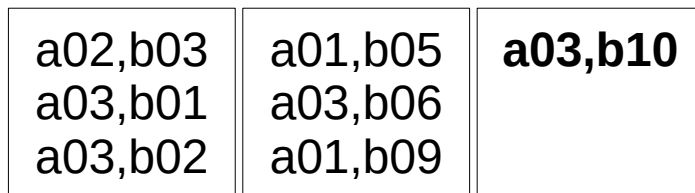
Fichier F1



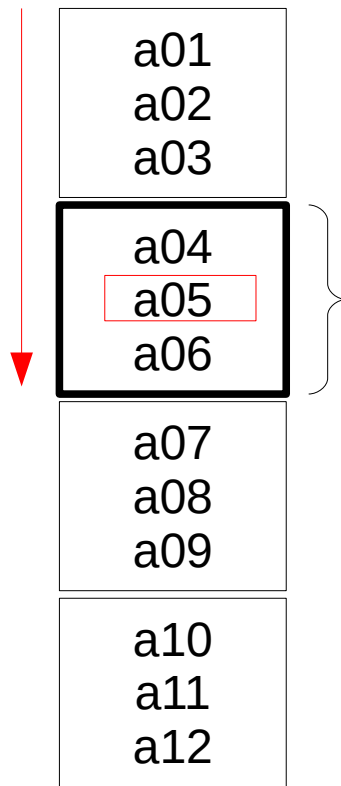
Fichier F2



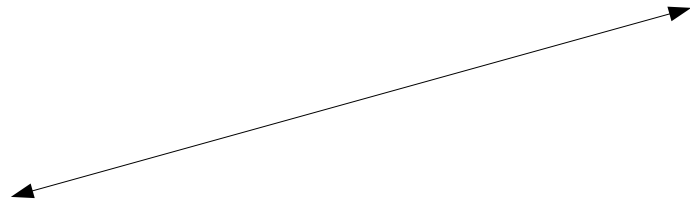
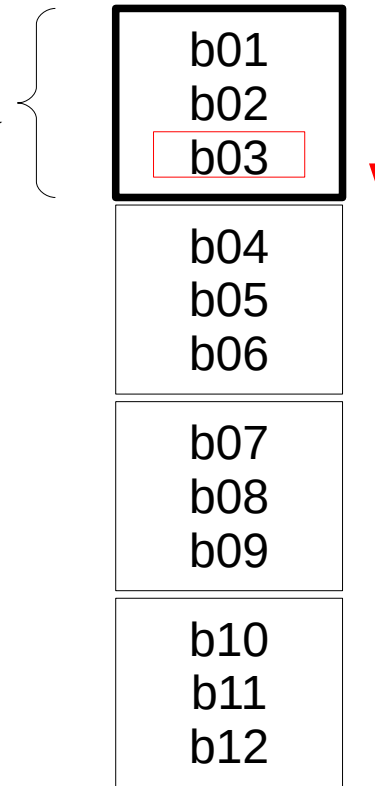
Fichier Résultat



## Fichier F1



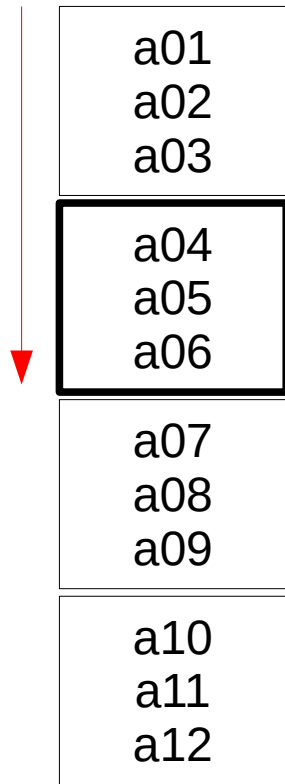
## Fichier F2



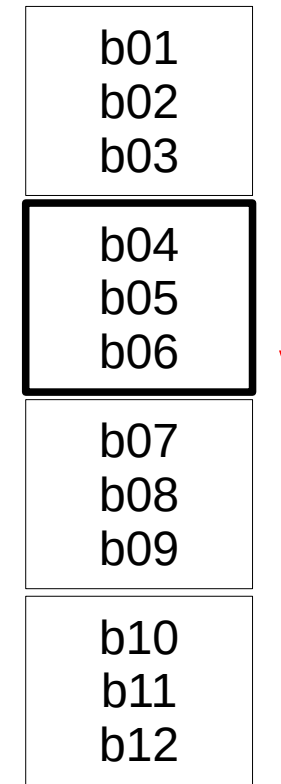
## Fichier Résultat

a02,b03	a01,b05	a03,b10
a03,b01	a03,b06	<b>a05,b03</b>
a03,b02	a01,b09	

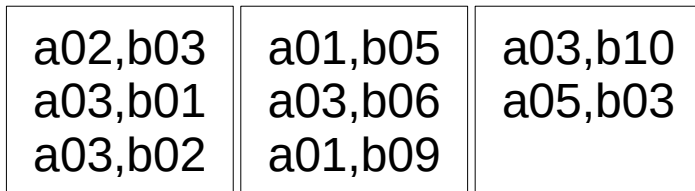
## Fichier F1



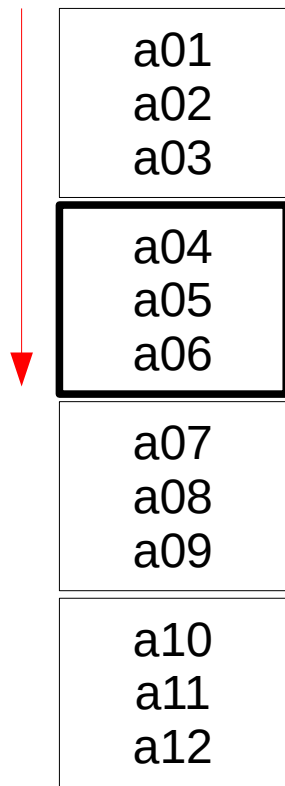
## Fichier F2



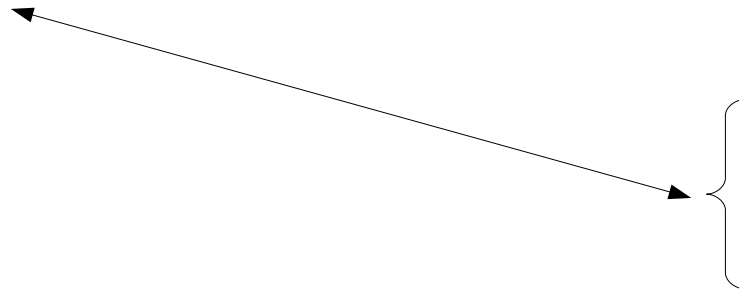
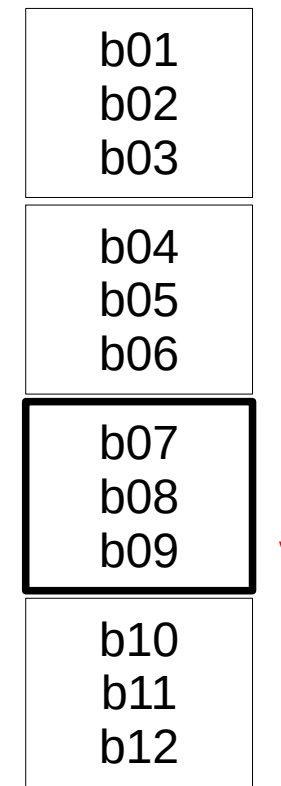
## Fichier Résultat



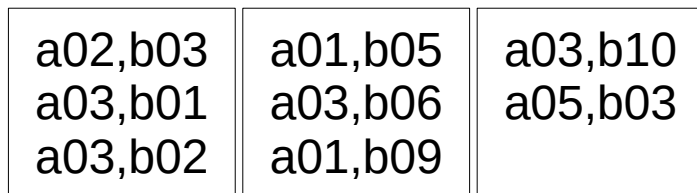
## Fichier F1



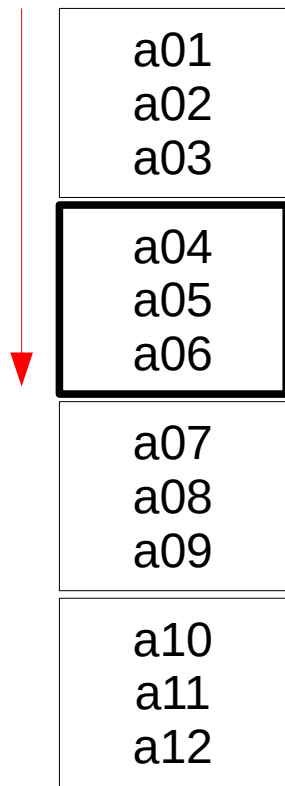
## Fichier F2



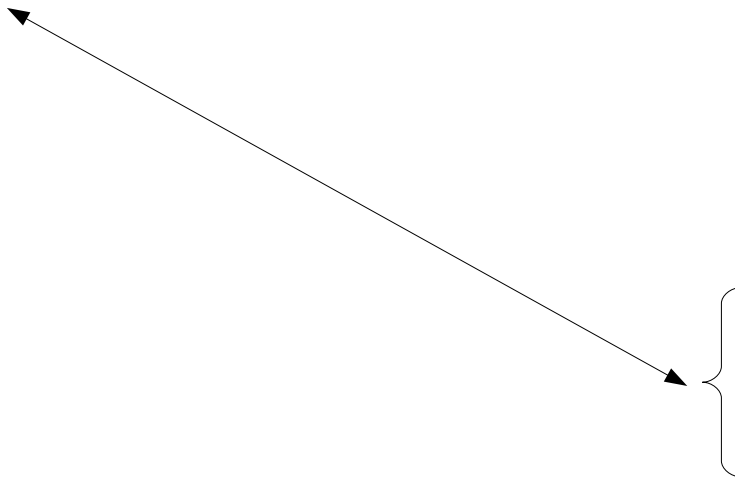
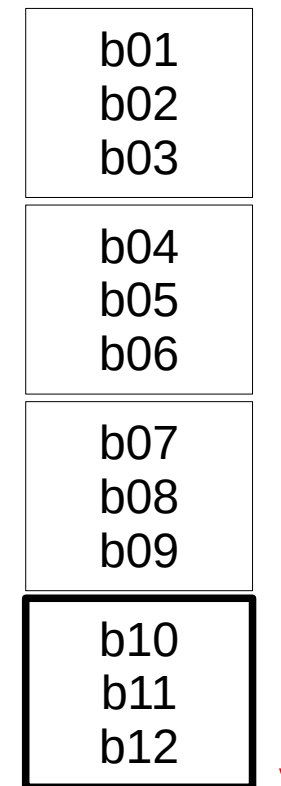
## Fichier Résultat



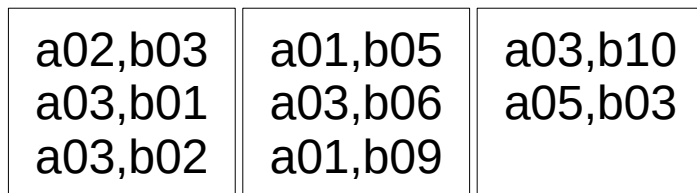
## Fichier F1



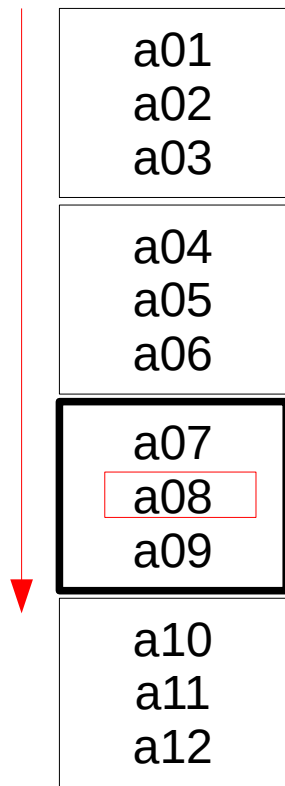
## Fichier F2



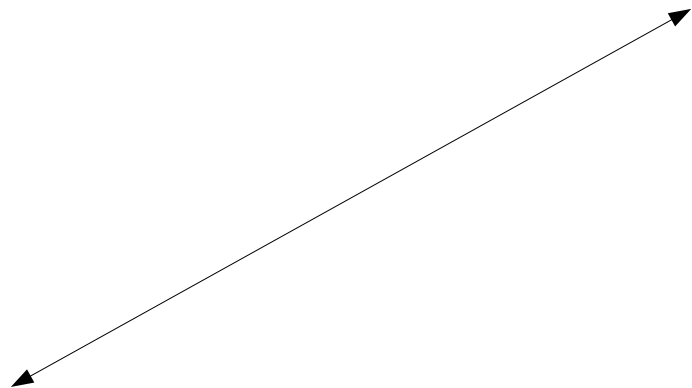
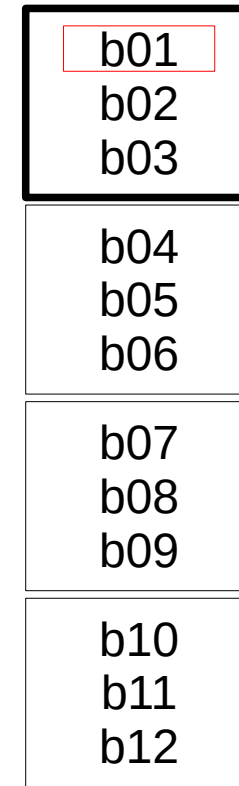
## Fichier Résultat



## Fichier F1



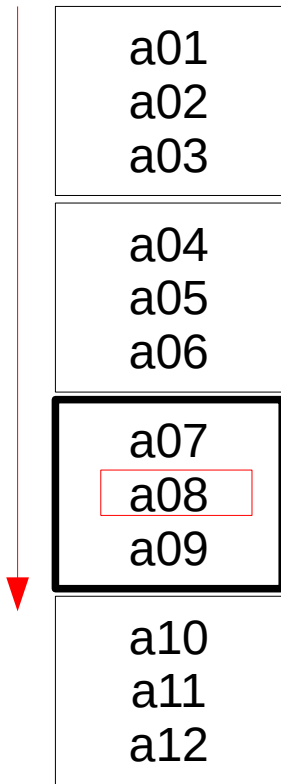
## Fichier F2



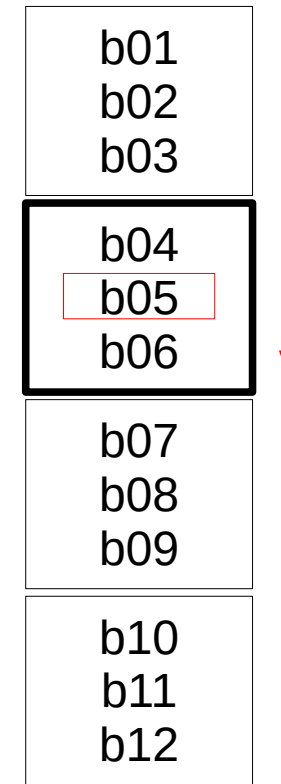
## Fichier Résultat

a02,b03	a01,b05	a03,b10
a03,b01	a03,b06	a05,b03
a03,b02	a01,b09	<b>a08,b01</b>

## Fichier F1



## Fichier F2

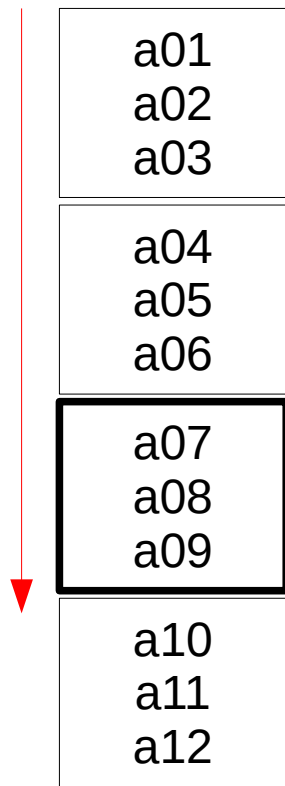


## Fichier Résultat

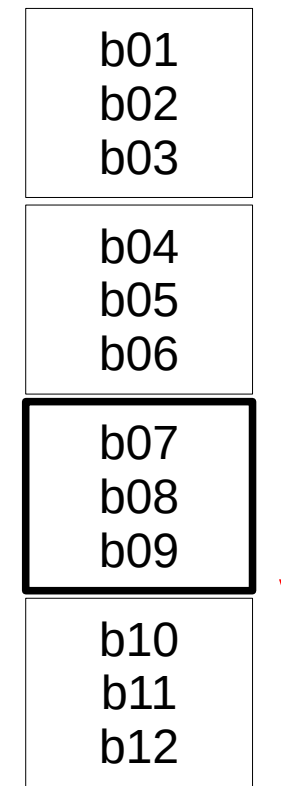
a02,b03	a01,b05	a03,b10	<b>a08,b05</b>
a03,b01	a03,b06	a05,b03	
a03,b02	a01,b09	a08,b01	



## Fichier F1



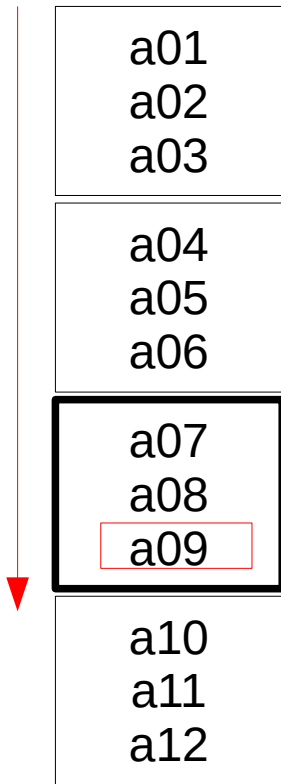
## Fichier F2



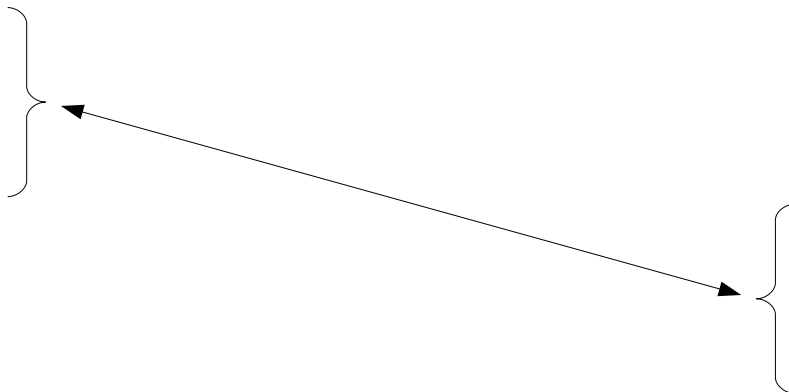
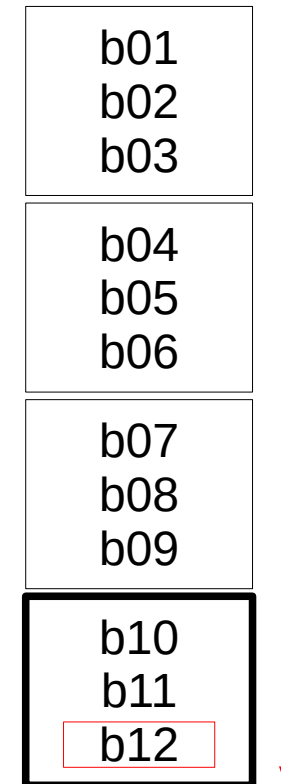
## Fichier Résultat

a02,b03	a01,b05	a03,b10	a08,b05
a03,b01	a03,b06	a05,b03	
a03,b02	a01,b09	a08,b01	

## Fichier F1



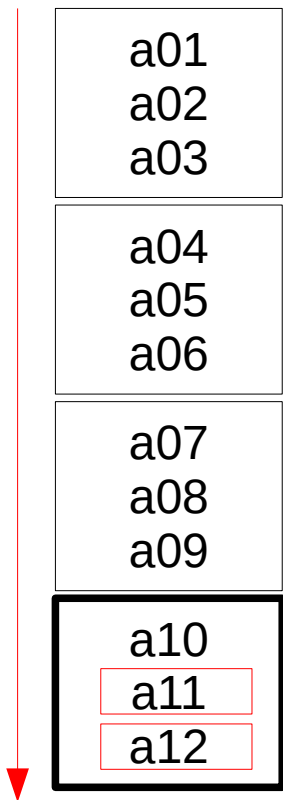
## Fichier F2



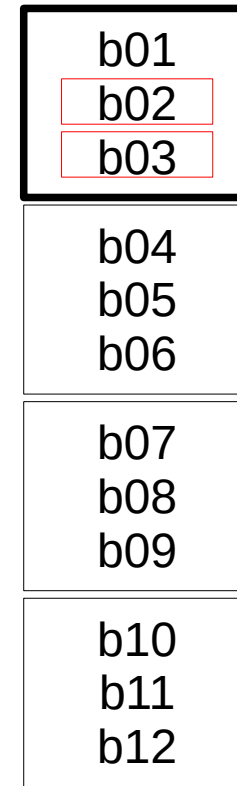
## Fichier Résultat

a02,b03 a03,b01 a03,b02	a01,b05 a03,b06 a01,b09	a03,b10 a05,b03 a08,b01	a08,b05 <b>a09,b12</b>
-------------------------------	-------------------------------	-------------------------------	---------------------------

## Fichier F1



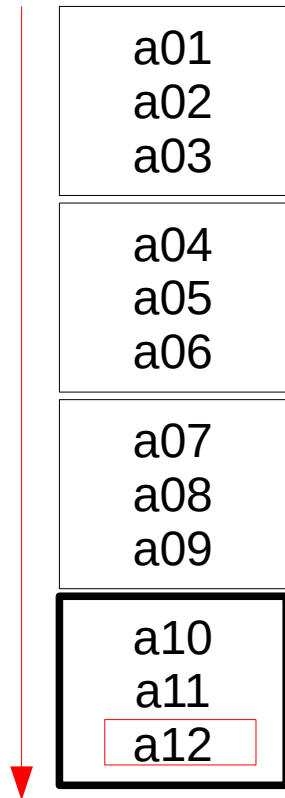
## Fichier F2



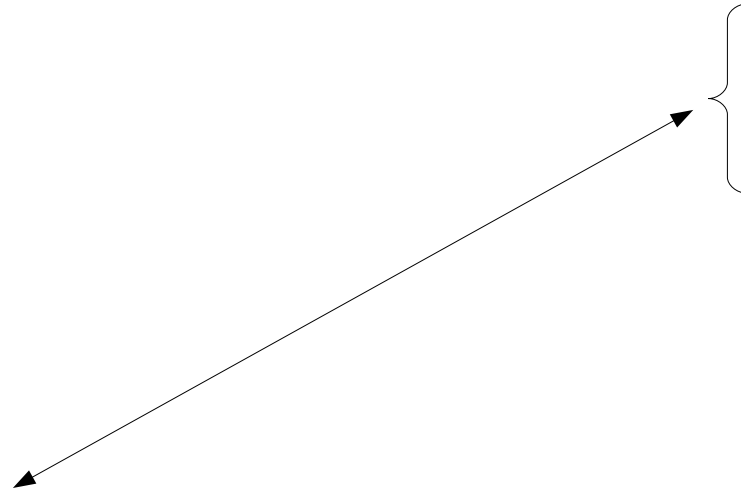
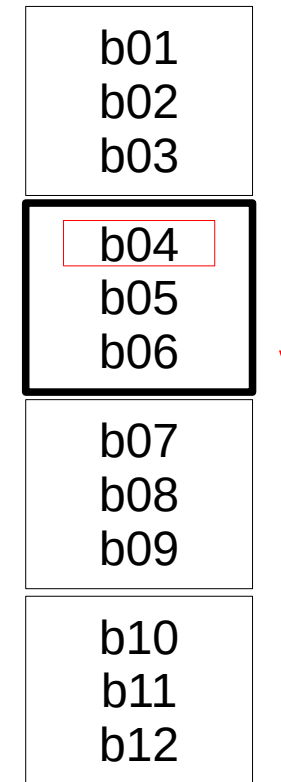
## Fichier Résultat

a02,b03	a01,b05	a03,b10	a08,b05	<b>a11,b03</b>
a03,b01	a03,b06	a05,b03	a09,b12	<b>a12,b03</b>
a03,b02	a01,b09	a08,b01	<b>a11,b02</b>	

## Fichier F1



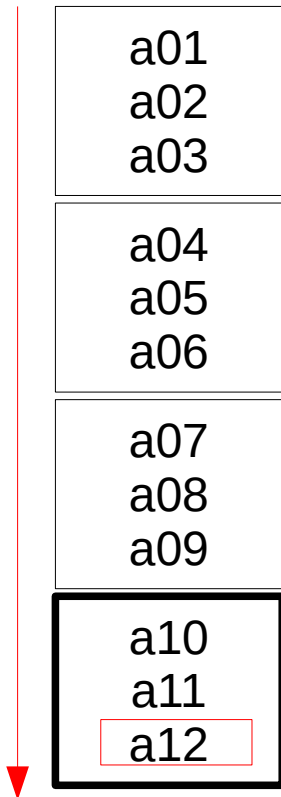
## Fichier F2



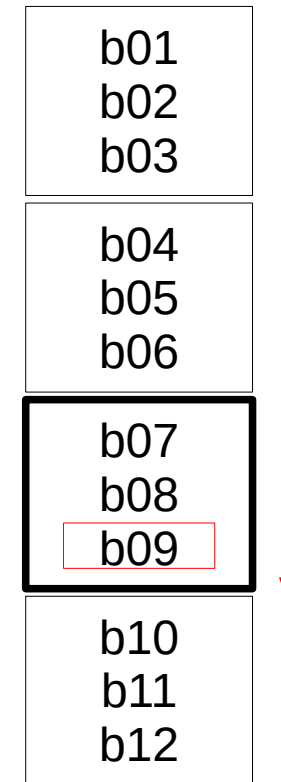
## Fichier Résultat

a02,b03	a01,b05	a03,b10	a08,b05	a11,b03
a03,b01	a03,b06	a05,b03	a09,b12	a12,b03
a03,b02	a01,b09	a08,b01	a11,b02	<b>a12,b04</b>

## Fichier F1



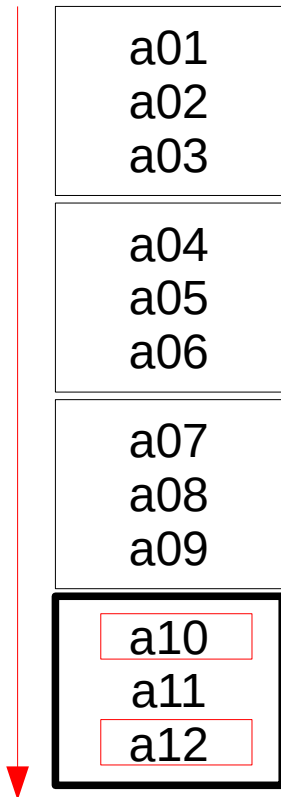
## Fichier F2



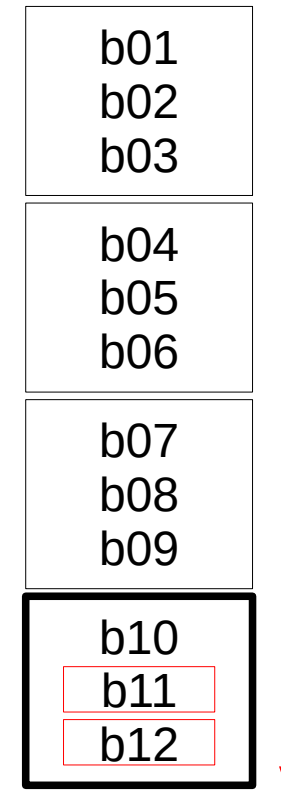
## Fichier Résultat

a02,b03 a03,b01 a03,b02	a01,b05 a03,b06 a01,b09	a03,b10 a05,b03 a08,b01	a08,b05 a09,b12 a11,b02	a11,b03 a12,b03 a12,b04	<b>a12,b09</b>
-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	----------------

## Fichier F1



## Fichier F2



## Fichier Résultat

a02,b03 a03,b01 a03,b02	a01,b05 a03,b06 a01,b09	a03,b10 a05,b03 a08,b01	a08,b05 a09,b12 a11,b02	a11,b03 a12,b03 a12,b04	a12,b09 <b>a10,b11</b> <b>a12,b11</b>	<b>a12,b12</b>
-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	---	----------------

# Algorithme : Boucles\_Imbriquées (version améliorée)

$i_3 \leftarrow 1 ; j_3 \leftarrow 1$

**Pour**  $i_1 = 1 \dots N1$  *// Boucle Externe : Parcours de F1 ...*

LireDir( F1,  $i_1$ , buf1 )

**Pour**  $i_2 = 1 \dots N2$  *// Boucle Interne : Parcours de F2 ...*

LireDir( F2,  $i_2$ , buf2 )

**Pour**  $j_1 = 1 \dots \text{buf1.NB}$  *// boucles ne contenant pas*

**Pour**  $j_2 = 1 \dots \text{buf2.NB}$  *// d'opération d'E/S physiques*

*... SI ( C( e1, e2 ) ) ... EcrireDir(F3,  $i_3$ , buf3) ... FSI ...*

**FP** *//  $j_2$*

**FP** *//  $j_1$*

**FP** *//  $i_2$*  *--- Fin de la Boucle Interne (F2) ---*

**FP** *//  $i_1$*  *--- Fin de la Boucle Externe (F1) ---*

F1 est parcouru **une seule fois** ( N1 lectures )

F2 est parcouru **autant de fois que de blocs dans F1** (N1\*N2 lectures)

**Coût total (en lectures) =  $N1 + N1 * N2$**  (le coût en lecture a été **divisé par  $b_1$** )

Le coût en écritures dépend de la condition **C** (entre 0 et  $b_1 * b_2 * N1 * N2 / b'$ )

(  $b_j$  : capacité d'un bloc du fichier d'entrée  $F_j$  ,  $b'$  : capacité d'un bloc du fichier de sortie )

## **Algorithme : Boucles\_Imbriquées (version améliorée)**

Avec 3 buffers en MC :

F1 est parcouru 1 seule fois ( N1 lectures )

F2 est parcouru autant de fois que de blocs dans F1 (N2\*N1 lectures)

**Coût total (en lectures) =  $N1 + N2 * N1$**

**Peut-on faire mieux avec plus de buffers en MC ?**



# Algorithme : Boucles\_Imbriquées (version améliorée)

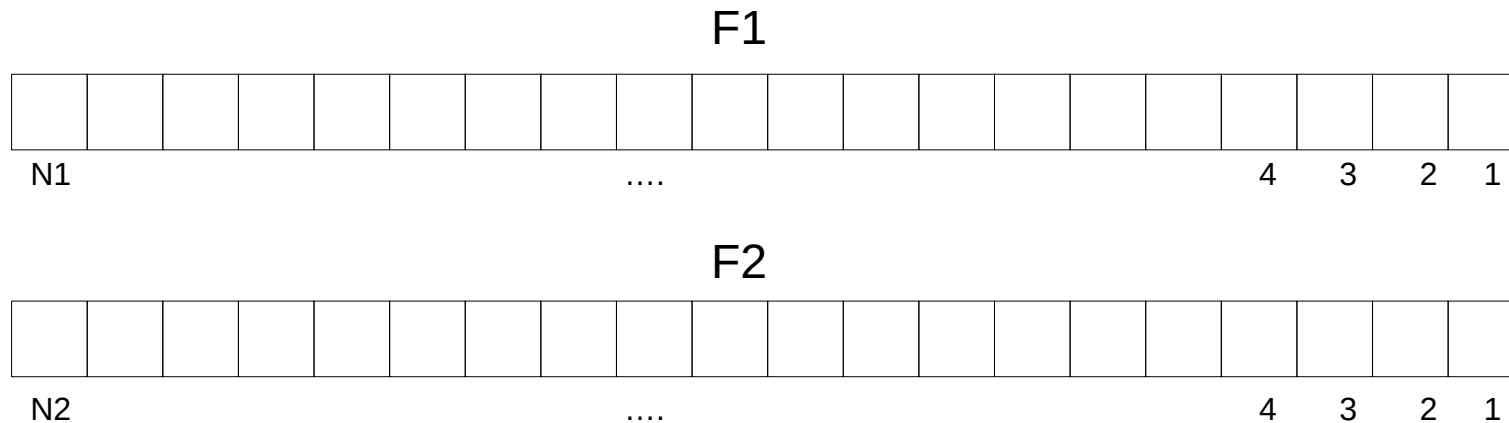
Avec 3 buffers en MC : Coût total (en lectures) =  $N1 + N2 * N1$

F1 est parcouru 1 seule fois (  $N1$  lectures )

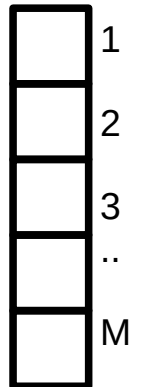
F2 est parcouru autant de fois que de blocs dans F1 ( $N2*N1$  lectures)

## Avec M buffers en MC :

*On peut diminuer le nombre de parcours de F2  
en diminuant le nombre d'itérations de la boucle externe !*



M buffers  
en MC



# Algorithme : Boucles\_Imbriquées (version améliorée)

Avec 3 buffers en MC : Coût total (en lectures) =  $N1 + N2 * N1$

F1 est parcouru 1 seule fois (  $N1$  lectures )

F2 est parcouru autant de fois que de blocs dans F1 ( $N2 * N1$  lectures)

## Avec M buffers en MC :

*On peut diminuer le nombre de parcours de F2*

*en diminuant le nombre d'itérations de la boucle externe !*

*Au lieu de lire F1 bloc par bloc, on va le parcourir par **groupe de plusieurs blocs** :*

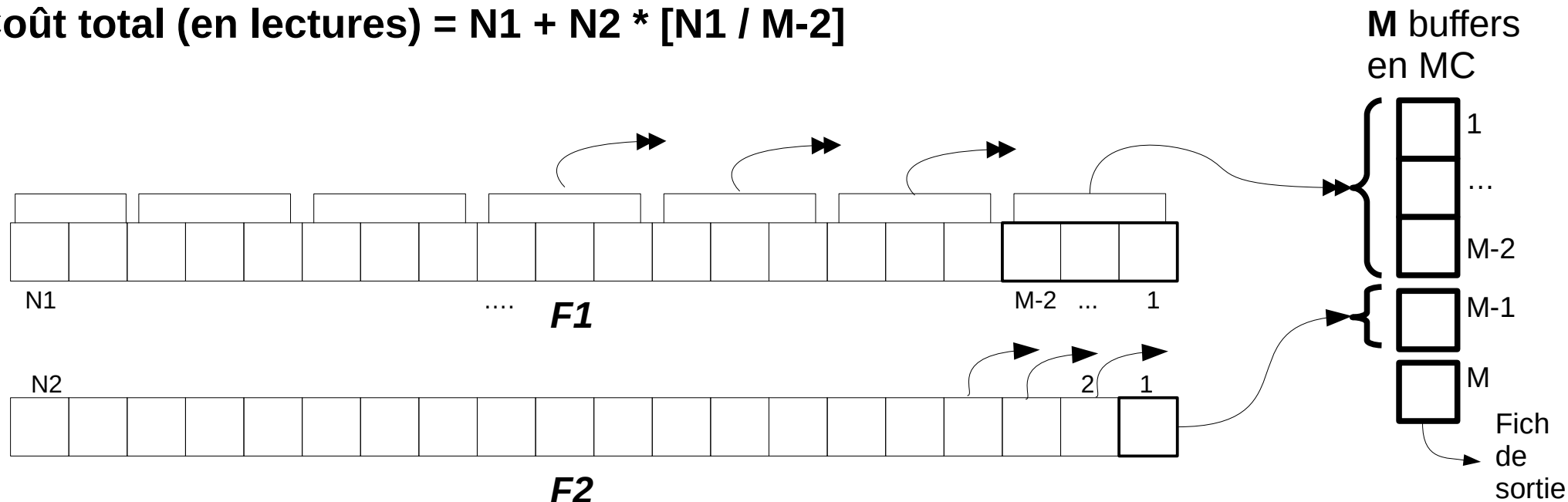
→ **M-2 buffers (le maximum de buffers) pour lire F1** (buf[1 .. M-2])

→ **1 seul buffer pour lire F2** (buf[M-1])

→ **1 buffer pour la sortie** (buf[M])

Le nombre de parcours de F2 est maintenant :  **$[N1 / M-2]$**

**Coût total (en lectures) =  $N1 + N2 * [N1 / M-2]$**



# Algorithme : Boucles\_Imbriquées (version améliorée avec M buffers)

$i_3 \leftarrow 1 ; j_3 \leftarrow 1 ; i_1 \leftarrow 1$

**Pour**  $i_1 = 1 .. N1$ , par pas de  $M-2$  // Boucle Externe : Parcours de F1 ...

**Pour**  $k=1..M-2$  : **SI** ( $i_1+k-1 \leq N1$ ) LireDir( F1,  $i_1+k-1$  , buf[ k ] ) **FSI** **FP**

**Pour**  $i_2 = 1 .. N2$  // Boucle Interne : Parcours de F2 ...

LireDir( F2,  $i_2$ , buf[M-1] )

// jointure en MC entre buf[ 1..M-2 ] et buf[ M-1 ]. Le résultat dans buf[ M ]

**Pour** chaque enreg e1 dans le groupe buf[ 1..M-2 ]

**Pour**  $j_2 = 1 .. \text{buf}[ M-1 ].\text{NB}$

$e2 \leftarrow \text{buf}[ M-1 ].\text{Tab}[ j_2 ]$

**SI** (  $C( e1, e2 )$  ) // C : condition de jointure

$\text{buf}[ M ].\text{Tab}[ j_3 ] \leftarrow \langle e1 : e2 \rangle ; j_3++$

**SI** (  $j_3 > b'$  )

$\text{buf}[ M ].\text{NB} = b ;$  EcrireDir( F3,  $i_3$ , buf[ M ] ) ;  $j_3 \leftarrow 1 ; i_3++$

**FSI**

**FSI**

**FP** // chaque e2 dans buf[M-1]

**FP** // chaque e1 dans buf[1..M-2]

**FP** //  $i_2 ++$

--- Fin de la Boucle Interne (F2) ---

**FP** //  $i_1 += M-2$

--- Fin de la Boucle Externe (F1) ---

# Algorithme de Jointure par Tri-Fusion

( Sort-Merge Join Algorithm )

F1 et F2 composés resp. de N1 et N2 blocs

M buffers sont disponibles en MC.

Applicable uniquement pour les **équi-jointures**

## Principe :

- Trier les fichiers F1 et F2
- Parcourir les 2 fichiers en ordre croissant pour vérifier les correspondances éventuelles entre les enregistrements des 2 fichiers (comme une fusion)

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12

Après la phase de tri  
des deux fichiers

## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12

## Fichier Résultat

--

Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12

Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12

Fichier Résultat

--

Fichier F1

10,a01 20,a02 30,a03	←
30,a04 35,a05 35,a06	
35,a07 40,a08 50,a09	
65,a10 75,a11 87,a12	

Fichier F2

→ 15,b01 15,b02 20,b03	
22,b04 30,b05 35,b06	
35,b07 40,b08 45,b09	
50,b10 65,b11 65,b12	

Fichier Résultat

--

Fichier F1

10,a01 20,a02 30,a03	←
30,a04 35,a05 35,a06	
35,a07 40,a08 50,a09	
65,a10 75,a11 87,a12	

Fichier F2

←	15,b01 15,b02 20,b03
	22,b04 30,b05 35,b06
	35,b07 40,b08 45,b09
	50,b10 65,b11 65,b12

Fichier Résultat

--



## Fichier F1

10,a01 <b>20,a02</b> 30,a03
30,a04 35,a05 35,a06
35,a07 40,a08 50,a09
65,a10 75,a11 87,a12

## Fichier F2

15,b01 15,b02 <b>20,b03</b>
22,b04 30,b05 35,b06
35,b07 40,b08 45,b09
50,b10 65,b11 65,b12

## Fichier Résultat

20,a02,b03
------------

## Fichier F1

10,a01 20,a02 30,a03
30,a04 35,a05 35,a06
35,a07 40,a08 50,a09
65,a10 75,a11 87,a12

## Fichier F2

15,b01 15,b02 20,b03
22,b04 30,b05 35,b06
35,b07 40,b08 45,b09
50,b10 65,b11 65,b12

## Fichier Résultat

20,a02,b03
------------

## Fichier F1

10,a01
20,a02
30,a03



30,a04
35,a05
35,a06

35,a07
40,a08
50,a09

65,a10
75,a11
87,a12

## Fichier F2

15,b01
15,b02
20,b03



22,b04
30,b05
35,b06

35,b07
40,b08
45,b09

50,b10
65,b11
65,b12

## Fichier Résultat

20,a02,b03
------------

## Fichier F1

10,a01 20,a02 <b>30,a03</b>
30,a04 35,a05 35,a06
35,a07 40,a08 50,a09
65,a10 75,a11 87,a12

## Fichier F2

15,b01 15,b02 20,b03
22,b04 <b>30,b05</b> 35,b06
35,b07 40,b08 45,b09
50,b10 65,b11 65,b12

## Fichier Résultat

20,a02,b03 30,a03,b05
--------------------------

## Fichier F1

10,a01
20,a02
30,a03
<b>30,a04</b>
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
<b>30,b05</b>
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03
30,a03,b05
30,a04,b05

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03
30,a03,b05
30,a04,b05

## Fichier F1

10,a01
20,a02
30,a03
30,a04
<b>35,a05</b>
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
<b>35,b06</b>
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06
30,a03,b05	
30,a04,b05	

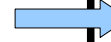
## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
<b>35,a06</b>
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
<b>35,b06</b>
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06
30,a03,b05	35,a06,b06
30,a04,b05	



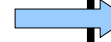
## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
<b>35,a07</b>
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
<b>35,b06</b>
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06
30,a03,b05	35,a06,b06
30,a04,b05	35,a07,b06

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12

## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12

## Fichier Résultat

20,a02,b03	35,a05,b06
30,a03,b05	35,a06,b06
30,a04,b05	35,a07,b06

## Fichier F1

10,a01
20,a02
30,a03
30,a04
<b>35,a05</b>
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
<b>35,b07</b>
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07
30,a03,b05	35,a06,b06	
30,a04,b05	35,a07,b06	

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
<b>35,a06</b>
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
<b>35,b07</b>
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07
30,a03,b05	35,a06,b06	35,a06,b07
30,a04,b05	35,a07,b06	

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
<b>35,a07</b>
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
<b>35,b07</b>
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07
30,a03,b05	35,a06,b06	35,a06,b07
30,a04,b05	35,a07,b06	35,a07,b07

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07
30,a03,b05	35,a06,b06	35,a06,b07
30,a04,b05	35,a07,b06	35,a07,b07

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
<b>40,a08</b>
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
<b>40,b08</b>
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07	40,a08,b08
30,a03,b05	35,a06,b06	35,a06,b07	
30,a04,b05	35,a07,b06	35,a07,b07	

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12

## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07	40,a08,b08
30,a03,b05	35,a06,b06	35,a06,b07	
30,a04,b05	35,a07,b06	35,a07,b07	



## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07	40,a08,b08
30,a03,b05	35,a06,b06	35,a06,b07	
30,a04,b05	35,a07,b06	35,a07,b07	

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
<b>50,a09</b>
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
<b>50,b10</b>
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07	40,a08,b08
30,a03,b05	35,a06,b06	35,a06,b07	50,a09,b10
30,a04,b05	35,a07,b06	35,a07,b07	

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07	40,a08,b08
30,a03,b05	35,a06,b06	35,a06,b07	50,a09,b10
30,a04,b05	35,a07,b06	35,a07,b07	

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
<b>65,a10</b>
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
<b>65,b11</b>
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07	40,a08,b08
30,a03,b05	35,a06,b06	35,a06,b07	50,a09,b10
30,a04,b05	35,a07,b06	35,a07,b07	65,a10,b11

## Fichier F1

10,a01  
20,a02  
30,a03

30,a04  
35,a05  
35,a06

35,a07  
40,a08  
50,a09

**65,a10** ←  
75,a11  
87,a12

## Fichier F2

15,b01  
15,b02  
20,b03

22,b04  
30,b05  
35,b06

35,b07  
40,b08  
45,b09

50,b10  
65,b11  
→ **65,b12**

## Fichier Résultat

20,a02,b03  
30,a03,b05  
30,a04,b05

35,a05,b06  
35,a06,b06  
35,a07,b06

35,a05,b07  
35,a06,b07  
35,a07,b07

40,a08,b08  
50,a09,b10  
65,a10,b11

65,a10,b12

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07	40,a08,b08	65,a10,b12
30,a03,b05	35,a06,b06	35,a06,b07	50,a09,b10	
30,a04,b05	35,a07,b06	35,a07,b07	65,a10,b11	

## Fichier F1

10,a01
20,a02
30,a03
30,a04
35,a05
35,a06
35,a07
40,a08
50,a09
65,a10
75,a11
87,a12



## Fichier F2

15,b01
15,b02
20,b03
22,b04
30,b05
35,b06
35,b07
40,b08
45,b09
50,b10
65,b11
65,b12



Fin de l'algorithme

## Fichier Résultat

20,a02,b03	35,a05,b06	35,a05,b07	40,a08,b08	65,a10,b12
30,a03,b05	35,a06,b06	35,a06,b07	50,a09,b10	
30,a04,b05	35,a07,b06	35,a07,b07	65,a10,b11	

## Deuxième phase de l'algorithme de jointure par tri-fusion (après le tri de F1 et F2)

Pour éviter la **re-lecture de blocs déjà lus** (pour les valeurs dupliquées), on peut utiliser un tableau **E[ ]** pour stocker les enreg ayant la même valeur courante de l'attribut de jointure :

... // initialisations(i1, i2, i3, j1, j2, j3) et lecture des premiers blocs de F1 et F2

**TQ** ( non FinF1 et non FinF2 )

**SI** (buf1.tab[j1].attr < buf2.tab[j2].attr )   Avancer\_dans(F1, buf1, i1, j1, FinF1 )

**SINON**

**SI** ( buf1.tab[j1].attr > buf2.tab[j2].attr )   Avancer\_dans(F2, buf2, i2, j2, FinF2 )

**SINON** // égalité d'attributs entre F1 et F2

**E[1]** ← buf1.tab[j1] ; k ← 1

**TQ** ( non FinF1 et buf1.tab[j1].attr = **E[k]**.attr )

                k++

                Avancer\_dans(F1, buf1, i1, j1, FinF1 )

**FTQ**

**TQ** ( non FinF2 et buf2.tab[j2].attr = **E[1]**.attr )

**Pour** i = 1 .. k-1

                    buf3.tab[j3] ← concat( **E[i]** , buf2.tab[j2] ) ; j3 ++

**SI** ( j3 > b' ) buf3.NB ← b' ; EcrireDir(F3, i3 ; buf3 ) ; i3 ++ ; j3 ← 1 ; **FSI**

**FP**

                Avancer\_dans(F2, buf2, i2, j2, FinF2 )

**FTQ**

**FSI**

**FSI**

**FTQ**

**SI** ( j3 > 1 )   buf3.NB ← j3 - 1 ; EcrireDir(F3, i3 ; buf3 ) ; i3 ++   **FSI**



# Coût de l'algorithme de jointure par tri-fusion

L'algorithme effectue : 2 Tris et 1 Parcours de type Fusion

## Nombre d'opérations de lectures de blocs

Phase de lecture du Tri de F1	:	$N_1 (1 + \text{Log}_{M-1} [N_1 / M])$
Phase de lecture du Tri de F2	:	$N_2 (1 + \text{Log}_{M-1} [N_2 / M])$
Phase de lecture du parcours de type Fusion	:	$(N_1 + N_2)$

## Nombre d'opérations d'écritures de blocs

Phase d'écriture du Tri de F1	:	$N_1 (1 + \text{Log}_{M-1} [N_1 / M])$
Phase d'écriture du Tri de F2	:	$N_2 (1 + \text{Log}_{M-1} [N_2 / M])$
Phase d'écriture du parcours de type Fusion	:	$\alpha$ ( <i>dépend de la sélectivité</i> )

## Coût total :

$$2N_1 (1 + \text{Log}_{M-1} [N_1 / M]) + 2N_2 (1 + \text{Log}_{M-1} [N_2/M]) + (N_1 + N_2 + \alpha)$$

# Algorithme de Jointure par Hachage

( Hash Join Algorithm )

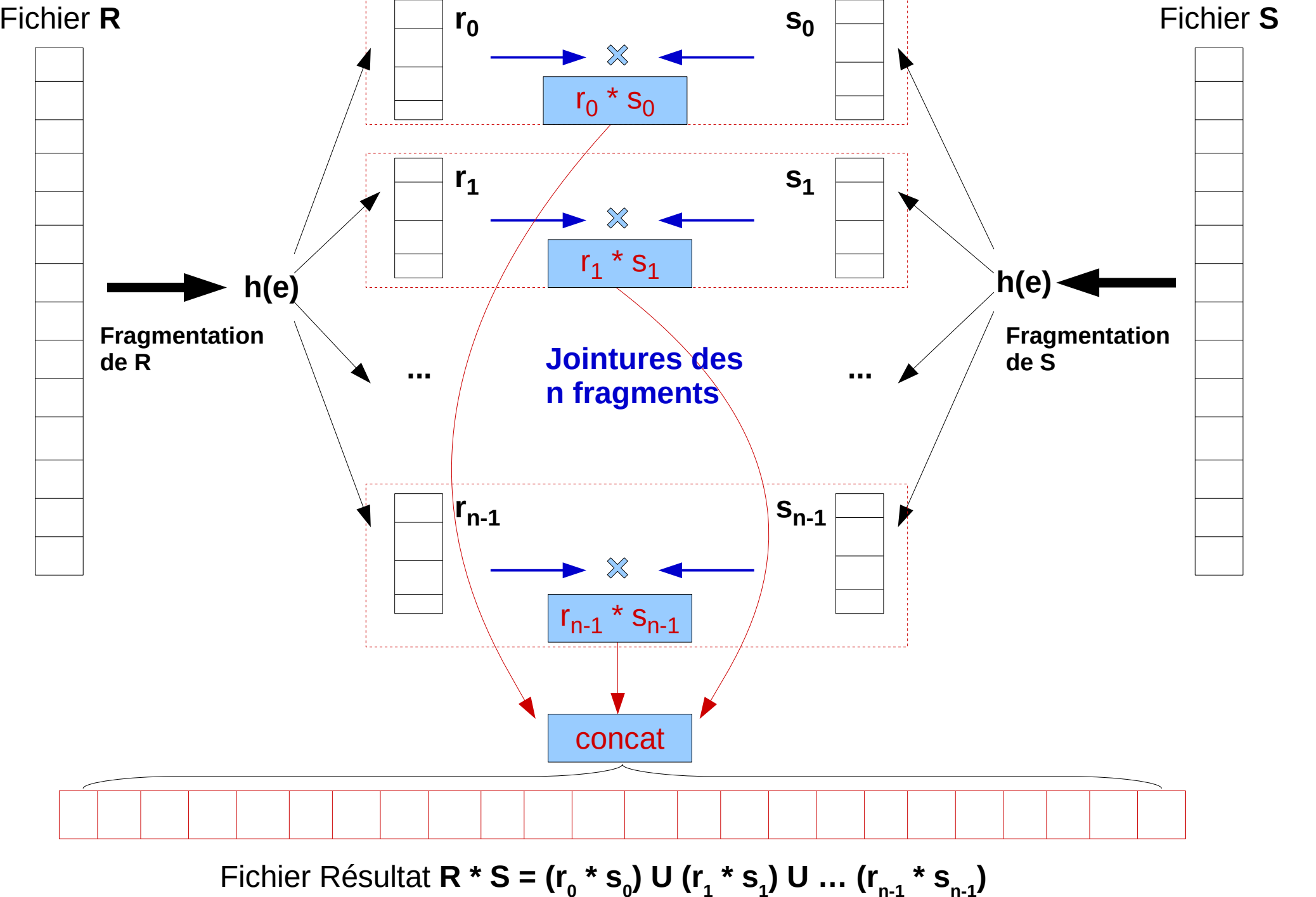
**R** et **S** composés resp. de **Nr** et **Ns** blocs

**M** buffers sont disponibles en **MC**.

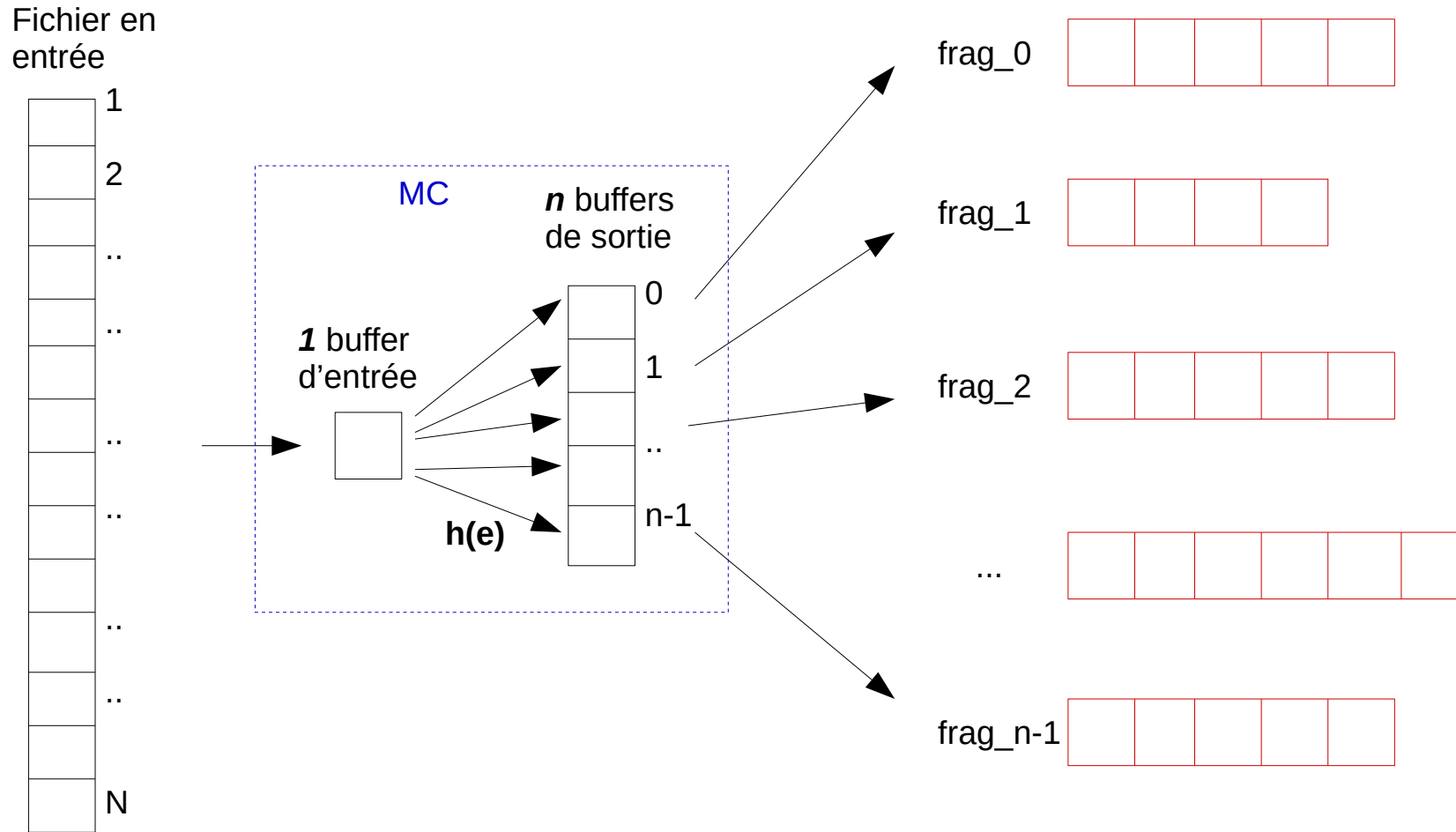
Applicable uniquement pour les **équi-jointures**

## Principe :

- Fragmenter **R** et **S** avec une fonction de hachage  
 $\mathbf{R} = [\mathbf{r}_0, \mathbf{r}_1, \dots \mathbf{r}_{n-1}]$  et  $\mathbf{S} = [\mathbf{s}_0, \mathbf{s}_1, \dots \mathbf{s}_{n-1}]$
- Effectuer plusieurs petites jointures entre fragments de même indice :  $\mathbf{r}_0 * \mathbf{s}_0, \mathbf{r}_1 * \mathbf{s}_1, \dots \mathbf{r}_{n-1} * \mathbf{s}_{n-1}$
- Concaténer les différents résultats obtenus



# Fragmentation des fichiers en entrée



**Coût de l'opération :**

**N Lectures** de blocs du fichier en entrée et **N + n Ecritures** de blocs pour les fragments de sortie

## Jointure d'un fragment : $r_i * s_i$

### a) Etape du « *Build* »

Construction d'une table de hachage en  
MC  $\Rightarrow$  Parcours du fragment  $S_i$

Table de hachage  
de  $S_i$  en MC



$s_i$  en MS



$h'(e)$

Soit  $T$  une table de hachage pouvant stocker les enregistrements du fichier  $S$

*// Etape du build ...*

**Pour**  $k = 1 .. nbBloc(s_i)$

$LireDir(s_i, k, buf_S)$  ;

**Pour**  $j = 1, buf_S.NB$  )

$e \leftarrow buf_S.tab[j]$  ;

**Insérer  $e$  dans la case  $h'(e.attr)$  de  $T$  ;**

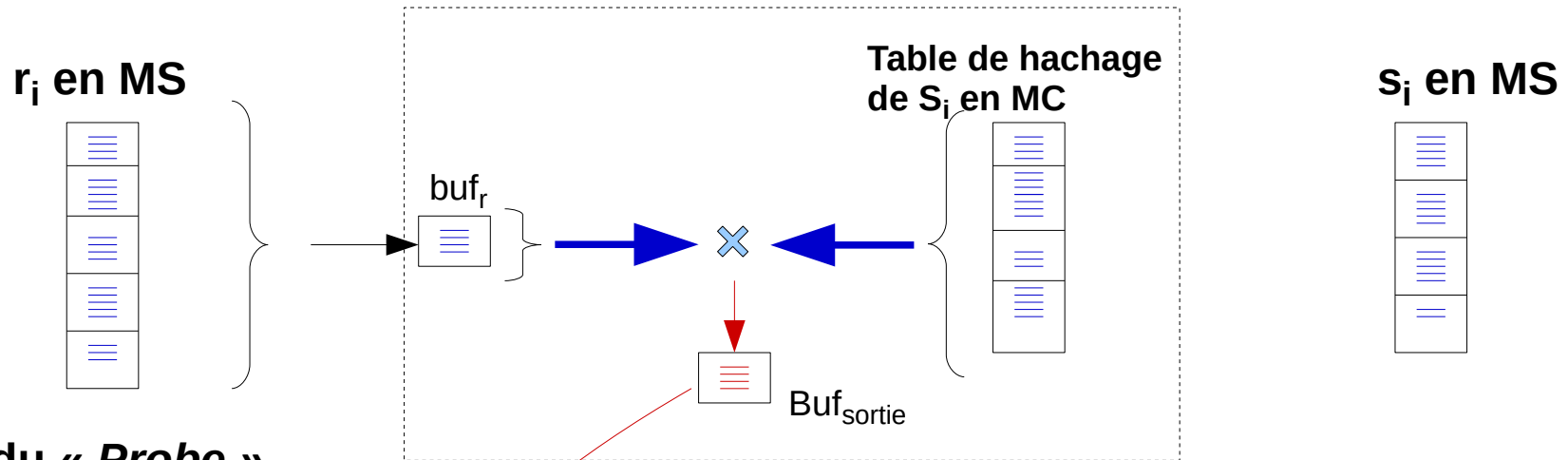
**FP ;**

**FP**

# Jointure d'un fragment : $r_i * s_i$

## a) Etape du « Build »

Construction d'une table de hachage en MC  $\Rightarrow$  Parcours du fragment  $s_i$



## b) Etape du « Probe »

Parcourir le fragment  $r_i$  (bloc par bloc) et utiliser la table de hachage pour effectuer la jointure avec les enreg de  $s_i$

// Etape du probe ...

**Pour**  $k = 1 \dots nbBloc(r_i)$

LireDir(  $r_i, k, buf_r$  ) ;

**Pour**  $j = 1, buf_r.NB$  )

$e \leftarrow buf_r.tab[j]$  ;

Rechercher les correspondances

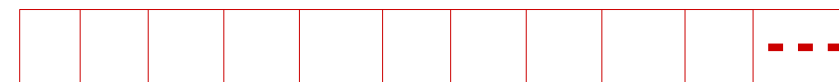
uniquement dans la case  $h'(e.attr)$  de  $T$  ;

*Si Condition vérifiée, mettre le résultat de la concaténation dans le fichier de sortie*

Concaténation en fin de fichier

**FP ;**

**FP**



Fichier Résultat

# Algorithme global de la jointure par Hachage

// En entrée  $R$  est composé de  $N_r$  blocs et  $S$  est composé de  $N_s$  blocs

**Fragmenter  $R$  par hachage ( $h$ ) en :  $r_0, r_1, \dots, r_{n-1}$  ;** //  $\Rightarrow$   **$N_r$  lectures +  $(N_r + n)$  écritures**

**Fragmenter  $S$  par hachage ( $h$ ) en :  $s_0, s_1, \dots, s_{n-1}$  ;** //  $\Rightarrow$   **$N_s$  lectures +  $(N_s + n)$  écritures**

**Pour  $i = 0, n-1$**

*/\* Construction d'un index par hachage du fragment ' $s_i$ ' : étape du **Build** \*/*

*Lire bloc par bloc, le fragment ' $s_i$ ' et construire un index par hachage en MC  
(en utilisant une autre fonction de hachage  $h'$ , **différente de  $h$** )*

*/\* Jointure entre les enregistrements de ' $r_i$ ' et ceux de ' $s_i$ ' : étape du **Probe** \*/*

**Pour** chaque enreg  $tr$  dans le frag ' $r_i$ ' (lire ' $r_i$ ' bloc par bloc)

- récupérer avec l'index en MC les enreg  $ts$  ayant la même valeur d'attribut que  $tr$   
( la condition de jointure )
- **Pour** chaque  $ts$  récupéré  
rajouter la concaténation  $\langle ts.tr \rangle$  dans le résultat

**FP**

**FP**

**FP**

Toutes les étapes 'Build' coûtent au total  $N_s + n$  lectures

Toutes les étapes 'Probe' coûtent au total  $N_r + n$  lectures +  $\alpha$  écritures

**Coût total =  $3(N_r + N_s) + 4n + \alpha$  opérations d'E/S**

# Comparaison théorique des 3 approches

Taille de F1 en blocs =  $N_1$ . La capacité maximale d'un bloc de F1 =  $b_1$  enregistrements

Taille de F2 en blocs =  $N_2$ . La capacité maximale d'un bloc de F2 =  $b_2$  enregistrements

## - Par Boucles Imbriquées : $O(N^2)$

algo Naïf (sans buffers)  $N_1 + N_2 * N_1 * b_1 + \alpha$  opérations d'E/S

algo amélioré avec M buffers  $N_1 + N_2 * [N_1 / M - 2] + \alpha$  opérations d'E/S  
(  $M \geq 3$  )

## - Par Tri-Fusion : $O(N \log N)$

$2N_1 (1 + \log_{M-1} [N_1 / M]) + 2N_2 (1 + \log_{M-1} [N_2 / M]) + (N_1 + N_2 + \alpha)$  opérations d'E/S

## - Par Hachage : $O(N)$

$3(N_1 + N_2) + 4n + \alpha$  opérations d'E/S

Le coût de la construction du fichier résultat ( $\alpha$ ) dépend de la condition **C** et des données.

$\alpha$  varie entre : **0** et  $(b_1 * N_1) * (b_2 * N_2) / b'$  écritures de blocs

(La capacité maximale d'un bloc du fichier résultat =  $b'$  enregistrements)



# Comparaison numérique des 3 approches

$N1 = N2 = 100\,000$  blocs /  $b1 = b2 = 20$  enregistrements par bloc / temps d'une E/S = 20 microsecondes

La construction du fichier résultat n'est pas comptabilisée ( $\alpha = 0$ )

Algorithmes	Nb d'E/S	Temps
<hr/>		
<b>Boucles Imbriquées</b>		
algo Naïf (sans buffers)	$2 * 10^{11}$	46 jours
algo amélioré avec 3 buffers	$10^{10}$	55 heures (2,3 jours)
algo amélioré avec 100 buffers	$102 * 10^6$	34 minutes
algo amélioré avec 1000 buffers	$10,12 * 10^6$	202 secondes
<b>Par Tri-Fusion</b>		
avec 100 buffers	$1,2 * 10^6$	24 secondes
avec 1000 buffers	866705	17,3 secondes
<b>Par Hachage</b>		
avec 100 buffers	606000	12,1 secondes
avec 1000 buffers	600600	12 secondes