

Bases de Données & Optimisation : Évaluation des opérateurs relationnels

Thomas Gerald

February 19, 2026

Laboratoire Interdisciplinaire des Sciences du Numérique – LISN, CNRS

thomas.gerald@lisn.upsaclay.fr

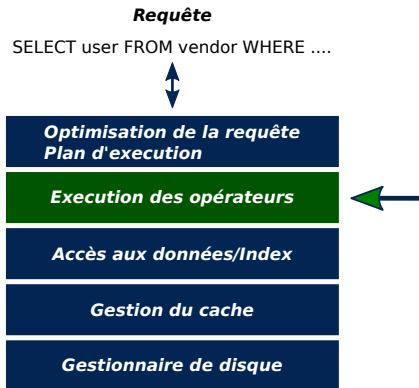
Évaluation d'opérateurs relationnels

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



“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 : $\pi_{a,b,c,\dots}$

π est 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

Algèbre relationnel (rappel ou pas) : Opérateurs unaires

Projection : $\pi_{a,b,c,\dots}$

π est 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 : $\sigma_{t.c=x,\dots}$

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 suivent 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

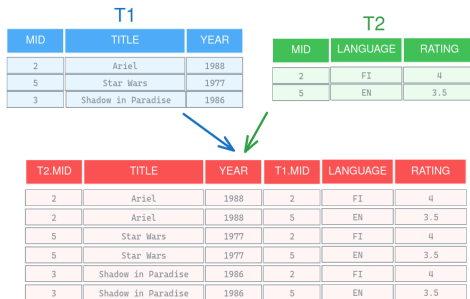
Produit cartésien : $\times$

$R \times S$ retourne une instance de relation contenant tous les champs de R suivent 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
```



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`

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
3	FI	1.5
7	EN	4.5
3	EN	1.5



T2.MID	TITLE	YEAR	T1.MID	LANGUAGE	RATING
2	Ariel	1988	2	FI	4
5	Star Wars	1977	5	EN	3.5
3	Shadow in Paradise	1986	3	FI	1.5
3	Shadow in Paradise	1986	3	EN	1.5

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 partitionnement)

Comment évaluer/implémenter la sélection ?

- Par un parcours séquentiel

Comment évaluer/implémenter la sélection ?

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

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 !!!

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)

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

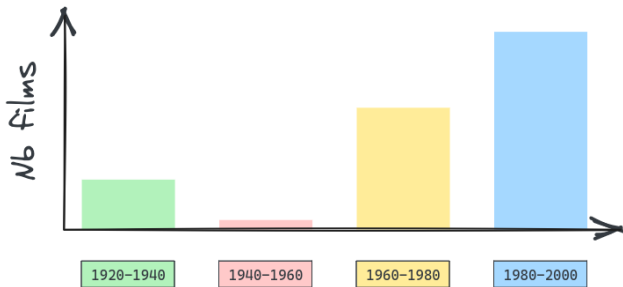
1927
1964
1968
1976
1976
1984
1984
1988
1996
1996
2000
2000
2000

Maintenir des distributions

Autres solutions ?

Maintenir des distributions

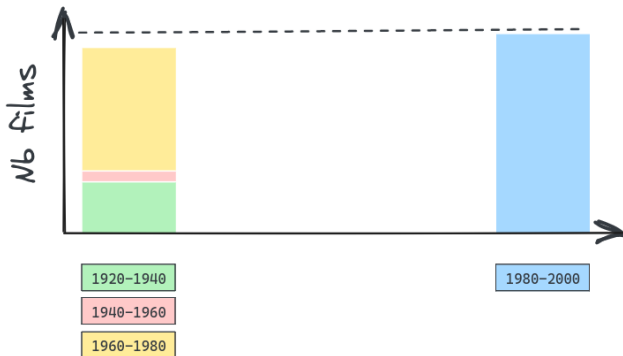
- Le nombre d'éléments par intervalle (régulier)



Autres solutions ?

Maintenir des distributions

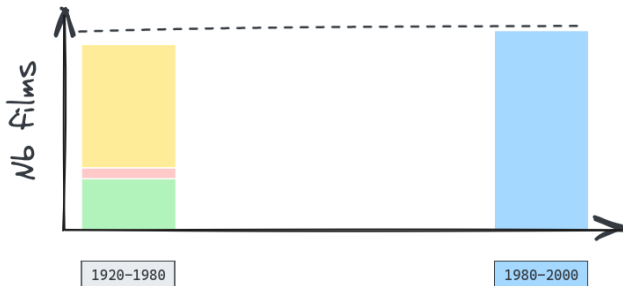
- Le nombre d'éléments par intervalle (régulier)
- Des intervalles avec le même nombre d'éléments (approximativement)



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

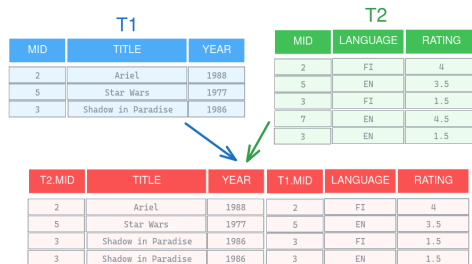
Évaluation de la jointure

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`

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

MID	LANGUAGE	RATING
2	FI	4
5	EN	3.5
3	FI	1.5
7	EN	4.5
3	EN	1.5

T2.MID	TITLE	YEAR	T1.MID	LANGUAGE	RATING
2	Ariel	1988	2	FI	4
5	Star Wars	1977	5	EN	3.5
3	Shadow in Paradise	1986	3	FI	1.5
3	Shadow in Paradise	1986	3	EN	1.5

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`

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

MID	LANGUAGE	RATING
2	FI	4
5	EN	3.5
3	FI	1.5
7	EN	4.5
3	EN	1.5

T2.MID	TITLE	YEAR	T1.MID	LANGUAGE	RATING
2	Ariel	1988	2	FI	4
5	Star Wars	1977	5	EN	3.5
3	Shadow in Paradise	1986	3	FI	1.5
3	Shadow in Paradise	1986	3	EN	1.5

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)

Les algorithmes itératifs

Itérer sur les deux relations (**loop**)

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 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

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	enr/pages	pages
T1	n octets = 10	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	m octets = 20	$S_{T2} = 25$	$ P_{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'intéresse au transfert disque buffer !!!)

Relation	taille enr	enr/pages	pages
T1	n octets = 10	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	m octets = 20	$S_{T2} = 25$	$ P_{T2} = 500$

Via le produit cartésien ?

- **Idée** : $T1 \times T2$ puis $\sigma_{T1.MID=T2.MID}$
- **Pas très efficaces** :
 - $T1 \times T2 \implies |P_{T1}|P_{T1} \times |P_{T2}|P_{T2}$ 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'intéresse au transfert disque buffer !!!)

Relation	taille enr	enr/pages	pages
T1	n octets = 10	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	m octets = 20	$S_{T1} = 25$	$ P_{T2} = 500$

Algorithme itératif

- **Idée** :
Pour tout n-uplet t de T1: $\leftarrow$ **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_{T1}^1
 - $e_1 \rightarrow$ chargements de toute les pages de T2

Jointures : Parcours des enregistrements



Chargement des pages

- P_{T1}^1
 - $e_1 \rightarrow$ 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_{T1}^1
 - $e_1 \rightarrow$ 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_{T1}^1
 - $e_1 \rightarrow$ 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_{T1}^2

Jointures : Parcours des enregistrements



Chargement des pages

- P_{T1}^1
 - e₁ → chargements de toute les pages de T2
 - e₂ → chargements de toute les pages de T2
 - e₃ → chargements de toute les pages de T2
- P_{T1}^2
 - e₁ → chargements de toute les pages de T2
 - e₂ → chargements de toute les pages de T2
 - e₃ → chargements de toute les pages de T2
- ...

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

Combien d'ouverture de pages ?

- On demande pour chaque enregistrement de T_1 une ouverture de page ($|P_{T_1}|S_{T_1}$)
On va considérer que l'on garde la page courante en mémoire $\rightarrow P_{T_1}^i$
- Pour chaque enregistrement de T_1 on ouvre $|P_{T_2}|$ pages

Donc $|P_{T_1}| + S_{T_1} \times |P_{T_1}| \times |P_{T_2}|$ 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 T_1 une ouverture de page ($|P_{T_1}|S_{T_1}$)
On va considérer que l'on garde la page courante en mémoire $\rightarrow P_{T_1}^i$
- Pour chaque enregistrement de T_1 on ouvre $|P_{T_2}|$ pages

Donc $|P_{T_1}| + S_{T_1} \times |P_{T_1}| \times |P_{T_2}|$ demande de pages !!!

Coût sur l'exemple

$\rightarrow 1000 + 1000 \times 50 \times 500 \approx 25M$ pages ouvertes

Optimisation possible ?

Combien d'ouverture de pages ?

- On demande pour chaque enregistrement de T_1 une ouverture de page ($|P_{T_1}|S_{T_1}$)
On va considérer que l'on garde la page courante en mémoire $\rightarrow P_{T_1}^i$
- Pour chaque enregistrement de T_1 on ouvre $|P_{T_2}|$ pages

Donc $|P_{T_1}| + S_{T_1} \times |P_{T_1}| \times |P_{T_2}|$ demande de pages !!!

Coût sur l'exemple

$\rightarrow 1000 + 1000 \times 50 \times 500 \approx 25M$ pages ouvertes

Optimisation possible ?

Si l'on parcourt dans la boucle externe T_2 au lieu de T_1 ?

Combien d'ouverture de pages ?

- On demande pour chaque enregistrement de T_1 une ouverture de page ($|P_{T_1}|S_{T_1}$)
On va considérer que l'on garde la page courante en mémoire $\rightarrow P_{T_1}^i$
- Pour chaque enregistrement de T_1 on ouvre $|P_{T_2}|$ pages

Donc $|P_{T_1}| + S_{T_1} \times |P_{T_1}| \times |P_{T_2}|$ demande de pages !!!

Coût sur l'exemple

$\rightarrow 1000 + 1000 \times 50 \times 500 \approx 25M$ pages ouvertes

Optimisation possible ?

Si l'on parcourt dans la boucle externe T_2 au lieu de T_1 ? $\rightarrow 500 + 500 \times 25 \times 1000 \approx 10M$

$\rightarrow$ 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	enr/pages	pages
T1	n octets = 10	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	m octets = 20	$S_{T2} = 25$	$ P_{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



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

- P_{T1}^1
 - $e_1, e_2, e_3 \dots \rightarrow$ chargements de toutes les pages de $T2$
 - $e_4, e_5, e_6 \dots \rightarrow$ chargements de toutes les pages de $T2$
- ...

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

Combien d'ouverture de pages ?

- On demande une fois toutes les pages de T_1
- Pour chaque page de T_1 on ouvre $|P_{T_2}|$ pages

Donc $|P_{T_1}| + |P_{T_2}| \times |P_{T_1}|$ demande de pages !!!

Coût sur l'exemple

→ $1000 + 1000 \times 500 = 500K + 1000$ pages ouvertes

Combien d'ouverture de pages ?

- On demande une fois toutes les pages de T_1
- Pour chaque page de T_1 on ouvre $|P_{T_2}|$ pages

Donc $|P_{T_1}| + |P_{T_2}| \times |P_{T_1}|$ demande de pages !!!

Coût sur l'exemple

→ $1000 + 1000 \times 500 = 500K + 1000$ pages ouvertes

Optimisation possible ?

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

→ La plus petite relation en externe (mais faible gain)

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:

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

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:

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

Nombre d'ouverture de pages

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

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:

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

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

Donc $|P_{T1}| + |P_{T2}|$ (E/S)

$1000 + 500 = 1500$ (E/S) dans l'exemple

Taille du buffer

$|P_{T1}| + p' + res = |P_{T1}| + 2$

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 $|P_{T2}| \times |P_{T1}|$ pages ouvertes

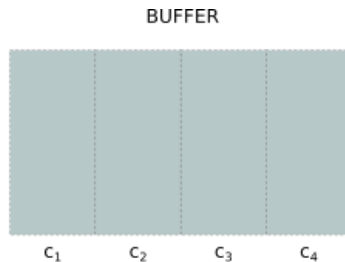
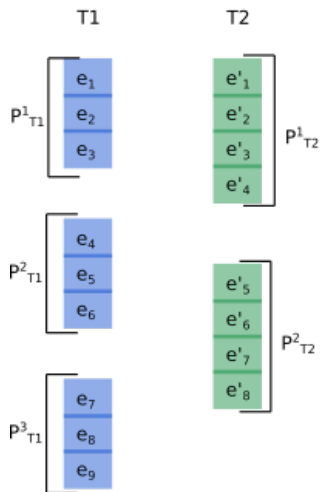
Utilisation des paquets de pages

Pour un paquet de page $P1$ de $T1$:

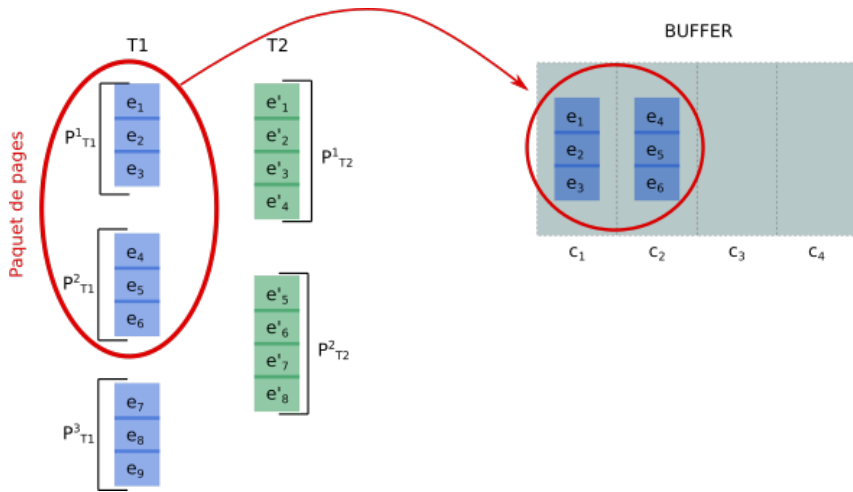
Pour toute les pages p' de $T2$:

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

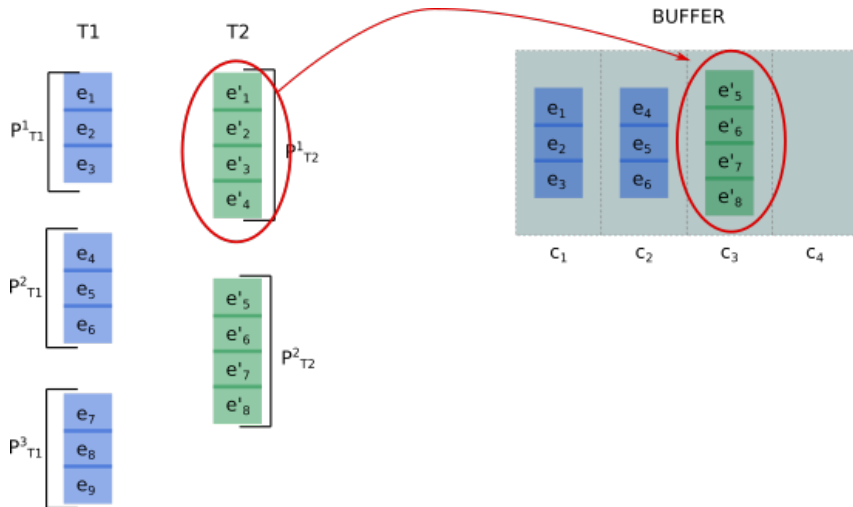
Jointures : par paquet de pages



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 $|P_{T2}| \times |P_{T1}|$ pages ouvertes

Coût ?

Utilisation des paquets de pages

Pour un paquet de page $P1$ de $T1$:

Pour toute les pages p' de $T2$:

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

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 $|P_{T2}| \times |P_{T1}|$ pages ouvertes

Utilisation des paquets de pages

Pour un paquet de page $P1$ de $T1$:

Pour toute les pages p' de $T2$:

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

Coût ?

- $\times |P_{T1}| + \frac{|P_{T1}|}{k} \times |P_{T2}|$ (E/S)
- Si $k = 10 \rightarrow 1000 + 100 \times 500 = 501000$ (E/S) Optimisation possible ?
→ 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 $\sigma_{mid=v_{mid}}(T_2)$.

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

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 $\sigma_{mid=v_{mid}}(T_2)$.

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

Algorithme itératif

Si T_2 a un index sur mid I_{mid}^2 on peut faire: Pour chaque enregistrement de e_i de T_1 :

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

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

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 $\sigma_{mid=v_{mid}}(T_2)$.

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

Algorithme itératif

Si T_2 a un index sur mid I_{mid}^2 on peut faire: Pour chaque enregistrement de e_i de T_1 :

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

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

Coût

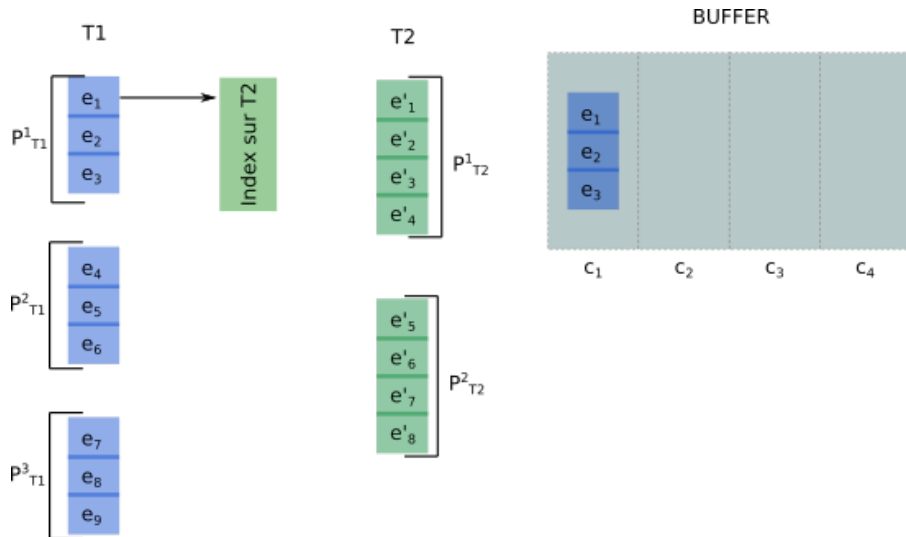
- Parcours de T_1
- Recherche pour chaque enregistrement de T_1

Donc $|P_{T_1}| + |P_{T_1}| \times P_{T_1} \times \text{recherches dans } S$

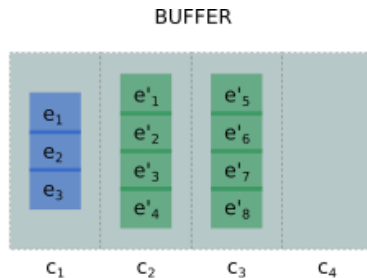
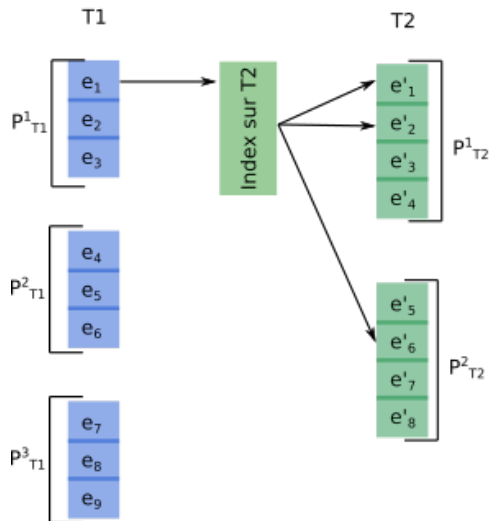
Optimisation possible

→ Utilisation de la plus petite relation en externe (ici T_2)

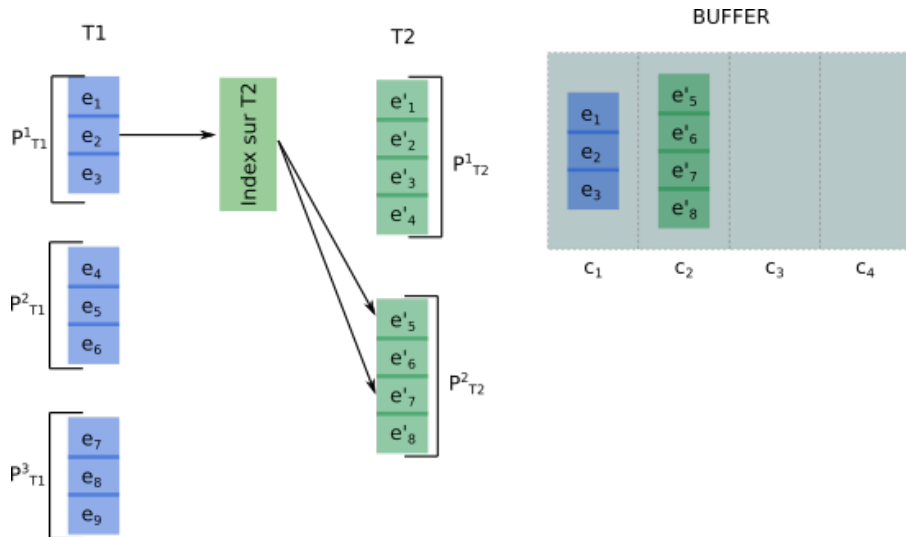
Jointures : itératif et index



Jointures : itératif et index



Jointures : itératif et index



Jointures : itératif et index (Exemples)

Rappel et notations

Rel	$ page $	$ fichier $
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

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

Rappel et notations

Rel	$ page $	$ fichier $
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T1}| + |P_{T1}| \times S_{T1} \times \text{recherches dans } S$
- **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)

Rappel et notations

Rel	$ page $	$ fichier $
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T1}| + |P_{T1}| \times S_{T1} \times \text{recherches dans } S$
- **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)

Rappel et notations

Rel	$ page $	$ fichier $
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T1}| + |P_{T1}| \times S_{T1} \times \text{recherches dans } S$
- **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

$$|P_{T1}| + |P_{T1}| \times P_{T1} \times (h + N_S^{mid})$$

Rappel et notations

Rel	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T1}| + |P_{T1}| \times P_{T1} \times \text{recherches dans } S$
- **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)

Rappel et notations

Rel	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T1}| + |P_{T1}| \times P_{T1} \times \text{recherches dans } S$
- **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** $\rightarrow \frac{N_S^{mid}}{P_{T2}}$

Rappel et notations

Rel	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T1}| + |P_{T1}| \times P_{T1} \times \text{recherches dans } S$
- **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

$$|P_{T1}| + |P_{T1}| \times P_{T1} \times \left(h + \frac{N_S^{mid}}{P_{T2}}\right)$$

Rappel et notations

Relation	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T1}| + |P_{T1}| \times S_{T1} \times \text{recherches dans } S$

- **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)

Rappel et notations

Relation	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

• **Coût** : $|P_{T1}| + |P_{T1}| \times S_{T1} \times \text{recherches dans } S$

• **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)

Rappel et notations

Relation	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T1} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

• **Coût** : $|P_{T1}| + |P_{T1}| \times S_{T1} \times \text{recherches dans } S$

• **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 $|P_{T1}| + |P_{T1}| \times S_{T1} \times (h + 1)$

Rappel et notations

Rel	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T2} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T2}| + |P_{T2}| \times S_{T1} \times \text{recherches dans T2}$
- **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

Rappel et notations

Rel	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T2} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T2}| + |P_{T2}| \times S_{T1} \times \text{recherches dans T2}$
- **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)

Rappel et notations

Rel	enr/page	pages
T1	$S_{T1} = 50$	$ P_{T2} = 1000$
T2	$S_{T2} = 25$	$ P_{T2} = 500$

- **Coût** : $|P_{T2}| + |P_{T2}| \times S_{T1} \times \text{recherches dans T2}$
- **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 $|P_{T2}| + |P_{T2}| \times P_{T2} \times 2$

Algorithmes par partitionnement

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**

Trier les relations T1 et T2 (build)

- Trier les deux relations (T1 & T2)
Tri fusion, tri rapide $\rightarrow n \log(n)$

Trier les relations T1 et T2 (build)

- Trier les deux relations (T1 & T2)
Tri fusion, tri rapide $\rightarrow n \log(n)$
- Quel coût (pages ouvertes, E/S) ?

Trier les relations T1 et T2 (build)

- Trier les deux relations (T1 & T2)
Tri fusion, tri rapide $\rightarrow n \log(n)$
- Quel coût (pages ouvertes, E/S) ?
 $2|P_{T1}| \log(|P_{T1}|) + 2|P_{T2}| \log(|P_{T2}|)$

Jointure sur relations triées (probe)

- Parcourir les données triées en parallèle

Trier les relations T1 et T2 (build)

- Trier les deux relations (T1 & T2)
Tri fusion, tri rapide $\rightarrow n \log(n)$
- Quel coût (pages ouvertes, E/S) ?
 $2|P_{T1}| \log(|P_{T1}|) + 2|P_{T2}| \log(|P_{T2}|)$

Jointure sur relations triées (probe)

- Parcourir les données triées en parallèle
- Quel coût (pages ouvertes, E/S) ?

Trier les relations T1 et T2 (build)

- Trier les deux relations (T1 & T2)
Tri fusion, tri rapide $\rightarrow n \log(n)$
- Quel coût (pages ouvertes, E/S) ?
 $2|P_{T1}| \log(|P_{T1}|) + 2|P_{T2}| \log(|P_{T2}|)$

Jointure sur relations triées (probe)

- Parcourir les données triées en parallèle
- Quel coût (pages ouvertes, E/S) ?

$$|P_{T1}| + |P_{T2}|$$

si unicité dans l'une des tables

- Sinon borné par $|P_{T1}| \times |P_{T2}|$
 $\rightarrow$ cas du produit cartésien

En pratique efficace

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

T1



A	
1	
2	
2	
3	
4	
6	
8	
8	
9	

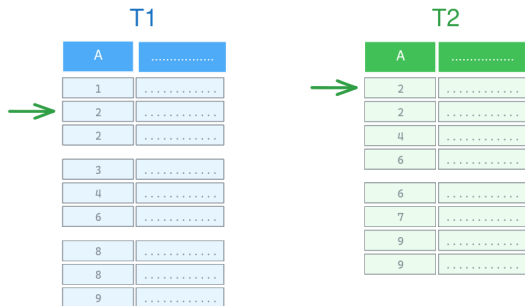
T2



A	
2	
2	
4	
6	
6	
7	
9	
9	

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

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

T1



A	
1	
2	
2	
3	
4	
6	
8	
8	
9	

T2



A	
2	
2	
4	
6	
6	
7	
9	
9	

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 partitionnement - Tri & fusion: la fusion de relations triées

T1



A	
1	
2	
2	
3	
4	
6	
8	
8	
9	

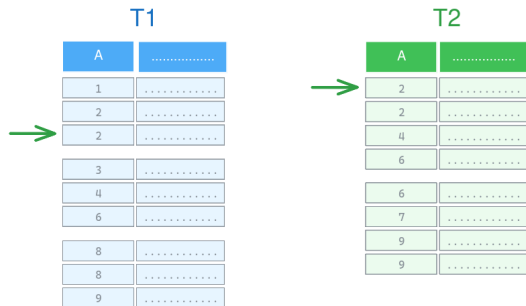
T2



A	
2	
2	
4	
6	
6	
7	
9	
9	

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 partitionnement - 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 T_1 , fusionner chaque couple de pages
→ Un fragment (en sortie) correspond à deux pages
- **Étape 2 :** Prendre les précédents fragments T_1 , fusionner chaque couple de fragments
→ Un fragment (en sortie) correspond à deux pages
- ...
Obtention d'un fragment trié sur tous les enregistrements/pages

Algorithm 1 algorithme de fusion de deux fragments

Data: A, B fragments

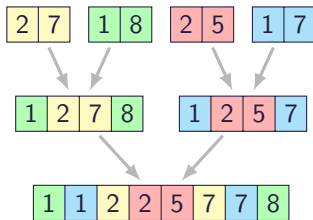
Result: L un fragment trié

```
 $i \leftarrow 1; j \leftarrow 1; L \leftarrow []$  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
```

Coût du Tri :

- Étape 1 :

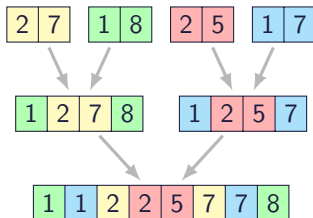
- Prendre deux à deux les pages de T_1

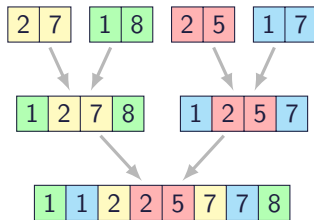


Coût du Tri :

- Étape 1 :

- Prendre deux à deux les pages de $T_1 \rightarrow M$ lectures
- Écrire les fragments de T_1 (même taille que T_1)





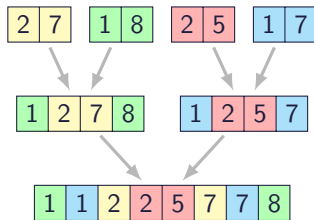
Coût du Tri :

• Étape 1 :

- Prendre deux à deux les pages de $T1 \rightarrow M$ lectures
- Écrire les fragments de $T1$ (même taille que $T1$) $\rightarrow M$ écritures

• Étape 2 :

- Prendre deux à deux les $\frac{|P_{T1}|}{2}$ fragments précédents de taille 2 (M pages écrites)



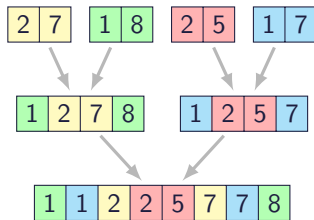
Coût du Tri :

• Étape 1 :

- Prendre deux à deux les pages de $T1 \rightarrow M$ lectures
- Écrire les fragments de $T1$ (même taille que $T1$) $\rightarrow M$ écritures

• Étape 2 :

- Prendre deux à deux les $\frac{|P_{T1}|}{2}$ fragments précédents de taille 2 (M pages écrites) $\rightarrow M$ lectures
- Écrire les nouveaux fragments de taille 4



Coût du Tri :

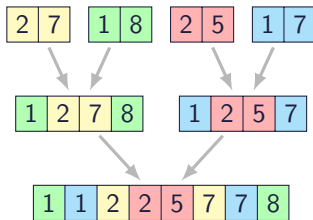
• Étape 1 :

- Prendre deux à deux les pages de $T1 \rightarrow M$ lectures
- Écrire les fragments de $T1$ (même taille que $T1$) $\rightarrow M$ écritures

• Étape 2 :

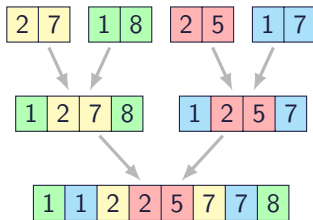
- Prendre deux à deux les $\frac{|P_{T1}|}{2}$ fragments précédents de taille 2 (M pages écrites) $\rightarrow M$ lectures
- Écrire les nouveaux fragments de taille 4 $\rightarrow M$ écritures

Coût du Tri pour l'étape i



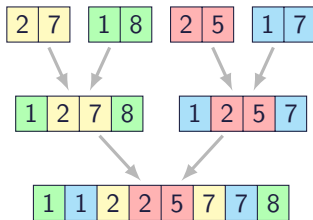
Coût du Tri pour l'étape i

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



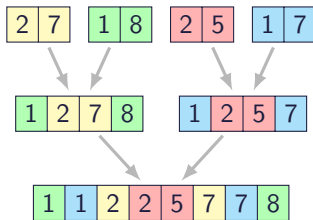
Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{|P_{T1}|}{2^i}$ fragments précédents de taille 2^i
→ $|P_{T1}|$ lectures



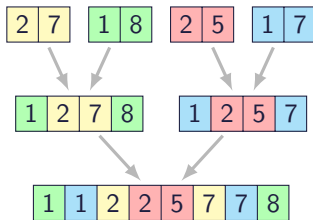
Coût du Tri pour l'étape i

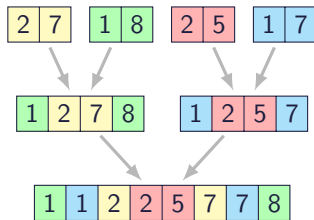
- Prendre deux à deux les $\frac{|P_{T1}|}{2^i}$ fragments précédents de taille 2^i
→ $|P_{T1}|$ lectures
- Écrire les nouveaux fragments de taille 2^{i+1}



Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{|P_{T1}|}{2^i}$ fragments précédents de taille 2^i
→ $|P_{T1}|$ lectures
- Écrire les nouveaux fragments de taille 2^{i+1}
→ $|P_{T1}|$ écritures



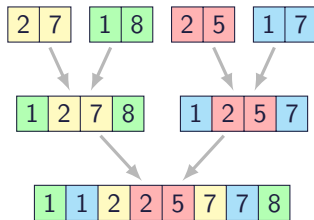


Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{|P_{T1}|}{2^i}$ fragments précédents de taille 2^i
→ $|P_{T1}|$ lectures
- Écrire les nouveaux fragments de taille 2^{i+1}
→ $|P_{T1}|$ écritures

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

$$2|P_{T1}| \times \text{nombre_étape}$$



Coût du Tri pour l'étape i

- Prendre deux à deux les $\frac{|P_{T1}|}{2^i}$ fragments précédents de taille 2^i
→ $|P_{T1}|$ lectures
- Écrire les nouveaux fragments de taille 2^{i+1}
→ $|P_{T1}|$ écritures

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

$$2|P_{T1}| \times \text{nombre_étape}$$

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

Optimisation du tri

- Diminuer le nombre d'étape

Optimisation du tri

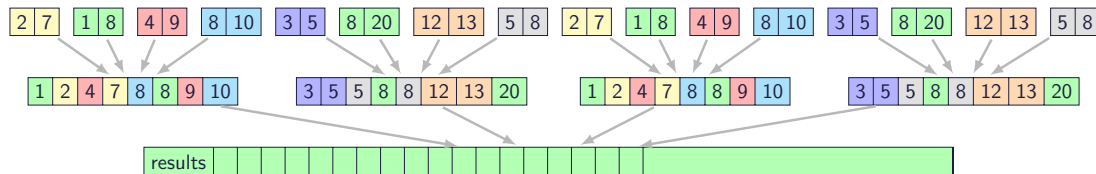
- Diminuer le nombre d'étape
- Augmentation du nombre de pages ou de fragments fusionnés (ici k)

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

Optimisation du tri

- Diminuer le nombre d'étape

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

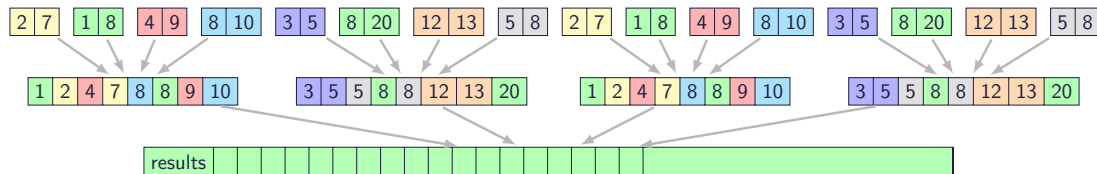


Jointure par partitionnement - Tri & fusion : Tri multi-fusion

Optimisation du tri

- Diminuer le nombre d'étape

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



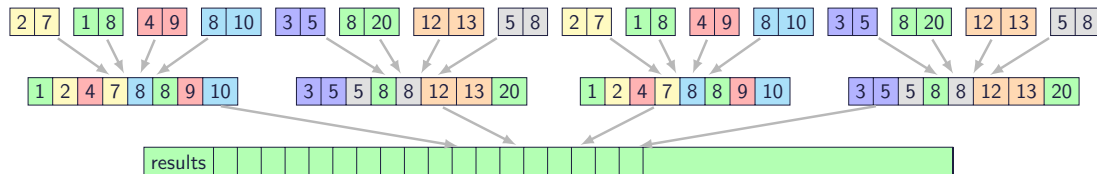
Coût ?

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

Optimisation du tri

- Diminuer le nombre d'étape

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



Coût ?

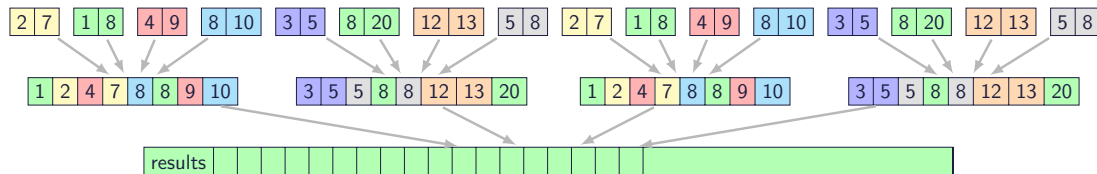
- $\rightarrow 2F_T \times \lceil 1 + \log_k(F_T) \rceil$
- Taille du buffer

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

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 : $k + 1$

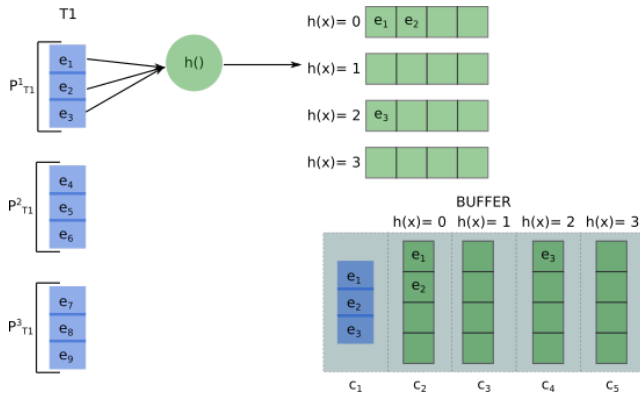
Principe de la jointure par Hachage

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

Jointure par partitionnement - Hachage (Grace Hash Join)

Principe de la jointure par Hachage

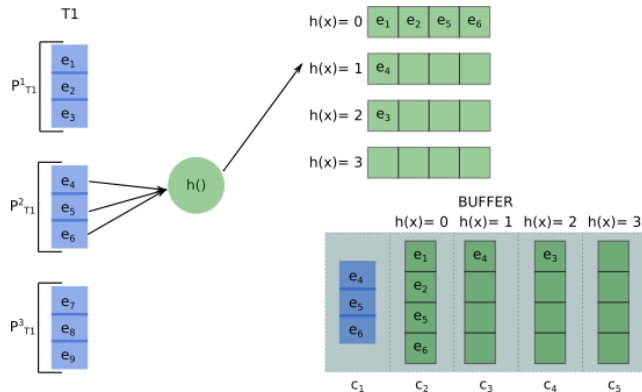
1. **build** : Création des tables de hachage pour les deux relations
2. **probe** : Joindre les deux tables par parcours



Jointure par partitionnement - Hachage (Grace Hash Join)

Principe de la jointure par Hachage

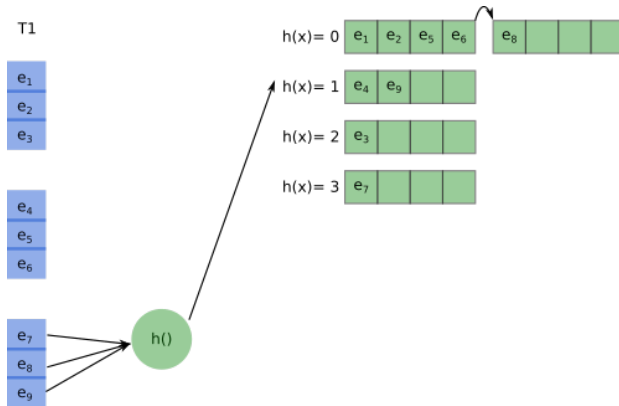
1. **build** : Création des tables de hachage pour les deux relations
2. **probe** : Joindre les deux tables par parcours



Jointure par partitionnement - Hachage (Grace Hash Join)

Principe de la jointure par Hachage

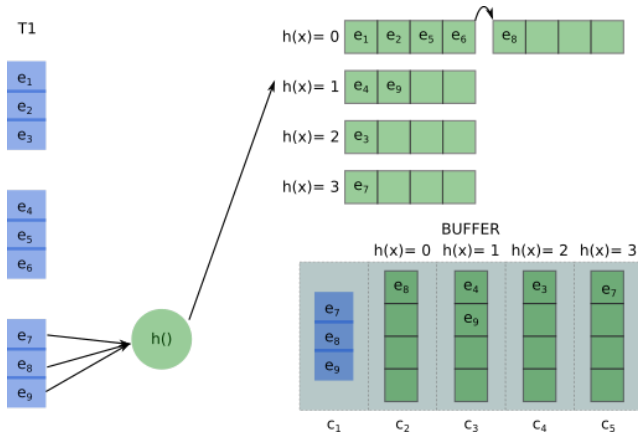
1. **build** : Création des tables de hachage pour les deux relations
2. **probe** : Joindre les deux tables par parcours



Jointure par partitionnement - Hachage (Grace Hash Join)

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 partitionnement - Hachage (Grace Hash Join) (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

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 partitionnement - Hachage (Grace Hash Join) (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 T_1 , 1 lecture/1 écriture de toutes les pages
- Parcours de T_2 : Idem

Donc $2 \times |P_{T_1}| + 2 \times |P_{T_2}|$ (Mais avoir une taille de buffer suffisante)

Taille du buffer

Jointure par partitionnement - Hachage (Grace Hash Join) (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 T_1 , 1 lecture/1 écriture de toutes les pages
- Parcours de T_2 : Idem

Donc $2 \times |P_{T_1}| + 2 \times |P_{T_2}|$ (Mais avoir une taille de buffer suffisante)

Taille du buffer

- 1 cellule pour stocker la page en lecture

Jointure par partitionnement - Hachage (Grace Hash Join) (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 T_1 , 1 lecture/1 écriture de toutes les pages
- Parcours de T_2 : Idem

Donc $2 \times |P_{T_1}| + 2 \times |P_{T_2}|$ (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 partitionnement - Hachage (Grace Hash Join) (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 T_1 , 1 lecture/1 écriture de toutes les pages
- Parcours de T_2 : Idem

Donc $2 \times |P_{T_1}| + 2 \times |P_{T_2}|$ (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 partitionnement - Hachage (Grace Hash Join)

Principe de la jointure par Hachage

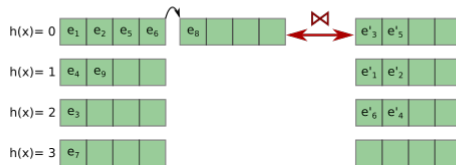
1. **build** : Création des tables de hachage pour les deux relations
2. **probe** : Joindre les deux tables par parcours

T1

e_1
e_2
e_3

e_4
e_5
e_6

e_7
e_8
e_9



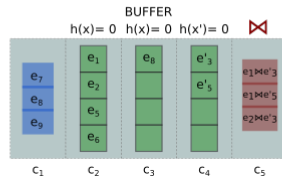
T2

e'_1
e'_2
e'_3

P^1_{T2}

e'_4
e'_5
e'_6

P^2_{T2}



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

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 partitionnement - Hachage (Grace Hash Join)

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 $|P_{T1}| + |P_{T2}|$

Taille du buffer

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

Donc $\text{buffer} \geq \text{taille_max_paquet} + 2$

Coût Total

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

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

Coût Total

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

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

$$\rightarrow 3|P_{T1}| + 3|P_{T2}|$$

Récapitulatif (jointures itératives)

Nom	Coût	Condition requises	Utilisation
Par enregistrements	$ P_{T1} + P_{T1} \times S_{T1} \times P_{T2} $	aucune	✗
Page à Page	$ P_{T1} + P_{T1} \times P_{T2} $	aucune	✓
Paquets	$ P_{T1} + \frac{ P_{T1} }{k} \times P_{T2} $	cache de $k + 1$	✓
Index	$ P_{T1} + P_{T1} \times S_{T1} \times \text{recherche}$	index sur T2	✓

Récapitulatif (jointures par partitionnement)

Nom	Coût Build	Coût Probe	Conditions requises
Tri-fusion	$ P_{T1} \lceil \log_2(P_{T1}) \rceil + P_{T2} \lceil \log_2(P_{T2}) \rceil$	$ P_{T1} + P_{T2} $	aucune
Tri-multi-fusion	$ P_{T1} \lceil \log_k(P_{T1}) \rceil + P_{T2} \lceil \log_k(P_{T2}) \rceil$	$ P_{T1} + P_{T2} $	cache de taille $k+1$
Grace hash-join	$2 P_{T1} + 2 P_{T2} $	$ P_{T1} + P_{T2} $	cache de taille $b+1$ (b étant le nombre de bucket) et taille de cache supérieur au nombre de page de tous les buckets