Bases de Données & Optimisation : Évaluation des opérateurs relationnels

Thomas Gerald

September 29, 2025

Laboratoire Interdisciplinaire des Sciences du Numérique – LISN, CNRS
thomas.gerald@lisn.upsaclay.fr

Évaluation d'opérateurs relationnels

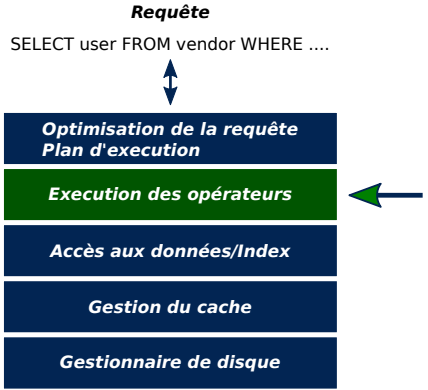
Évaluation d'opérateurs relationnels

Évaluation des opérateurs

- Un bref rappel sur les différentes opérations (+SQL)
- Comment exécuter un opérateur

Objectifs :

- Être capable de déterminer quelle implémentation de l'opérateur choisir en fonction des index et des informations disponibles.
- Être en mesure de dérouler et d'expliquer les algorithmes pour les opérations de jointure



Algèbre relationnel

"Dans la théorie des bases de données, l'algèbre relationnelle est une théorie qui utilise des structures algébriques pour modéliser les données et définir des requête. Cette théorie a été introduite par Edgar F. Codd."

Une petite révision sur les opérateurs

- **Les opérateurs unaires :** La projection π, la sélection σ
- **Les opérateurs ensemblistes :** L'union $\cup$, l'intersection $\cap$, la différence $-$, Le produit cartésien $\times$
- **Les jointures :** La jointure $\bowtie$ (d'autres existent)

Algèbre relationnel (rappel ou pas) : Opérateurs unaires

Projection : $\pi_{a,b,c...}$
π est l la projection, elle consiste en la sélection des champs.

- $\pi_{title,year}(T1)$
-

```sql
SELECT title, year FROM T1;
```

T1

MID	TITLE	YEAR
2	Ariel	1988
5	Star Wars	1977
3	Shadow in Paradise	1986
7	Dancer in the Dark	2000
80	Nausicaä	1984
71	Billy Elliot	2000

Algèbre relationnel (rappel ou pas) : Opérateurs unaires

Projection : $\pi_{a,b,c...}$
π est l la projection, elle consiste en la sélection des champs.

- $\pi_{title,year}(T1)$
-

```
1 SELECT title, year FROM T1;
```

T1

MID	TITLE	YEAR
2	Ariel	1988
5	Star Wars	1977
3	Shadow in Paradise	1986
7	Dancer in the Dark	2000
80	Nausicaä	1984
71	Billy Elliot	2000

Sélection : $\sigma_{t.c=x,...}$
σ est la sélection, elle sélectionne les enregistrements à partir d'une condition.

- $\sigma_{year \geq 1986}(T1)$
-

```
1 SELECT * FROM T1 WHERE year >= 1986
```

T1

MID	TITLE	YEAR
2	Ariel	1988
5	Star Wars	1977
3	Shadow in Paradise	1986
7	Dancer in the Dark	2000
80	Nausicaä	1984
71	Billy Elliot	2000

Les opérateurs ensemblistes

- Union ($\cup$) : $R \cup S$ retourne une instance de relation contenant les tuples de R ou de S (non dupliqué)
- Intersection ($\cap$) : $R \cap S$ retourne une instance de relation contenant les tuples présents dans R et S
- Différence ($-$) : $R - S$ retourne une instance de relation contenant les tuples présents dans R non présents dans S
- Produit cartésien ($\times$) : $R \times S$ retourne une instance de relation contenant tous les champs de R suivient par les champs de S. Le résultats contenant tous les couples (r, s) pour $\forall r \in R$ et $\forall s \in S$.

Algèbre relationnel (rappel ou pas) : Le produit cartésien

Produit cartésien : $\times$

$R \times S$ retourne une instance de relation contenant tous les champs de R suivient par les champs de S. Le résultats contenant tous les couples (r, s) pour $\forall r \in R$ et $\forall s \in S$.

- $T1 \times T2$
-

```
1 SELECT * FROM T1 CROSS JOIN T2
```

T1

MID	TITLE	YEAR
2	Ariel	1988
5	Star Wars	1977
3	Shadow in Paradise	1986

T2

MID	LANGUAGE	RATING
2	FI	4
5	EN	3.5

T2.MID	TITLE	YEAR	T1.MID	LANGUAGE	RATING
2	Ariel	1988	2	FI	4
2	Ariel	1988	5	EN	3.5
5	Star Wars	1977	2	FI	4
5	Star Wars	1977	5	EN	3.5
3	Shadow in Paradise	1986	2	FI	4
3	Shadow in Paradise	1986	5	EN	3.5

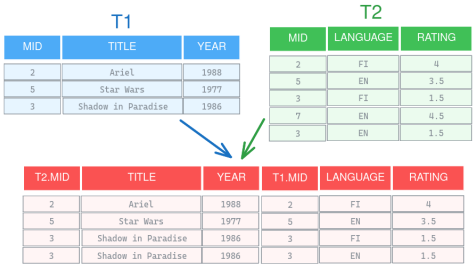
Algèbre relationnel (rappel ou pas) : La jointure

La jointure : $\bowtie_{a=b}$

$\bowtie$ est la jointure elle peut-être écrite comme le produit cartésien associé à une condition entre deux relations.

- $T1 \bowtie_{T1.MID=T2.MID} T2$
-

```
1 SELECT * FROM T1 INNER JOIN T2 ON
      T1.MID=T2.MID
```



Opérateurs relationnels : Évaluation

Comment évaluer/implémenter ces opérateurs ?
Les opérateurs unaires

- **Projection** : parcours
- **Sélection** : parcours ou index
  → Comment sélectionner la méthode ? (sélectivité et histogrammes)

Les opérateurs binaire Parcours minimisant l'ouverture de pages..

- **Union** : parcours
- **Différence** : parcours
- **Produit cartésien** : parcours (couteux!!!)
- **Jointure** : (couteux !!)
  → Comment faire la jointure ? (par itération, par partitionement)

Opérateurs relationnels : Évaluation, la Sélection

Comment évaluer/implémenter la sélection ?

- Par un parcours séquentiel

Opérateurs relationnels : Évaluation, la Sélection

Comment évaluer/implémenter la sélection ?

- Par un parcours séquentiel
- Par l'utilisation d'un index (si disponible)

Opérateurs relationnels : Évaluation, la Sélection

Comment évaluer/implémenter la sélection ?

- Par un parcours séquentiel
- Par l'utilisation d'un index (si disponible)

Mais... *!!! Attention*
L'utilisation d'un index peut-être plus couteux qu'un parcours !!!

Opérateurs relationnels : Évaluation, la Sélection

Comment évaluer/implémenter la sélection ?

- Par un parcours séquentiel
- Par l'utilisation d'un index (si disponible)

Mais... *!!! Attention*
L'utilisation d'un index peut-être plus couteux qu'un parcours !!!

La sélectivité :
La sélectivité d'une condition correspond au nombre d'enregistrements/clefs qui satisferons la condition.

- Si sélectivité grande ? Préférence pour le parcours
- Si sélectivité faible ? Préférence pour l'utilisation d'index (si disponible)

Opérateurs relationnels : Sélectivité

Hypothèse : répartition uniforme des données

Une colonne c et x une valeur on cherche $\sigma_{c=x}(R)$

- T le nombre d'enregistrements
- $V_{min} = 1927$ et $v_{max} = 2000$

$$SEL(\sigma_{c=x}(R)) = \frac{T}{v_{max} - v_{min}}$$

Exemple :
$\sigma_{T1.date=1927}(R)$

- $T = 13$
- $v_{max} = 2000$
- $v_{min} = 1927$

$SEL(\sigma_{T1.date=1927}(R)) = \frac{13}{73} = 0.17$
On estime donc qu'il y aura 0.17 éléments correspondants

1927
1964
1968
1976
1976
1984
1984
1988
1996
1996
2000
2000
2000

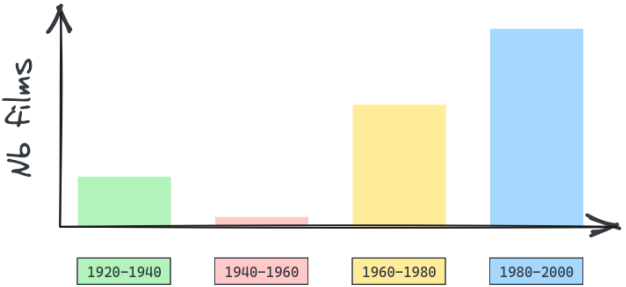
Autres solutions ?

Maintenir des distributions

Autres solutions ?

Maintenir des distributions

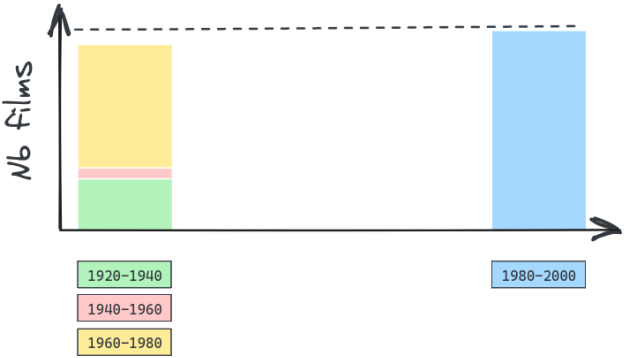
- **Le nombre d'éléments par intervalle (régulier)**



Autres solutions ?

Maintenir des distributions

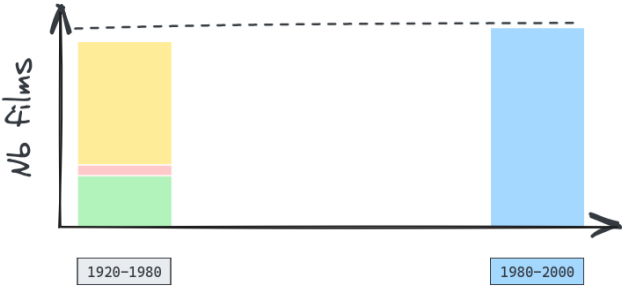
- Le nombre d'éléments par intervalle (régulier)
- **Des intervalles avec le même nombre d'éléments (approximativement)**



Autres solutions ?

Maintenir des distributions

- Le nombre d'éléments par intervalle (régulier)
- **Des intervalles avec le même nombre d'éléments (approximativement)**



Dans PostgreSQL ?
Les statistiques sont enregistrés pour mesurer la sélectivité dans la table *pg_stats* :

- **most_common_vals** : Les 100 valeurs les plus fréquentes
- **most_common_freqs** : La fréquence des 100 valeurs les plus fréquentes
- **histogram_bounds** : Un histogramme découpant les valeurs en ensembles de mêmes tailles
- etc...

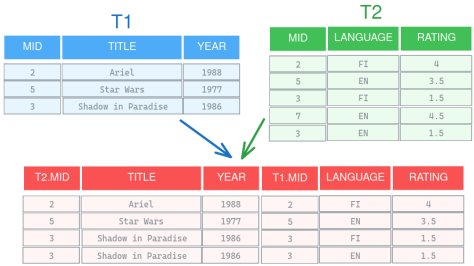
Évaluation de la jointure

Évaluation de la jointure

La jointure : $\bowtie_{a=b}$
$\bowtie$ est la jointure elle peut-être écrite comme le produit cartésien associé à une condition entre deux relations.

- $T1 \bowtie_{T1.MID=T2.MID} T2$
-

```
1 SELECT * FROM T1 INNER JOIN T2 ON
      T1.MID=T2.MID
```



Comment évaluer l'opération de jointure ?

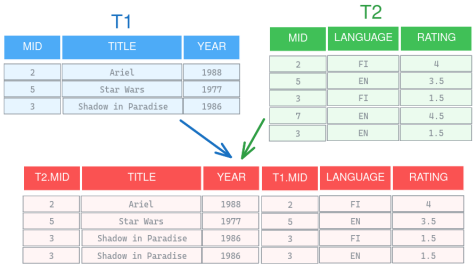
Évaluation de la jointure

La jointure : $\bowtie_{a=b}$

$\bowtie$ est la jointure elle peut-être écrite comme le produit cartésien associé à une condition entre deux relations.

- $T1 \bowtie_{T1.MID=T2.MID} T2$
-

```
1 SELECT * FROM T1 INNER JOIN T2 ON
      T1.MID=T2.MID
```



Comment évaluer l'opération de jointure ?

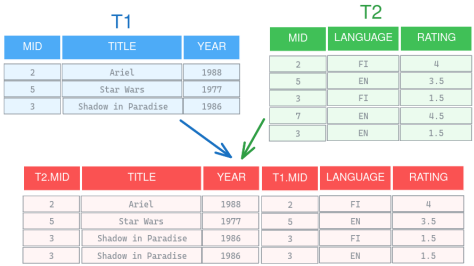
- Famille d'algorithmes par itération (parcourir les tables de jointure)

Évaluation de la jointure

La jointure : $\bowtie_{a=b}$
$\bowtie$ est la jointure elle peut-être écrite comme le produit cartésien associé à une condition entre deux relations.

- $T1 \bowtie_{T1.MID=T2.MID} T2$
-

```
1 SELECT * FROM T1 INNER JOIN T2 ON
      T1.MID=T2.MID
```



Comment évaluer l'opération de jointure ?

- Famille d'algorithmes par itération (parcourir les tables de jointure)
- Famille d'algorithmes par partitionnement (retrouver les parties à joindre)

Évaluation de la jointure : itératif ou partitionement

Les algorithmes itératifs
Itérer sur les deux relations (**loop**)

Évaluation de la jointure : itératif ou partitionement

Les algorithmes itératifs
Itérer sur les deux relations (**loop**)

- Itération par enregistrement
- Itération par page
- Itération par paquet de pages (quand taille du buffer faible)
- Itération avec Index

Évaluation de la jointure : itératif ou partitionement

Les algorithmes itératifs
Itérer sur les deux relations (**loop**)

- Itération par enregistrement
- Itération par page
- Itération par paquet de pages (quand taille du buffer faible)
- Itération avec Index

Les algorithmes par partitionnement
En deux étapes pré-traitement (construction des partitions) puis traitement

Évaluation de la jointure : itératif ou partitionement

Les algorithmes itératifs
Itérer sur les deux relations (**loop**)

- Itération par enregistrement
- Itération par page
- Itération par paquet de pages (quand taille du buffer faible)
- Itération avec Index

Les algorithmes par partitionnement
En deux étapes pré-traitement (construction des partitions) puis traitement

- Par tri (parcours des deux relations par tri)
- Par Hachage

Les jointures ($\bowtie$) : Un exemple

- **Schémas :** T1(MID, TITLE) T2(MID, DATE)
- **Requête :**
  SELECT * FROM T1 INNER JOIN T2
  ON T1.MID=T2.MID
  $T1 \bowtie_{T1.MID=T2.MID} T2$
- **Estimation du coût** (On s'intéresse au transfert disque buffer !!!)

Relation	taille enr	$\|page\|$	$\|fichier\|$
T1	n octets = 10	$P_{T1} = 50$	$F_{T1} = 1000$
T2	m octets = 20	$P_{T2} = 25$	$F_{T2} = 500$

Les jointures ($\bowtie$) : Un exemple

- **Schémas :** T1(MID, TITLE) T2(MID, DATE)
- **Requête :**
  SELECT * FROM T1 INNER JOIN T2
  ON T1.MID=T2.MID
  $T1 \bowtie_{T1.MID=T2.MID} T2$
- **Estimation du coût** (On s'interresses au transfert disque buffer !!!)

Relation	taille enr	$\|page\|$	$\|fichier\|$
T1	n octets = 10	$P_{T1} = 50$	$F_{T1} = 1000$
T2	m octets = 20	$P_{T2} = 25$	$F_{T2} = 500$

Via le produit cartésien ?

- **Idée :** $T1 \times T2$ puis $\sigma_{T1.MID=T2.MID}$
- **Pas très efficaces :**
  - $T1 \times T2 \implies F_{T1}P_{T1} \times F_{T2}P_{T2}$ enregistrements
  - Stockage du résultat intermédiaire
  - Stockage d'une "Table intermédiaire" de $25Go$
  - 125 millions de pages ouvertes

Les jointures ($\bowtie$) : Un exemple

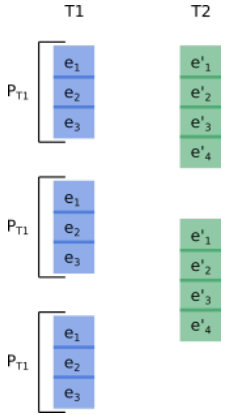
- **Schémas :** T1(MID, TITLE) T2(MID, DATE)
- **Requête :**
  SELECT * FROM T1 INNER JOIN T2
  ON T1.MID=T2.MID
  $T1 \bowtie_{T1.MID=T2.MID} T2$
- **Estimation du coût** (On s'interresses au transfert disque buffer !!!)

Relation	taille enr	$\|page\|$	$\|fichier\|$
T1	n octets = 10	$P_{T1} = 50$	$F_{T1} = 1000$
T2	m octets = 20	$P_{T2} = 25$	$F_{T2} = 500$

Algorithme itératif

- **Idée :**
  Pour tout n-uplet t de T1: ← Itération externe
  Pour tout n-uplet t' de T2:
  Si t[A] = t'[A]:
  res.add($t \bowtie t'$)
- **Pas très efficaces :**
  - Stockage des enregistrements nécessaires
  - Ouverture de toutes les pages plusieurs fois

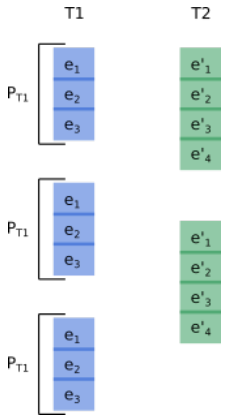
Jointures : Parcours des enregistrements



Chargement des pages

- P_{T1}^1

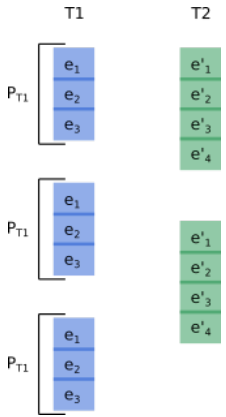
Jointures : Parcours des enregistrements



Chargement des pages

- P^1_{T1}
  - $e_1 \rightarrow$ chargements de toute les pages de T2

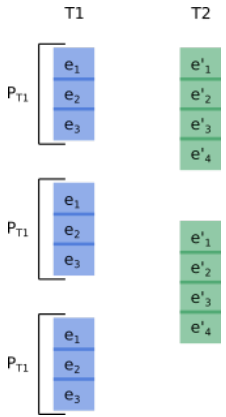
Jointures : Parcours des enregistrements



Chargement des pages

- P^1_{T1}
  - $e_1 \rightarrow$ chargements de toute les pages de T2
  - $e_2 \rightarrow$ chargements de toute les pages de T2

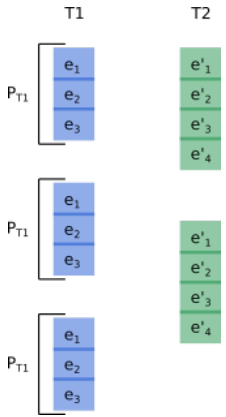
Jointures : Parcours des enregistrements



Chargement des pages

- P^1_{T1}
  - $e_1 \rightarrow$ chargements de toute les pages de T2
  - $e_2 \rightarrow$ chargements de toute les pages de T2
  - $e_3 \rightarrow$ chargements de toute les pages de T2

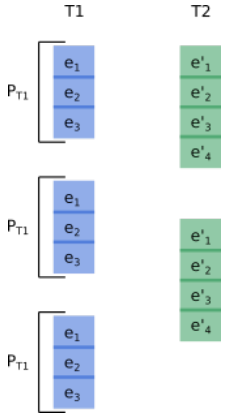
Jointures : Parcours des enregistrements



Chargement des pages

- P^1_{T1}
  - $e_1 \rightarrow$ chargements de toute les pages de T2
  - $e_2 \rightarrow$ chargements de toute les pages de T2
  - $e_3 \rightarrow$ chargements de toute les pages de T2
- P^2_{T1}

Jointures : Parcours des enregistrements



Chargement des pages

- P^1_{T1}
  - $e_1 \rightarrow$ chargements de toute les pages de T2
  - $e_2 \rightarrow$ chargements de toute les pages de T2
  - $e_3 \rightarrow$ chargements de toute les pages de T2
- P^2_{T1}
  - $e_1 \rightarrow$ chargements de toute les pages de T2
  - $e_2 \rightarrow$ chargements de toute les pages de T2
  - $e_3 \rightarrow$ chargements de toute les pages de T2
- ...

Chargement pour chaque enregistrement de T1 de Tous les éléments de T2 !!!

Jointures : Parcours des enregistrements

Combien d'ouverture de pages ?

- On demande pour chaque enregistrement de $T1$ une ouverture de page ($F_{T1}P_{T1}$)
  On va considèrer que l'on garde la page courante en mémoire $\rightarrow F_{T1}$
- Pour chaque enregistrement de T_1 on ouvre F_{T2} pages

Donc $F_{T1} + P_{T1} \times F_{T1} \times F_{T2}$ demande de pages !!!

Coût sur l'exemple
$\rightarrow 1000 + 1000 \times 50 \times 500 \approx 25M$ pages ouvertes

Combien d'ouverture de pages ?

- On demande pour chaque enregistrement de $T1$ une ouverture de page ($F_{T1}P_{T1}$)
  On va considèrer que l'on garde la page courante en mémoire $\rightarrow F_{T1}$
- Pour chaque enregistrement de T_1 on ouvre F_{T2} pages

Donc $F_{T1} + P_{T1} \times F_{T1} \times F_{T2}$ demande de pages !!!

Coût sur l'exemple
$\rightarrow 1000 + 1000 \times 50 \times 500 \approx 25M$ pages ouvertes
Optimisation possible ?

Jointures : Parcours des enregistrements

Combien d'ouverture de pages ?

- On demande pour chaque enregistrement de $T1$ une ouverture de page ($F_{T1}P_{T1}$)
  On va considèrer que l'on garde la page courante en mémoire $\rightarrow F_{T1}$
- Pour chaque enregistrement de T_1 on ouvre F_{T2} pages

Donc $F_{T1} + P_{T1} \times F_{T1} \times F_{T2}$ demande de pages !!!

Coût sur l'exemple
$\rightarrow 1000 + 1000 \times 50 \times 500 \approx 25M$ pages ouvertes

Optimisation possible ?

Si l'on parcours dans la boucle externe $T2$ au lieu de $T1$?

Combien d'ouverture de pages ?

- On demande pour chaque enregistrement de $T1$ une ouverture de page ($F_{T1}P_{T1}$)
  On va considèrer que l'on garde la page courante en mémoire $\rightarrow F_{T1}$
- Pour chaque enregistrement de T_1 on ouvre F_{T2} pages

Donc $F_{T1} + P_{T1} \times F_{T1} \times F_{T2}$ demande de pages !!!

Coût sur l'exemple
$\rightarrow 1000 + 1000 \times 50 \times 500 \approx 25M$ pages ouvertes

Optimisation possible ?

Si l'on parcours dans la boucle externe $T2$ au lieu de $T1$? $\rightarrow 500 + 500 \times 25 \times 1000 \approx 10M$

$\longrightarrow$ La plus petite relation en externe

Les jointures ($\bowtie$) : Un exemple

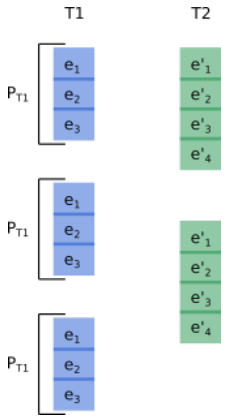
- **Schémas :** T1(MID, TITLE) T2(MID, DATE)
- **Requête :**
  SELECT * FROM T1 INNER JOIN T2
  ON T1.MID=T2.MID
  $T1 \bowtie_{T1.MID=T2.MID} T2$
- **Estimation du coût** (On s'intéresse au transfert disque buffer !!!)

Relation	taille enr	—page—	—fichier—
T1	n octets = 10	$P_{T1} = 50$	$F_{T1} = 1000$
T2	m octets = 20	$P_{T2} = 25$	$F_{T2} = 500$

Algorithme itératif (par page)

- **Idée :**
  Pour toutes les pages p de T1: $\leftarrow$ Itération externe
  Pour toute les pages p' de T2:
  res.add($p \bowtie_{p.mid=p'.mid} p'$)
- Pour chaque page de T1 on ouvre une seule fois chaque page de T2:
  - Stockage des enregistrements nécessaires
  - On test toutes les égalités des enregistrements entre eux

Jointures : Page à Page



Focus transfert : Quand une page de $T1$ et une page de $T2$ dont dans le buffer $\longrightarrow$ trater "tout" les couples avant de passer à la page suivante. Chargement des pages

- P^1_{T1}
  - $e_1, e_2, e_3 \cdots \rightarrow$ chargements de toute les pages de T2
  - $e_4, e_5, e_6 \cdots \rightarrow$ chargements de toute les pages de T2
- ...

Chargement pour chaque page de T1 de tous les éléments de T2 !!!

Jointures : Exemple

Combien d'ouverture de pages ?

- On demande une fois toutes les pages de T_1
- Pour chaque page de T_1 on ouvre F_{T2} pages

Donc $F_{T1} + F_{T2} \times F_{T1}$ demande de pages !!!

Coût sur l'exemple
$\rightarrow 1000 + 1000 \times 500 = 500K + 1000$ pages ouvertes

Jointures : Exemple

Combien d'ouverture de pages ?

- On demande une fois toutes les pages de T_1
- Pour chaque page de T_1 on ouvre F_{T2} pages

Donc $F_{T1} + F_{T2} \times F_{T1}$ demande de pages !!!

Coût sur l'exemple
$\rightarrow 1000 + 1000 \times 500 = 500K + 1000$ pages ouvertes

Optimisation possible ?

$\rightarrow 500 + 500 \times 1000 = 500K + 500$

$\longrightarrow$ La plus petite relation en externe (mais faible gain)

Jointures : Importance de la taille du buffer

Taille du buffer supérieur à T1

Si la taille du buffer est supérieurs à T1 ou T2 alors on charge T1 dans le buffer puis :

Pour toute les pages p' de T2:

res.add($T1 \bowtie_{T1.mid=p'.mid} p'$)

Jointures : Importance de la taille du buffer

Taille du buffer supérieur à T1
Si la taille du buffer est supérieurs à T1 ou T2 alors on charge T1 dans le buffer puis :

Pour toute les pages p' de T2:

res.add($T1 \bowtie_{T1.mid=p'.mid} p'$)

Nombre d'ouverture de pages

- Un parcours de T1 $\longleftarrow$ Stockage dans le buffer de toutes les pages

Jointures : Importance de la taille du buffer

Taille du buffer supérieur à T1
Si la taille du buffer est supérieurs à T1 ou T2 alors on charge T1 dans le buffer puis :

Pour toute les pages p' de T2:

res.add($T1 \bowtie_{T1.mid=p'.mid} p'$)

Nombre d'ouverture de pages

- Un parcours de T1 $\longleftarrow$ Stockage dans le buffer de toutes les pages
- Un parcours de T2 $\longleftarrow$ par page

Jointures : Importance de la taille du buffer

Taille du buffer supérieur à T1
Si la taille du buffer est supérieurs à T1 ou T2 alors on charge T1 dans le buffer puis :

Pour toute les pages p' de T2:

res.add($T1 \bowtie_{T1.mid=p'.mid} p'$)

Nombre d'ouverture de pages

- Un parcours de T1 $\longleftarrow$ Stockage dans le buffer de toutes les pages
- Un parcours de T2 $\longleftarrow$ par page

Donc $F_{T1} + F_{T2}$ (E/S)
$1000 + 500 = 1500$ (E/S) dans l'exemple

Taille du buffer
$F_{T1} + p' + res = F_{T1} + 2$

Jointures : Discussion sur l'utilisation du buffer

Buffer

- Si le buffer peut contenir toute les pages de $T1$ et $T2$ alors cas optimale pour l'approche itérative
- ~~Si le buffer plus petit que le nombre contenu dans l'un des deux fichier alors $F_{T2} \times F_{T1}$ pages ouvertes~~

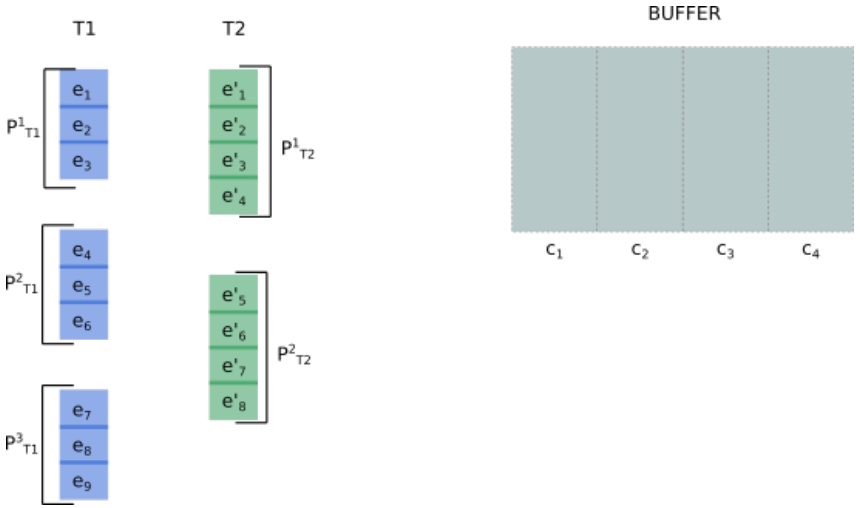
Utilisation des paquets de pages

Pour un paquet de page $P1$ de T1:

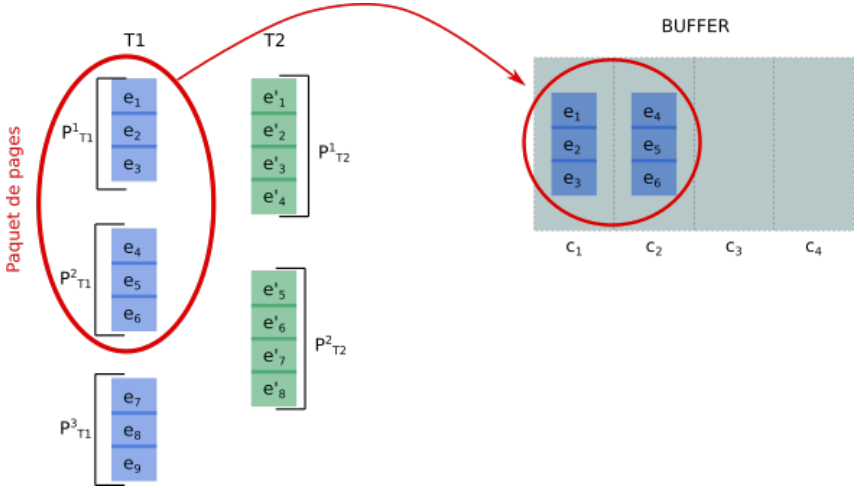
Pour toute les pages p' de T2:

res.add($P1 \bowtie_{P1.mid=p'.mid} p'$)

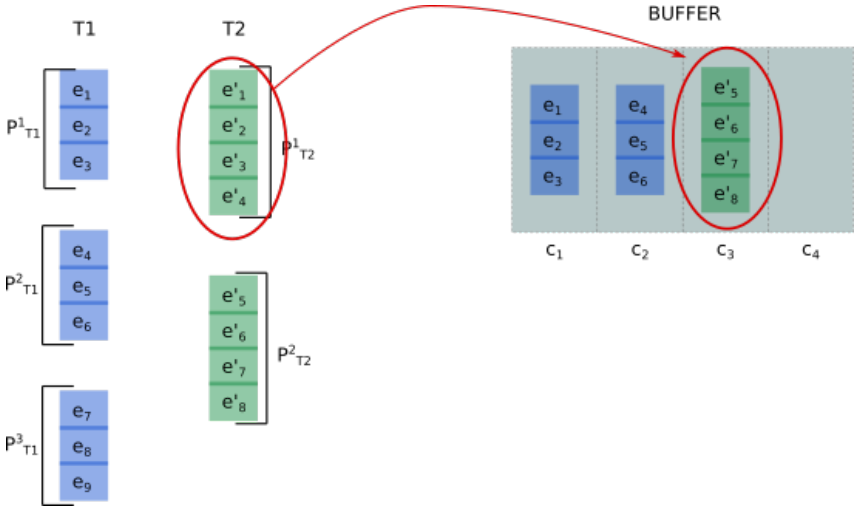
Jointures : par paquet de pages



Jointures : par paquet de pages



Jointures : par paquet de pages



Jointures : Discussion sur l'utilisation du buffer

Buffer

- Si le buffer peut contenir toute les pages de $T1$ et $T2$ alors cas optimale pour l'approche itérative
- ~~Si le buffer plus petit que le nombre contenu dans l'un des deux fichier alors $F_{T2} \times F_{T1}$ pages ouvertes~~

Coût ?

Utilisation des paquets de pages

Pour un paquet de page $P1$ de T1:

Pour toute les pages p' de T2:

res.add($P1 \bowtie_{P1.mid=p'.mid} p'$)

Jointures : Discussion sur l'utilisation du buffer

Buffer

- Si le buffer peut contenir toute les pages de $T1$ et $T2$ alors cas optimale pour l'approche itérative
- ~~Si le buffer plus petit que le nombre contenu dans l'un des deux fichier alors $F_{T2} \times F_{T1}$ pages ouvertes~~

Utilisation des paquets de pages
Pour un paquet de page $P1$ de T1:

Pour toute les pages p' de T2:

res.add($P1 \bowtie_{P1.mid=p'.mid} p'$)

Coût ?

- $\times F_{T1} + \frac{F_{T1}}{k} \times F_{T2}$ (E/S)
- Si $k = 10 \rightarrow 1000 + 100 \times 500 = 501000$ (E/S) Optimisation possible ?
  $\longrightarrow$ La plus petite relation en externe

Taille du buffer ?
Pour des paquets de tailles $k \rightarrow k + 2$ cellules pour le buffer

Utilisation de la sélection

Si e_i un enregistrement de la table T_1 et que $e_i.mid = v_{mid}$, alors, les enregistrements de T_2 joignable avec e_i sont tous les enregistrements de T_2 tel que $\sigma_{mid=v_{mid}}(T_2)$.

Utilisation de la sélection
Si e_i un enregistrement de la table T_1 et que $e_i.mid = v_{mid}$, alors, les enregistrements de T_2 joignable avec e_i sont tous les enregistrements de T_2 tel que $\sigma_{mid=v_{mid}}(T_2)$.

$\longrightarrow \sigma_{mid=v_{mid}}(T_2) \rightarrow$ utilisation d'un index sur $T2$ avec la clef de recherche *mid*

Jointures : itératif et index

Utilisation de la sélection

Si e_i un enregistrement de la table T_1 et que $e_i.mid = v_{mid}$, alors, les enregistrements de T_2 joignable avec e_i sont tous les enregistrements de T_2 tel que $\sigma_{mid=v_{mid}}(T_2)$.

$\longrightarrow \sigma_{mid=v_{mid}}(T_2) \rightarrow$ utilisation d'un index sur $T2$ avec la clef de recherche *mid*

Algorithme itératif

Si T_2 a un index sur *mid* I^2_{mid} on peut faire: Pour chaque enregistrement de e_i de T1:

Sélection sur index des $e'_j = \sigma_{mid=e_i.mid}(T_2)$:

res.add($e_i \bowtie e'_j$)

Jointures : itératif et index

Utilisation de la sélection
Si e_i un enregistrement de la table T_1 et que $e_i.mid = v_{mid}$, alors, les enregistrements de T_2 joignable avec e_i sont tous les enregistrements de T_2 tel que $\sigma_{mid=v_{mid}}(T_2)$.

$\longrightarrow \sigma_{mid=v_{mid}}(T_2) \rightarrow$ utilisation d'un index sur $T2$ avec la clef de recherche *mid*

Algorithme itératif
Si T_2 a un index sur *mid* I^2_{mid} on peut faire: Pour chaque enregistrement de e_i de T1:
Sélection sur index des $e'_j = \sigma_{mid=e_i.mid}(T_2)$:
res.add($e_i \bowtie e'_j$)

Coût

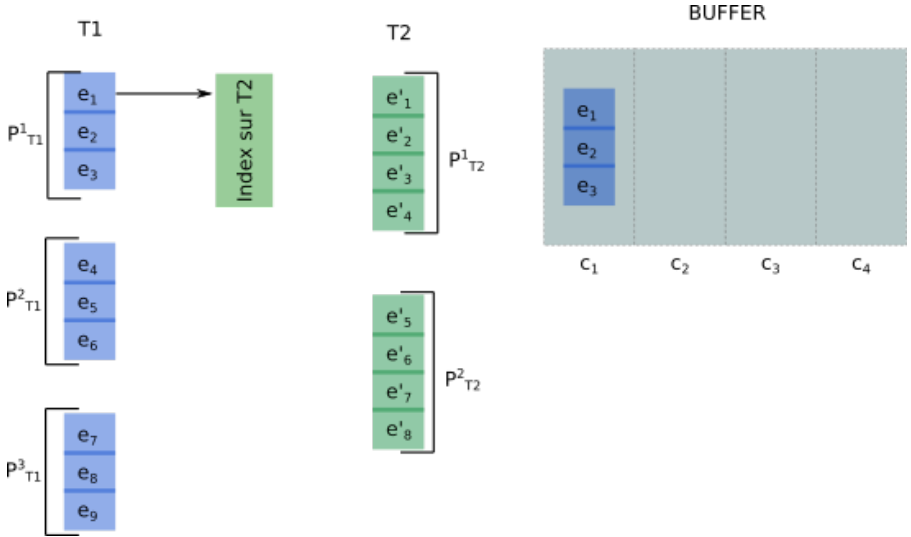
- Parcours de T_1
- Recherche pour chaque enregistrement de T_1

Donc $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S

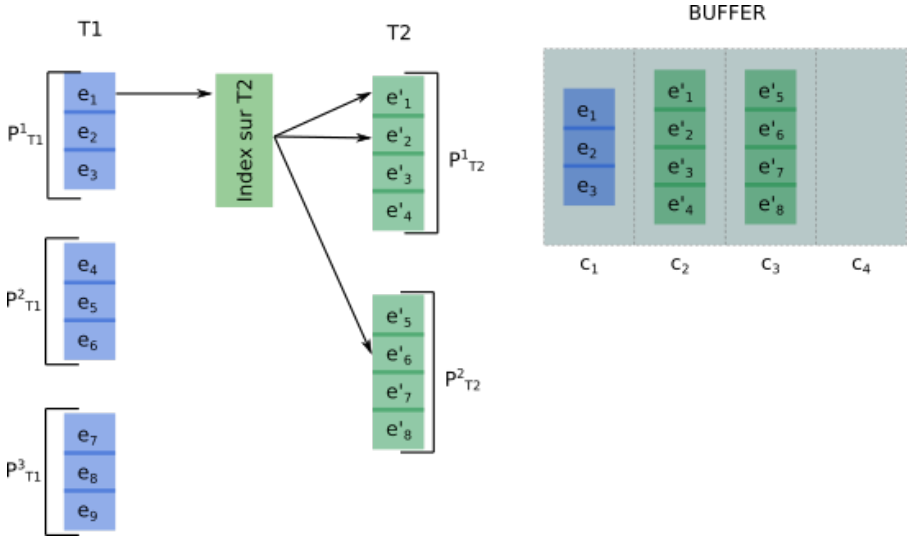
Optimisation possible

$\longrightarrow$ Utilisation de la plus petite relation en externe (ici $T2$)

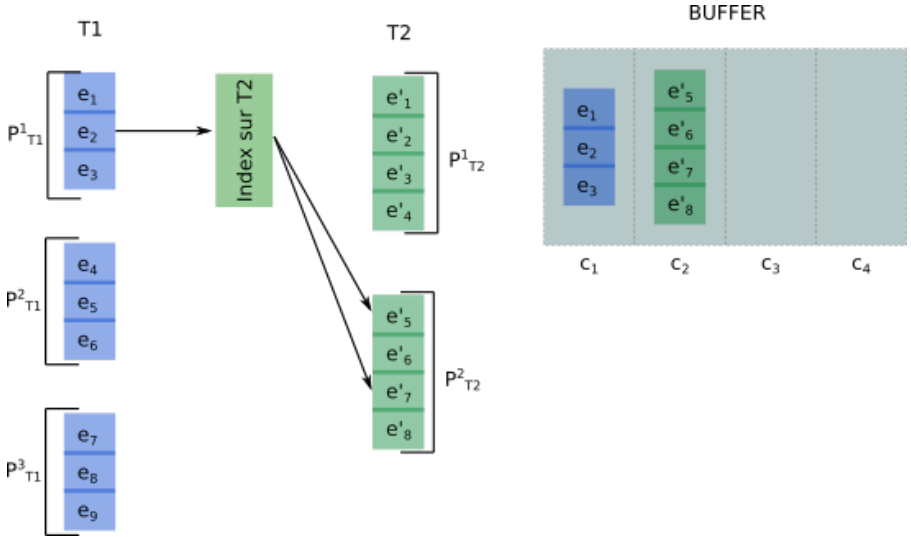
Jointures : itératif et index



Jointures : itératif et index



Jointures : itératif et index



Jointures : itératif et index (Exemples)

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Jointures : itératif et index (Exemples)

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas I: Type 2, non groupant, dense, arbre B+

- h pages de l'arbre (hauteur de l'arbre)

Jointures : itératif et index (Exemples)

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas I: Type 2, non groupant, dense, arbre B+

- h pages de l'arbre (hauteur de l'arbre)
- N_S^{mid} enregistrement ouvert dans les données (possiblement tous sur des pages différentes)

Jointures : itératif et index (Exemples)

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas I: Type 2, non groupant, dense, arbre B+

- h pages de l'arbre (hauteur de l'arbre)
- N_S^{mid} enregistrement ouvert dans les données (possiblement tous sur des pages différentes)

Donc coût de la recherche $h + N_S^{mid}$ et coût de la recherche total $F_{T1} + F_{T1} \times P_{T_1} \times (h + N_S^{mid})$

Jointures : itératif et index (Exemples)

Rel	$\|\|page\|\|$	$\|\|fichier\|\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas II: Type 2, groupant, dense, arbre B+

- h pages de l'arbre (hauteur de l'arbre)

Jointures : itératif et index (Exemples)

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas II: Type 2, groupant, dense, arbre B+

- h pages de l'arbre (hauteur de l'arbre)
- N_S^{mid} enregistrements ouverts en moyennnes, mais **groupant** $\rightarrow \frac{N_S^{mid}}{P_{T2}}$

Jointures : itératif et index (Exemples)

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas II: Type 2, groupant, dense, arbre B+

- h pages de l'arbre (hauteur de l'arbre)
- N_S^{mid} enregistrements ouverts en moyennnes, mais **groupant** $\rightarrow \frac{N_S^{mid}}{P_{T2}}$

Donc coût de la recherche $h + \frac{N_S^{mid}}{P_{T2}}$ et coût de la recherche total $F_{T1} + F_{T1} \times P_{T_1} \times (h + \frac{N_S^{mid}}{P_{T2}})$

Jointures : itératif et index (Exemples)

Rel	$\|\|page\|\|$	$\|\|fichier\|\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas III: Type 2, groupant, dense, arbre B+, mid unique dans $T2$

- h pages de l'arbre (hauteur de l'arbre)

Jointures : itératif et index (Exemples)

Rel	$\|\|page\|\|$	$\|\|fichier\|\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas III: Type 2, groupant, dense, arbre B+, mid unique dans $T2$

- h pages de l'arbre (hauteur de l'arbre)
- 1 enregistrement ouvert au maximum (donc une page)

Jointures : itératif et index (Exemples)

Rel	$\|\|page\|\|$	$\|\|fichier\|\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas III: Type 2, groupant, dense, arbre B+, mid unique dans $T2$

- h pages de l'arbre (hauteur de l'arbre)
- 1 enregistrement ouvert au maximum (donc une page)

Donc coût de la recherche $h + 1$ et coût de la recherche total $F_{T1} + F_{T1} \times P_{T_1} \times (h + 1)$

Jointures : itératif et index (Exemples)

Rel	$\|\|page\|\|$	$\|\|fichier\|\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas IV: Type 2, groupant, dense, Hachage, mid unique dans $T2$

- 1 page de l'index de hachage

Jointures : itératif et index (Exemples)

Rel	$\|\|page\|\|$	$\|\|fichier\|\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas IV: Type 2, groupant, dense, Hachage, mid unique dans $T2$

- 1 page de l'index de hachage
- 1 enregistrement ouvert au maximum (donc une page)

Jointures : itératif et index (Exemples)

Rel	$\|\|page\|\|$	$\|\|fichier\|\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Rappel et notations

- **Coût :** $F_{T1} + F_{T1} \times P_{T_1} \times$ recherches dans S
- **Notation :** N_S^{mid} le nombre d'enregistrement moyen pour une valeur de *mid*, h niveaux de l'arbre

Cas IV: Type 2, groupant, dense, Hachage, mid unique dans $T2$

- 1 page de l'index de hachage
- 1 enregistrement ouvert au maximum (donc une page)

Donc coût de la recherche 2 et coût de la recherche total $F_{T1} + F_{T1} \times P_{T_1} \times 2$

Algorithmes par partitionement

Rassemblement des données cibles

- Par tri (trier les tables puis faire la jointure)
- Par Hachage (hacher les tables puis joindre les buckets)

Deux étapes

1. Pré-traitement → **Build**
2. Traitement →**Probe**

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Jointure par tri-fusion

1. **TraitementPré-traitement (build)** $\rightarrow$ **Tri**
   - Trier les deux relations (T1 & T2)

   $\longrightarrow$ Quel algorithme ?

Jointure par partitionement - Tri & fusion

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Jointure par tri-fusion

1. **TraitementPré-traitement (build) → Tri**
   - Trier les deux relations (T1 & T2)

   ⟶ Quel algorithme ?

   Tri fusion, tri rapide → $nlog(n)$

Jointure par partitionement - Tri & fusion

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Jointure par tri-fusion

1. **TraitementPré-traitement (build) → Tri**
   - Trier les deux relations (T1 & T2)

   ⟶ Quel algorithme ?

   Tri fusion, tri rapide → $nlog(n)$
   - ⟶ Quel coût (pages ouvertes, E/S) ?

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Jointure par tri-fusion

1. **TraitementPré-traitement (build) → Tri**
   - Trier les deux relations (T1 & T2)

   ⟶ Quel algorithme ?

   Tri fusion, tri rapide → $nlog(n)$
   - ⟶ Quel coût (pages ouvertes, E/S) ?

     $2F_{T1}log(F_{T1}) + 2F_{T2}log(F_{T2})$

Jointure par partitionement - Tri & fusion

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Jointure par tri-fusion

1. **TraitementPré-traitement (build) → Tri**
   - Trier les deux relations (T1 & T2)

   ⟶ Quel algorithme ?

   Tri fusion, tri rapide → $nlog(n)$
   - ⟶ Quel coût (pages ouvertes, E/S) ?

   $2F_{T1}log(F_{T1}) + 2F_{T2}log(F_{T2})$
2. **Traitement (Probe) → Fusion**
   - Parcourir les données triées en parallèle

Jointure par partitionement - Tri & fusion

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Jointure par tri-fusion

1. **TraitementPré-traitement (build) → Tri**
   - Trier les deux relations (T1 & T2)

   ⟶ Quel algorithme ?

   Tri fusion, tri rapide → $nlog(n)$
   - ⟶ Quel coût (pages ouvertes, E/S) ?

   $2F_{T1}log(F_{T1}) + 2F_{T2}log(F_{T2})$
2. **Traitement (Probe) → Fusion**
   - Parcourir les données triées en parallèle

   ⟶ Quel coût (pages ouvertes, E/S) ?

Jointure par partitionement - Tri & fusion

Rel	$\|page\|$	$\|fichier\|$
T1	$P_{T1} = 50$	$F_{T1} = 1000$
T2	$P_{T2} = 25$	$F_{T2} = 500$

Jointure par tri-fusion

1. **TraitementPré-traitement (build) → Tri**
   - Trier les deux relations (T1 & T2)

   ⟶ Quel algorithme ?

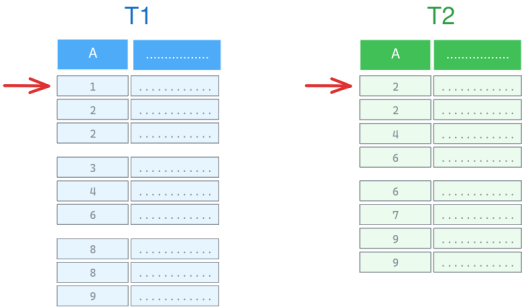
   Tri fusion, tri rapide → $nlog(n)$
   - ⟶ Quel coût (pages ouvertes, E/S) ?

   $2F_{T1}log(F_{T1}) + 2F_{T2}log(F_{T2})$
2. **Traitement (Probe) → Fusion**
   - Parcourir les données triées en parallèle

   ⟶ Quel coût (pages ouvertes, E/S) ?

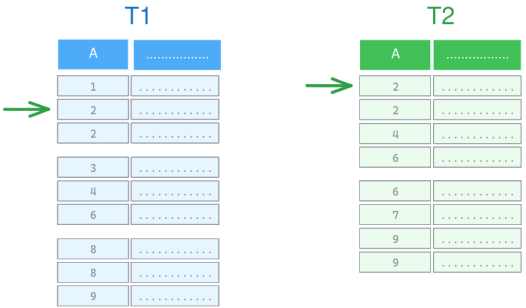
   $F_{T1} + F_{T2}$

Jointure par partitionement - Tri & fusion: la fusion de relations triées



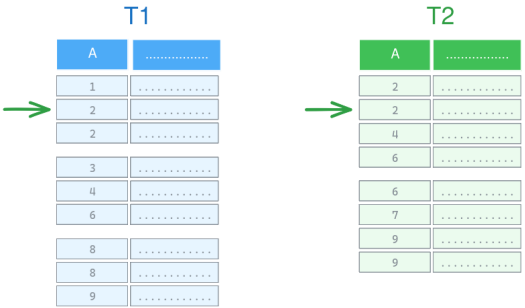
- Pas de jointure ($T_1.A_0 < T_2.A_0$)

Jointure par partitionement - Tri & fusion: la fusion de relations triées



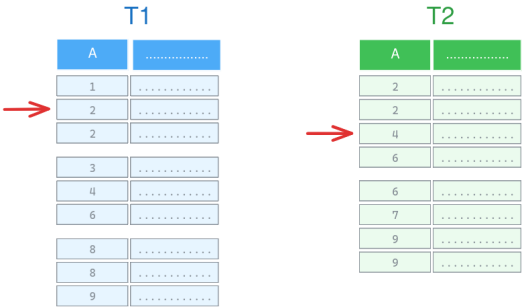
1. Pas de jointure ($T_1.A_0 < T_2.A_0$)
2. jointure pour ($T_1.A_1 = T_2.A_0$)

Jointure par partitionement - Tri & fusion: la fusion de relations triées



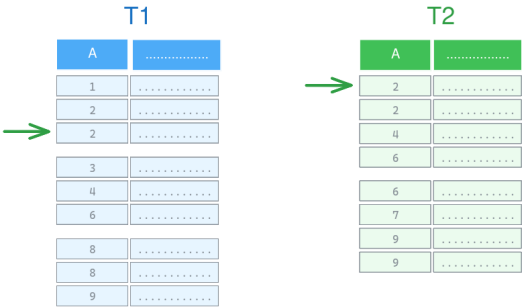
1. Pas de jointure ($T_1.A_0 < T_2.A_0$)
2. jointure pour ($T_1.A_1 = T_2.A_0$)
3. jointure pour ($T_1.A_1 = T_2.A_1$)

Jointure par partitionement - Tri & fusion: la fusion de relations triées



1. Pas de jointure ($T_1.A_0 < T_2.A_0$)
2. jointure pour ($T_1.A_1 = T_2.A_0$)
3. jointure pour ($T_1.A_1 = T_2.A_1$)
4. Pas de jointure ($T_1.A_1 < T_2.A_2$)

Jointure par partitionement - Tri & fusion: la fusion de relations triées

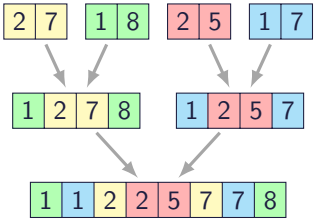


1. Pas de jointure ($T_1.A_0 < T_2.A_0$)
2. jointure pour ($T_1.A_1 = T_2.A_0$)
3. jointure pour ($T_1.A_1 = T_2.A_1$)
4. Pas de jointure ($T_1.A_1 < T_2.A_2$)
5. Jointure pour ($T_1.A_2 = T_2.A_0$)

Le tri fusion
Fusionner des fragments de plus en plus importants

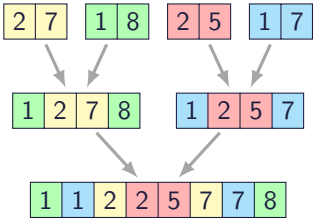
- **Étape 1 :** Prendre les pages deux à deux de $T1$, fusionner chaque couple de pages
  $\rightarrow$ Un fragment (en sortie) correspond à deux pages
- **Étape 2 :** Prendre les précédents fragments $T1$, fusionner chaque couple de fragments
  $\rightarrow$ Un fragment (en sortie) correspond à deux pages
- ...
  Obtention d'un fragment trié sur tous les enregistrements/pages

Data: A, B fragments
Result: L un fragment trié
$i \leftarrow 1$; $j \leftarrow 1$; $L \leftarrow []$;
while $|L| \neq 0$ **do**
  if $j < |B|$ *AND* $A_i > B_j$ **then**
    $L_{i+j} \leftarrow B_j$;
    $j \leftarrow j + 1$;
  else
    $L_{i+j} \leftarrow A_i$;
    $i \leftarrow i + 1$;
  end
end
Algorithm 1: algorithme de fusion de deux fragments



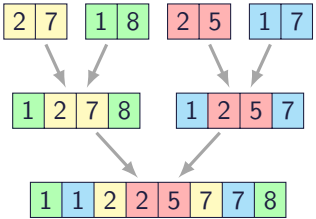
Coût du Tri :

- **Étape 1 :**
  - Prendre deux à deux les pages de $T1$



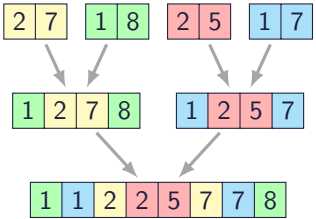
Coût du Tri :

- **Étape 1 :**
  - Prendre deux à deux les pages de $T1 \longrightarrow M$ lectures
  - Écrire les fragments de $T1$ (même taille que $T1$)



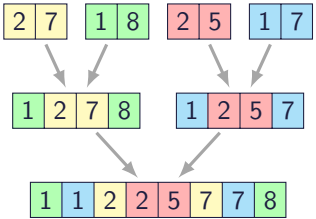
Coût du Tri :

- **Étape 1 :**
  - Prendre deux à deux les pages de $T1 \longrightarrow M$ lectures
  - Écrire les fragments de $T1$ (même taille que $T1) \longrightarrow M$ écritures
- **Étape 2 :**
  - Prendre deux à deux les $\frac{F_{T1}}{2}$ fragments précédents de taille 2 (M pages écrites)



Coût du Tri :

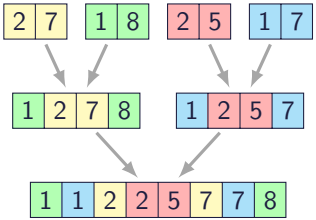
- **Étape 1 :**
  - Prendre deux à deux les pages de $T1 \longrightarrow M$ lectures
  - Écrire les fragments de $T1$ (même taille que $T1) \longrightarrow M$ écritures
- **Étape 2 :**
  - Prendre deux à deux les $\frac{F_{T1}}{2}$ fragments précédents de taille 2 (M pages écrites) $\longrightarrow M$ lectures
  - Écrire les nouveau fragments de taille 4

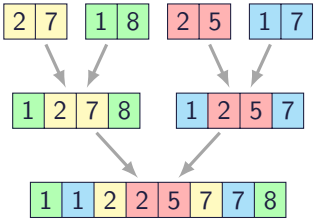


Coût du Tri :

- **Étape 1 :**
  - Prendre deux à deux les pages de $T1 \longrightarrow M$ lectures
  - Écrire les fragments de $T1$ (même taille que $T1) \longrightarrow M$ écritures
- **Étape 2 :**
  - Prendre deux à deux les $\frac{F_{T1}}{2}$ fragments précédents de taille 2 (M pages écrites) $\longrightarrow M$ lectures
  - Écrire les nouveau fragments de taille $4 \longrightarrow M$ écritures

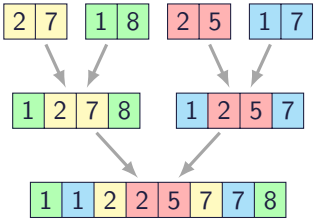
Coût du Tri pour l'étape i





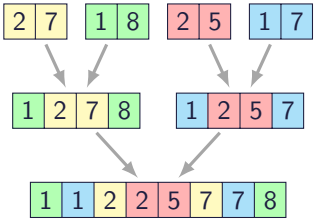
Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{F_{T1}}{2^i}$ fragments précédents de taille $\mathbf{2^i}$



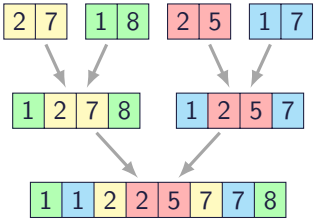
Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{F_{T1}}{2^i}$ fragments précédents de taille $\mathbf{2^i}$
  $\longrightarrow F_{T1}$ lectures



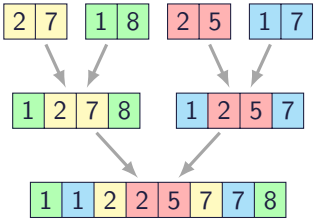
Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{F_{T1}}{2^i}$ fragments précédents de taille $\mathbf{2^i}$
  $\longrightarrow F_{T1}$ lectures
- Écrire les nouveau fragments de taille $\mathbf{2^{i+1}}$



Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{F_{T1}}{2^i}$ fragments précédents de taille $\mathbf{2^i}$
  $\longrightarrow F_{T1}$ lectures
- Écrire les nouveau fragments de taille $\mathbf{2^{i+1}}$
  $\longrightarrow F_{T1}$ écritures



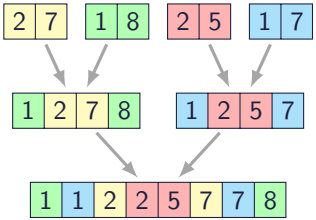
Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{F_{T1}}{2^i}$ fragments précédents de taille $\mathbf{2^i}$
  $\longrightarrow F_{T1}$ lectures
- Écrire les nouveau fragments de taille $\mathbf{2^{i+1}}$
  $\longrightarrow F_{T1}$ écritures

Coût du Tri fusion (toujours pour les E/S)

$$2F_{T1} \times nombre\_étape$$

Jointure par partitionement - Tri & fusion : le tri des relations



Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{F_{T1}}{2^i}$ fragments précédents de taille $\mathbf{2^i}$
  $\longrightarrow F_{T1}$ lectures
- Écrire les nouveau fragments de taille $\mathbf{2^{i+1}}$
  $\longrightarrow F_{T1}$ écritures

Coût du Tri fusion (toujours pour les E/S)

$$2F_{T1} \times nombre\_étape$$

$$\rightarrow 2F_{T1} \times \lceil 1 + log_2(F_{T1}) \rceil$$

Optimisation du tri

- Diminuer le nombre d'étape

Jointure par partitionement - Tri & fusion : Tri multi-fusion

Optimisation du tri

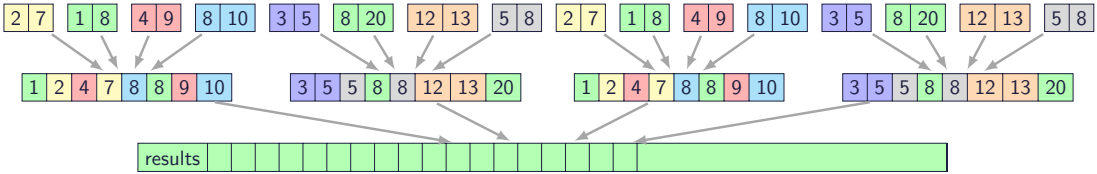
- Diminuer le nombre d'étape

$\rightarrow$ Augmentation du nombre de pages ou de fragments fusionnés (ici k)

Optimisation du tri

- Diminuer le nombre d'étape

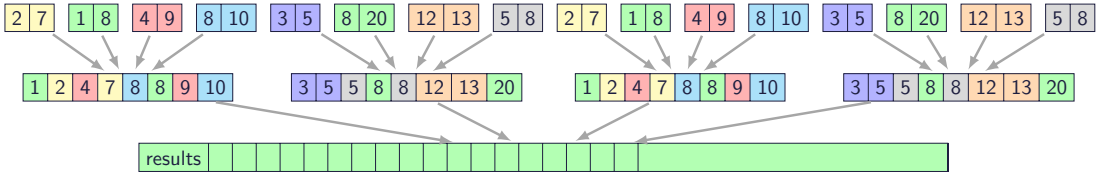
→ Augmentation du nombre de pages ou de fragments fusionnés (ici k)



Optimisation du tri

- Diminuer le nombre d'étape

→ Augmentation du nombre de pages ou de fragments fusionnés (ici k)

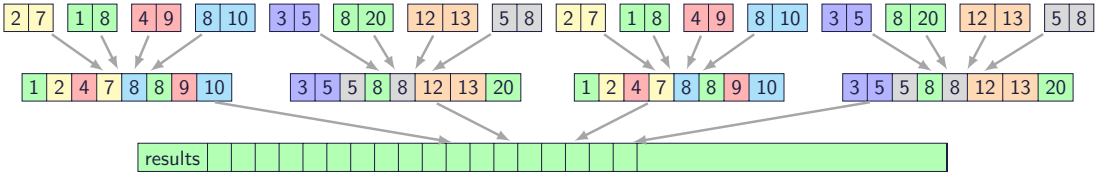


Coût ?

Optimisation du tri

- Diminuer le nombre d'étape

→ Augmentation du nombre de pages ou de fragments fusionnés (ici k)



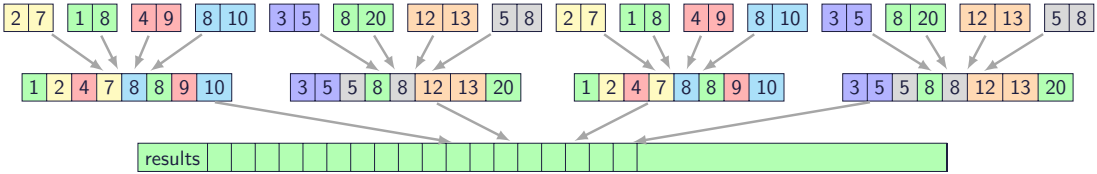
Coût ?

- $\rightarrow 2F_T \times \lceil 1 + log_k(F_T) \rceil$

Optimisation du tri

- Diminuer le nombre d'étape

→ Augmentation du nombre de pages ou de fragments fusionnés (ici k)



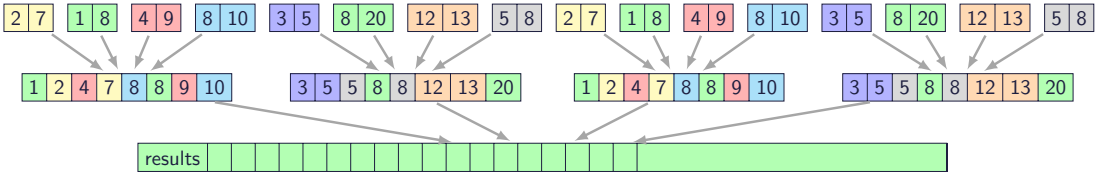
Coût ?

- $\rightarrow 2F_T \times \lceil 1 + log_k(F_T) \rceil$
- **Taille du buffer**

Optimisation du tri

- Diminuer le nombre d'étape

→ Augmentation du nombre de pages ou de fragments fusionnés (ici k)



Coût ?

- $\rightarrow 2F_T \times \lceil 1 + log_k(F_T) \rceil$
- **Taille du buffer** : k + 1

Jointure par partitionement - Hachage

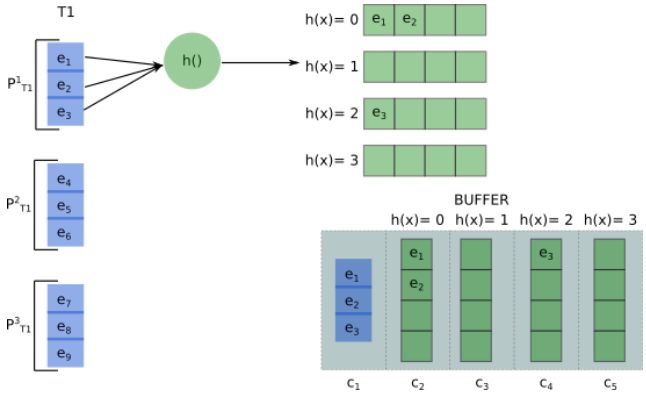
Principe de la jointure par Hachage

1. **build :** Création des tables de hachage pour les deux relations
2. **probe :** Joindre les deux tables par parcours

Jointure par partitionement - Hachage

Principe de la jointure par Hachage

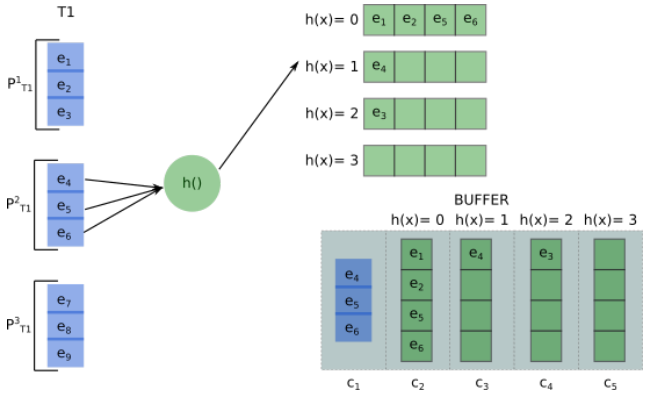
1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours



Jointure par partitionement - Hachage

Principe de la jointure par Hachage

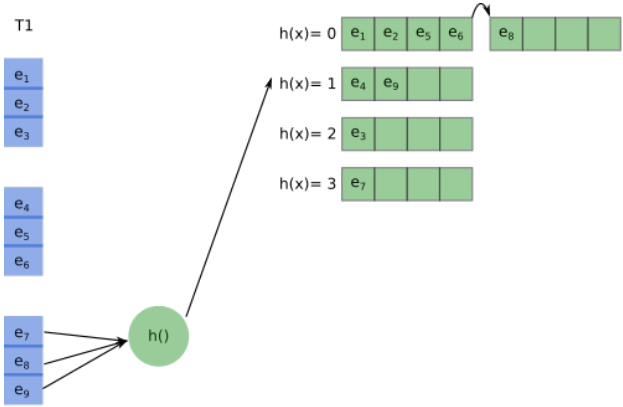
1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours



Jointure par partitionement - Hachage

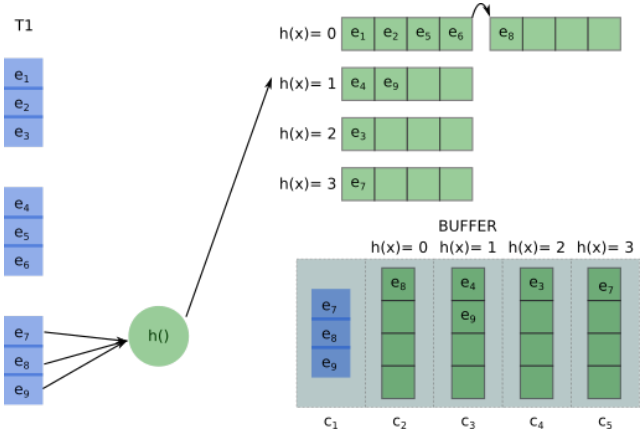
Principe de la jointure par Hachage

1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours



Principe de la jointure par Hachage

1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours



Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours

Coût de la création des tables de hachage

Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours

Coût de la création des tables de hachage

- Parcours de $T1$, 1 lecture/1 écriture de toutes les pages
- Parcours de $T2$: Idem

Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours

Coût de la création des tables de hachage

- Parcours de $T1$, 1 lecture/1 écriture de toutes les pages
- Parcours de $T2$: Idem

Donc $2 \times F_{T1} + 2 \times F_{T2}$ (Mais avoir une taille de buffer suffisante)

Taille du buffer

Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours

Coût de la création des tables de hachage

- Parcours de $T1$, 1 lecture/1 écriture de toutes les pages
- Parcours de $T2$: Idem

Donc $2 \times F_{T1} + 2 \times F_{T2}$ (Mais avoir une taille de buffer suffisante)

Taille du buffer

- 1 cellule pour stocker la page en lecture

Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours

Coût de la création des tables de hachage

- Parcours de $T1$, 1 lecture/1 écriture de toutes les pages
- Parcours de $T2$: Idem

Donc $2 \times F_{T1} + 2 \times F_{T2}$ (Mais avoir une taille de buffer suffisante)

Taille du buffer

- 1 cellule pour stocker la page en lecture
- 1 cellule par bucket (k)

Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build : Création des tables de hachage pour les deux relations**
2. **probe :** Joindre les deux tables par parcours

Coût de la création des tables de hachage

- Parcours de $T1$, 1 lecture/1 écriture de toutes les pages
- Parcours de $T2$: Idem

Donc $2 \times F_{T1} + 2 \times F_{T2}$ (Mais avoir une taille de buffer suffisante)

Taille du buffer

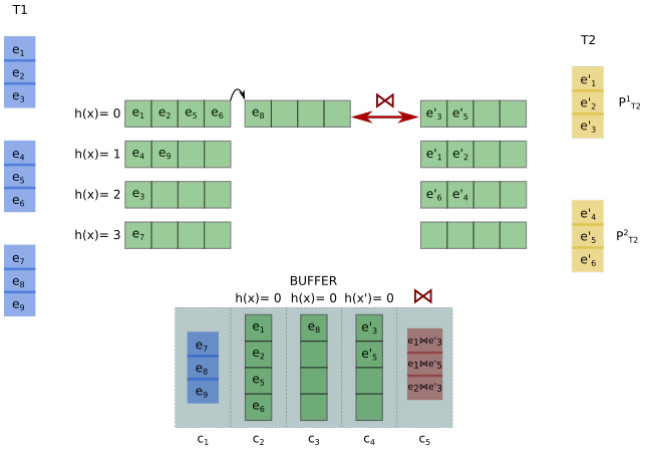
- 1 cellule pour stocker la page en lecture
- 1 cellule par bucket (k)

Donc $K + 1$

Jointure par partitionement - Hachage

Principe de la jointure par Hachage

1. **build :** Création des tables de hachage pour les deux relations
2. **probe : Joindre les deux tables par parcours**



Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build :** des tables de hachage pour les deux relations
2. **probe : Joindre les deux tables par parcours**

Coût de la Fusion-jointure

Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build :** des tables de hachage pour les deux relations
2. **probe : Joindre les deux tables par parcours**

Coût de la Fusion-jointure

Parcours des pages/paquets de $T1$ (dans la table de hachage)
  Parcours des pages $T2$ (correspondant aux pages de $T1$)
    Jointure des deux pages

Jointure par partitionement - Hachage (Coût)

Principe de la jointure par Hachage

1. **build :** des tables de hachage pour les deux relations
2. **probe : Joindre les deux tables par parcours**

Coût de la Fusion-jointure

Parcours des pages/paquets de $T1$ (dans la table de hachage)

Parcours des pages $T2$ (correspondant aux pages de $T1$)

Jointure des deux pages Donc $F_{T1} + F_{T2}$

Taille du buffer

Stockage des paquets de $T1$, d'une page de T2 et de l'espace pour la jointure

Donc buffer $\geq$ *taille_max_paquet* $+ 2$

Jointure par partitionement - Hachage (Coût)

Coût Total
Somme des coût du traitement (**build**) et du traitement (**probe**)

- Coût du hachage (**build**) $2F_{T1} + 2F_{T2}$
- Cout de la fusion (**probe**) $F_{T1} + F_{T2}$

Jointure par partitionement - Hachage (Coût)

Coût Total
Somme des coût du traitement (**build**) et du traitement (**probe**)

- Coût du hachage (**build**) $2F_{T1} + 2F_{T2}$
- Cout de la fusion (**probe**) $F_{T1} + F_{T2}$

$\rightarrow 3F_{T1} + 3F_{T2}$