Comment optimiser l'exécution des requêtes SQL sur une base de données?

Eliott PAQUET

2025

Sommaire

Sommaire

Description du traitement d'une requête

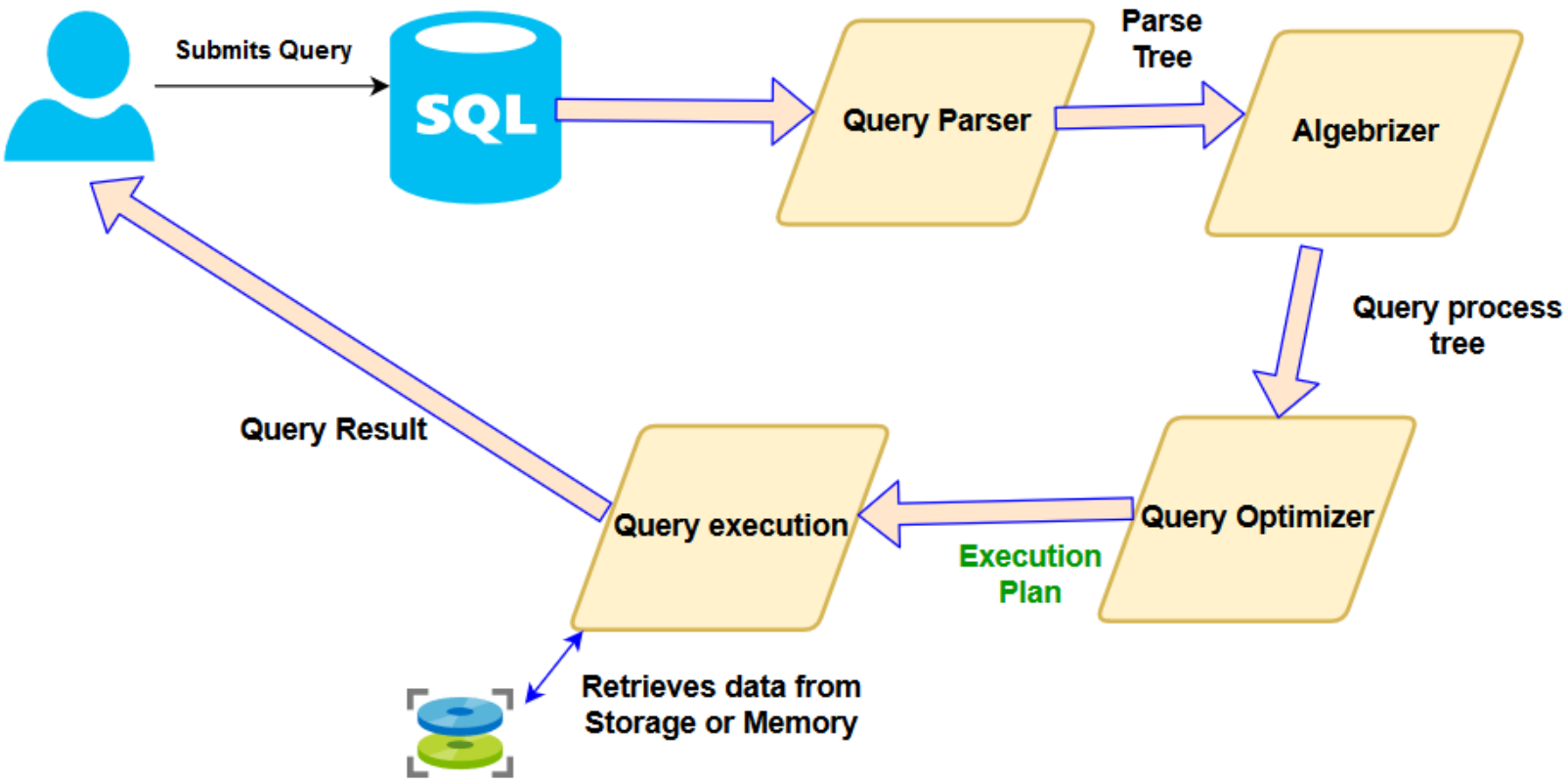
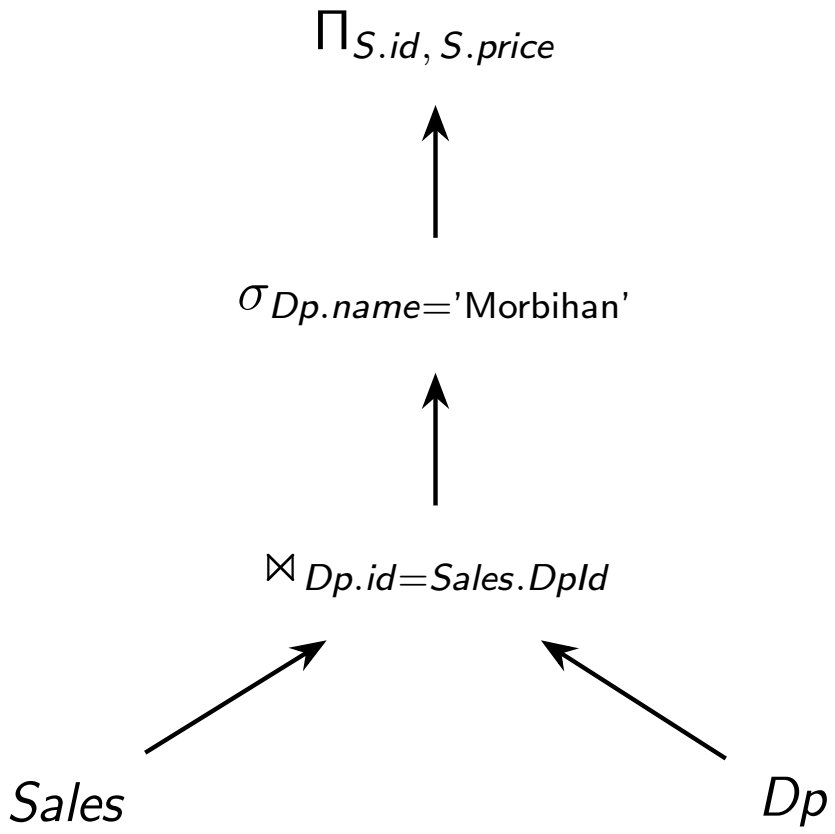


Figure: Traitement d'une requête SQL par un SGBD.

Edgar F. Codd (1970)

Calcul relationnel $\Longleftrightarrow$ Algèbre relationnelle

```sql
SELECT S.id, S.price
FROM Sales AS S
JOIN Dp ON Dp.id = S.DpId
WHERE Dp.name = 'Morbihan';
```



Les bases de données orientées colonnes

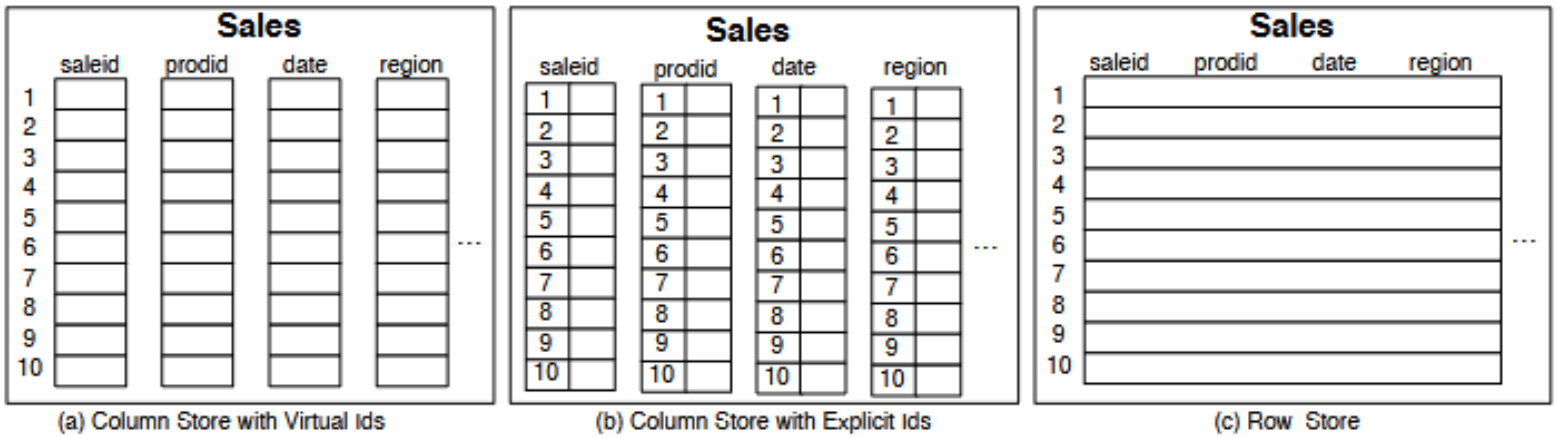
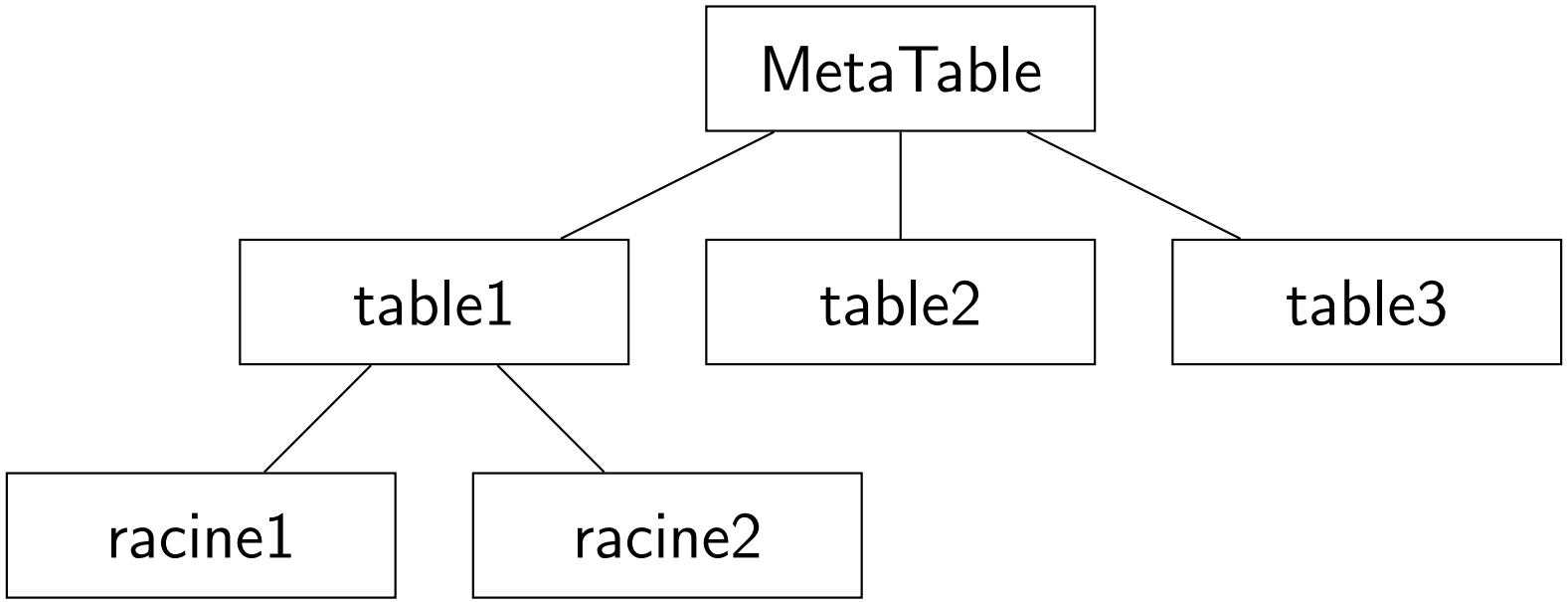


Figure: Rendu physique des BDD orientées colonnes et BDD orientées lignes

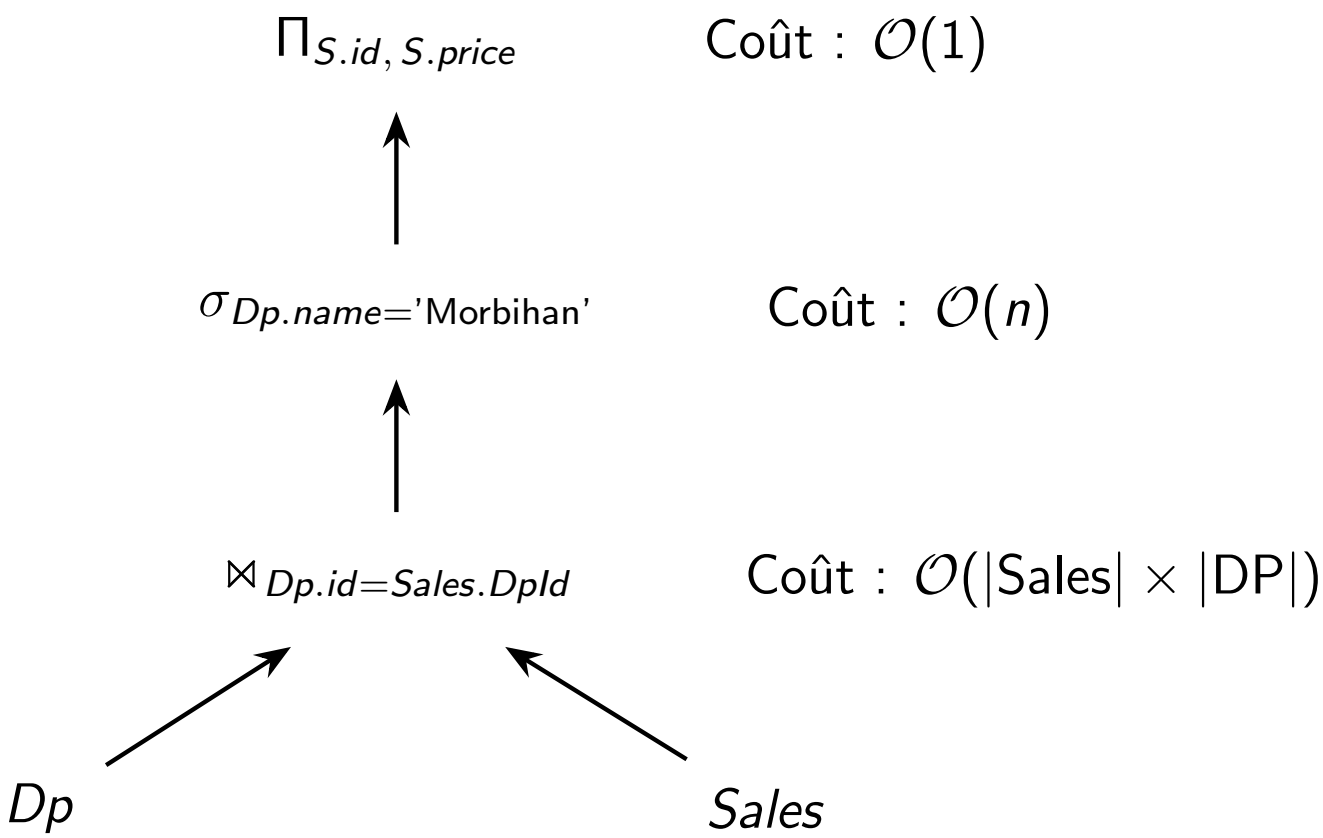
Notre gestion des tables pendant l'exécution



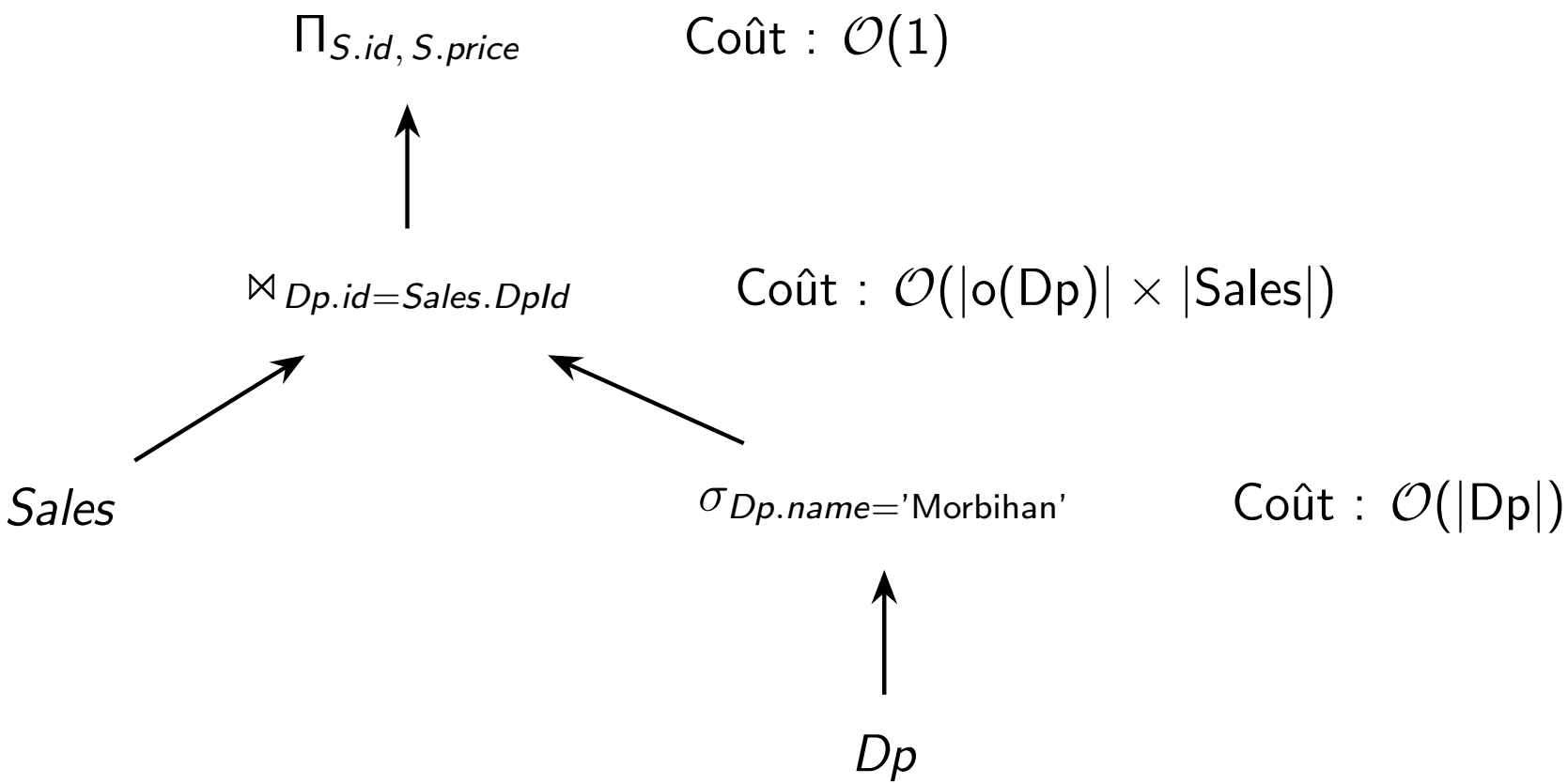
- **Racine**: valeurs chargées.
- **Table**: liste de lignes encore valides.
- **MetaTable**: jointure de plusieurs *Tables*.
- **Particularité**: modifier les lignes d'une *Table* (jointure ou sélection) modifie toutes les *Tables*.

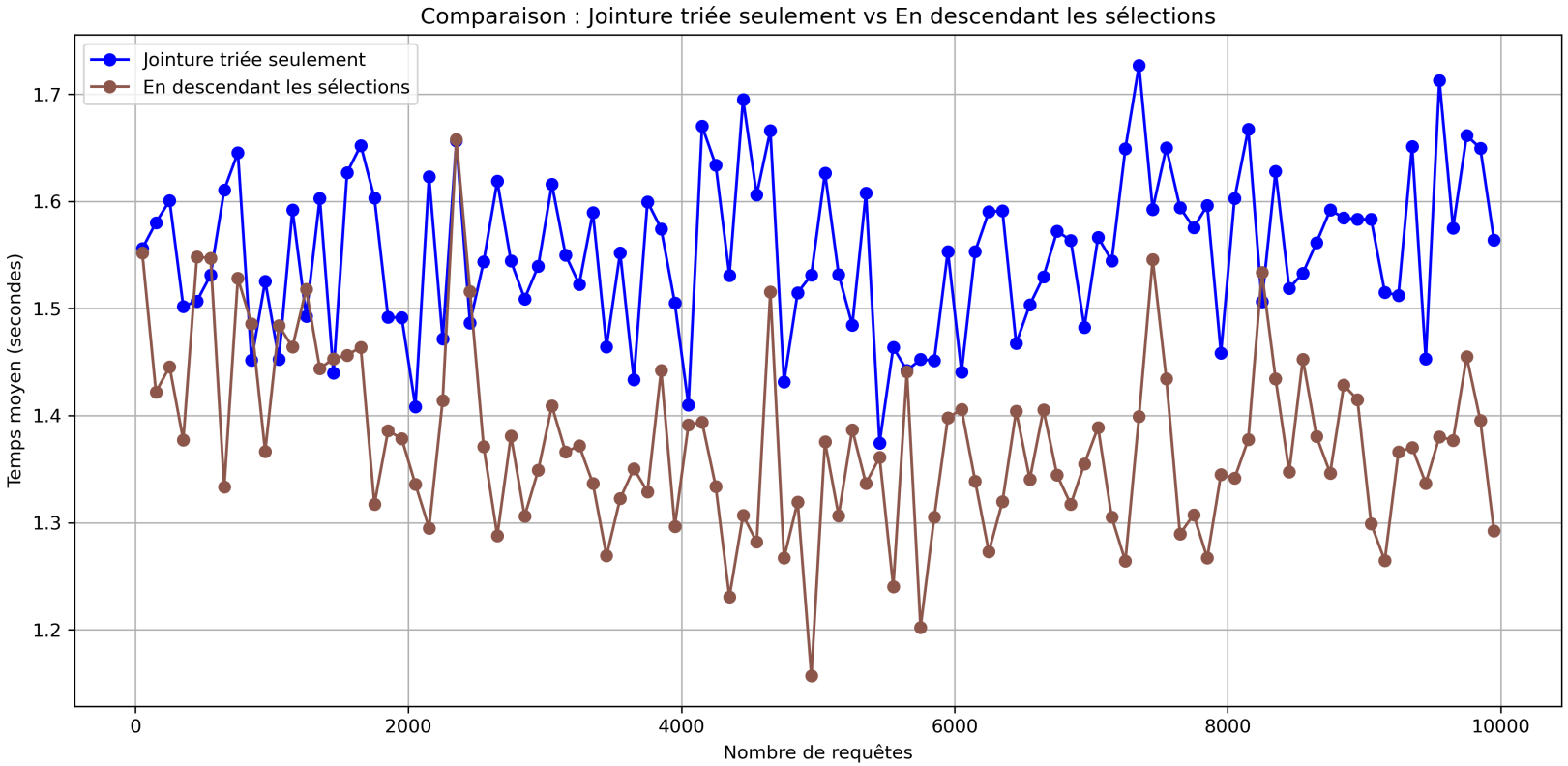
Sommaire

La descente des sélections

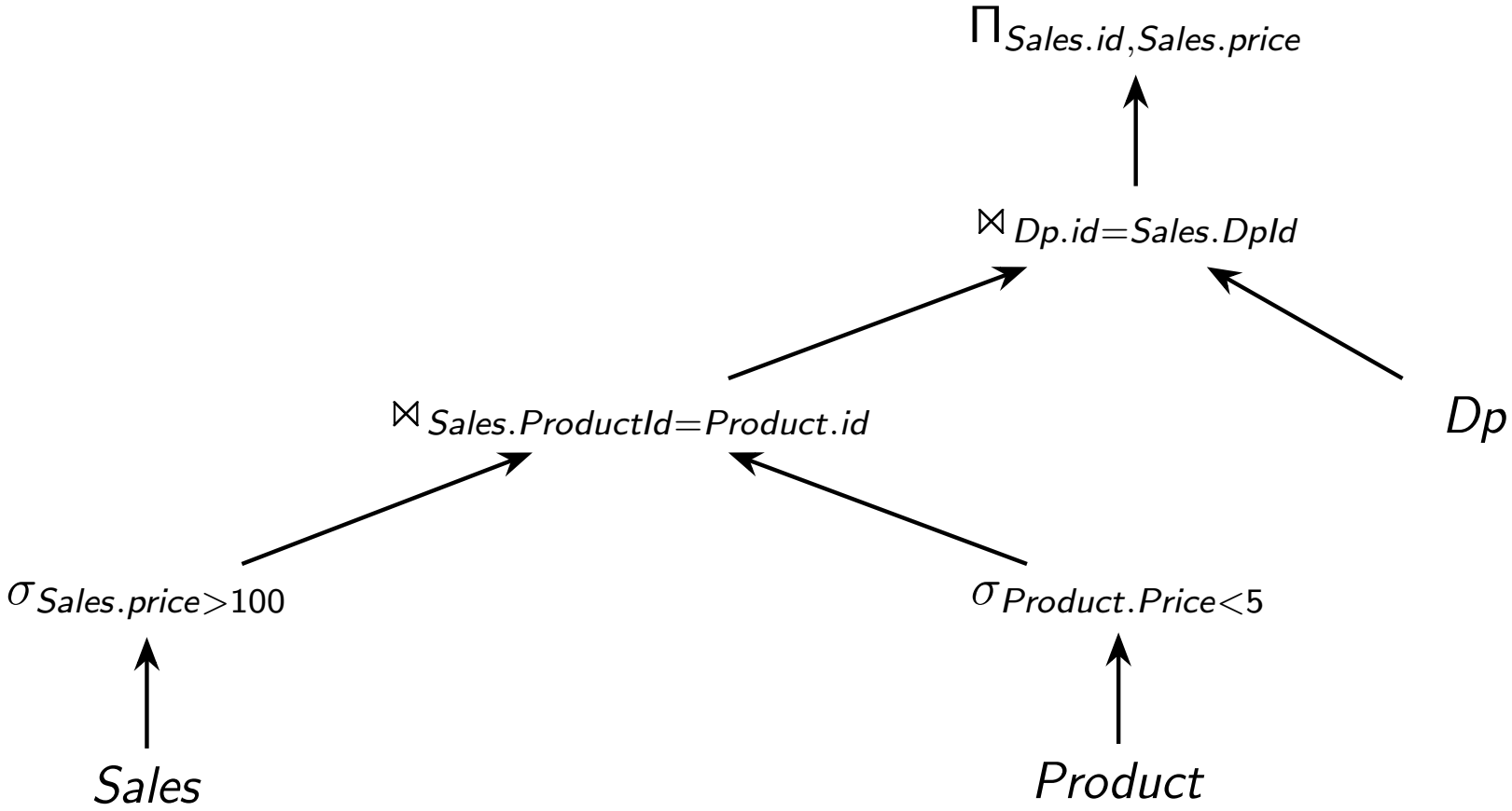


La descente des sélections

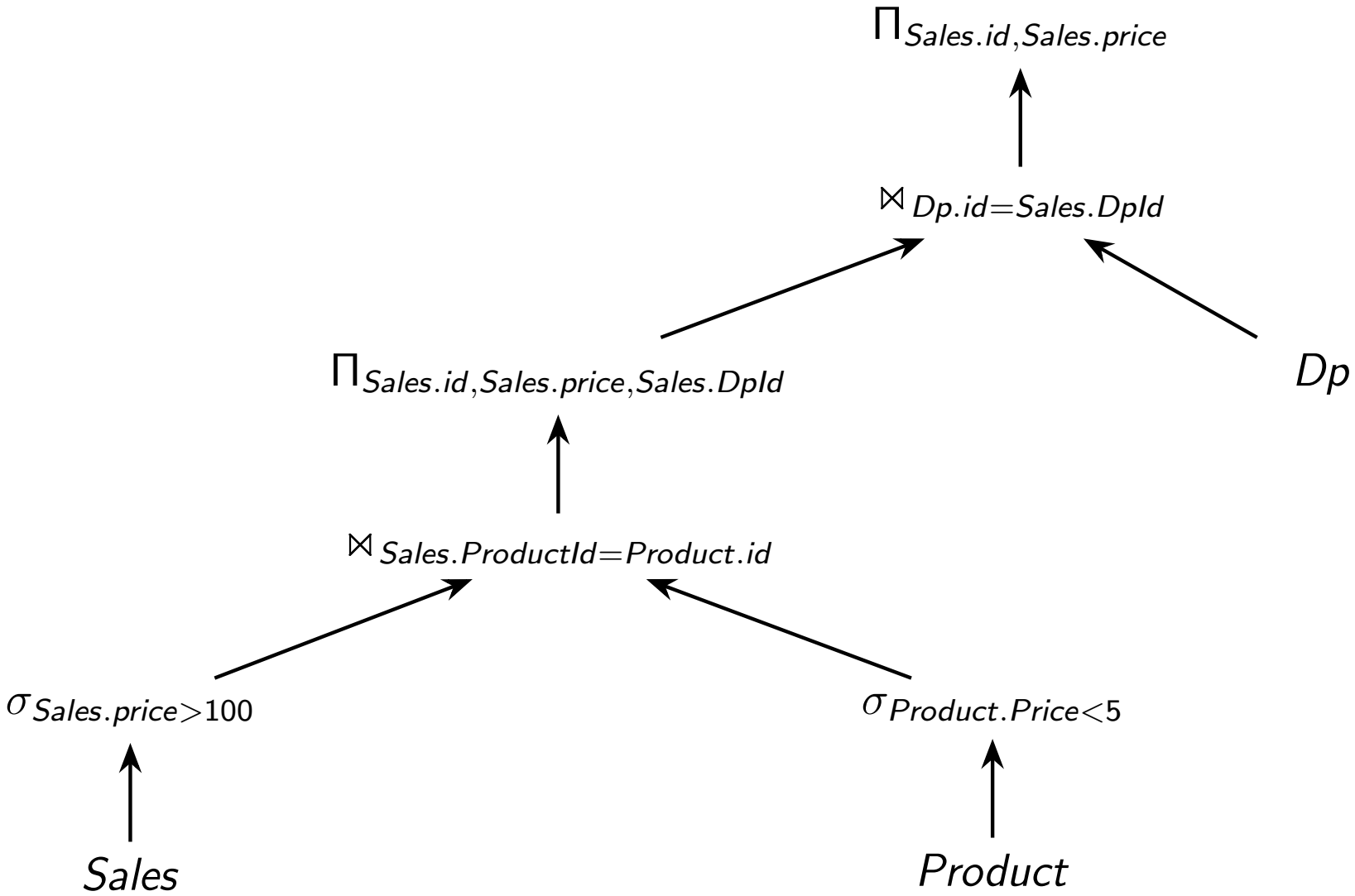


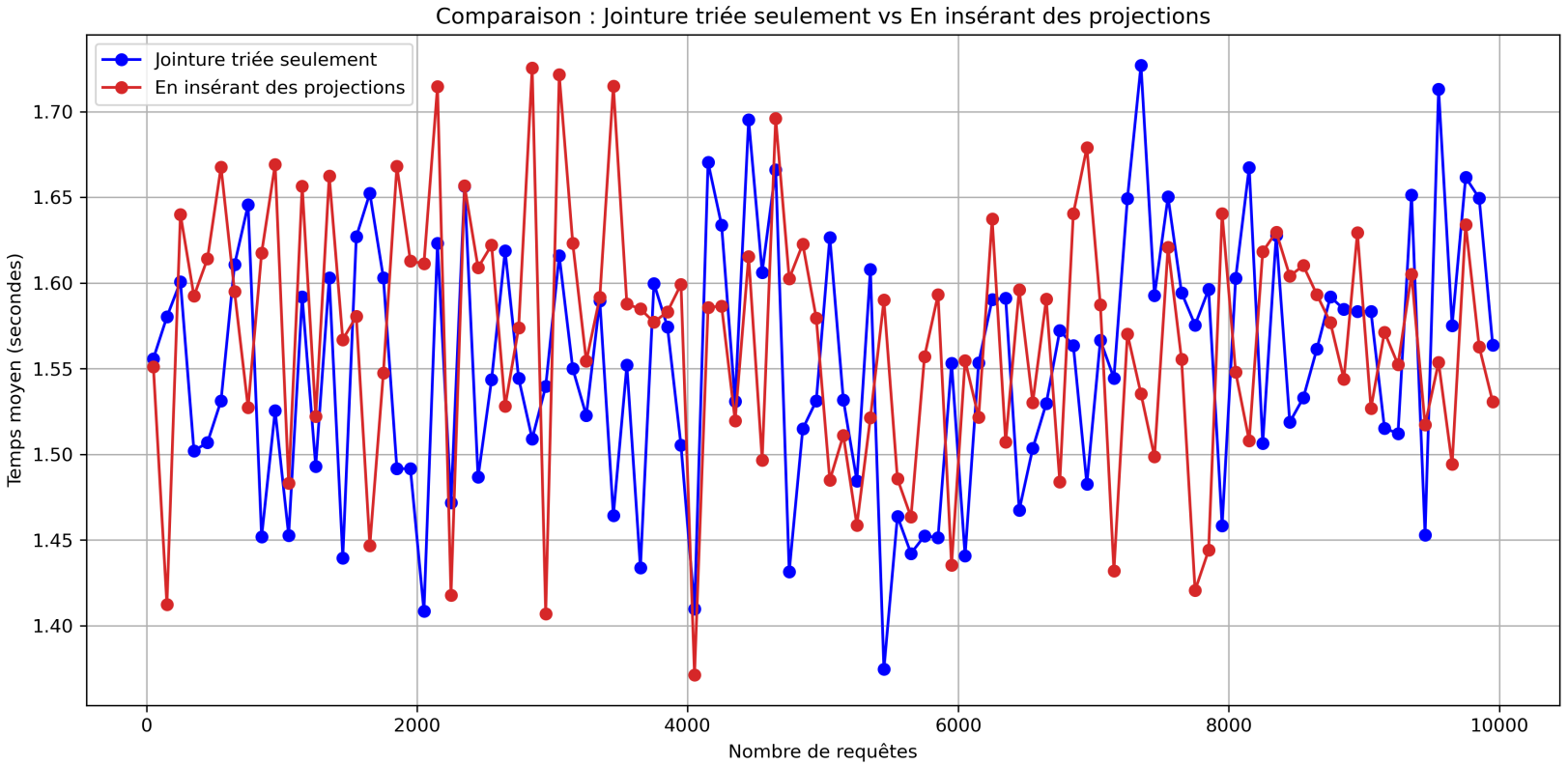

Comparaison : Jointure triée seulement vs En descendant les sélections
Jointure triée seulement
En descendant les sélections
Temps moyen (secondes)
1.2
1.3
1.4
1.5
1.6
1.7
0
2000
4000
6000
8000
10000
Nombre de requêtes

Cas problématique



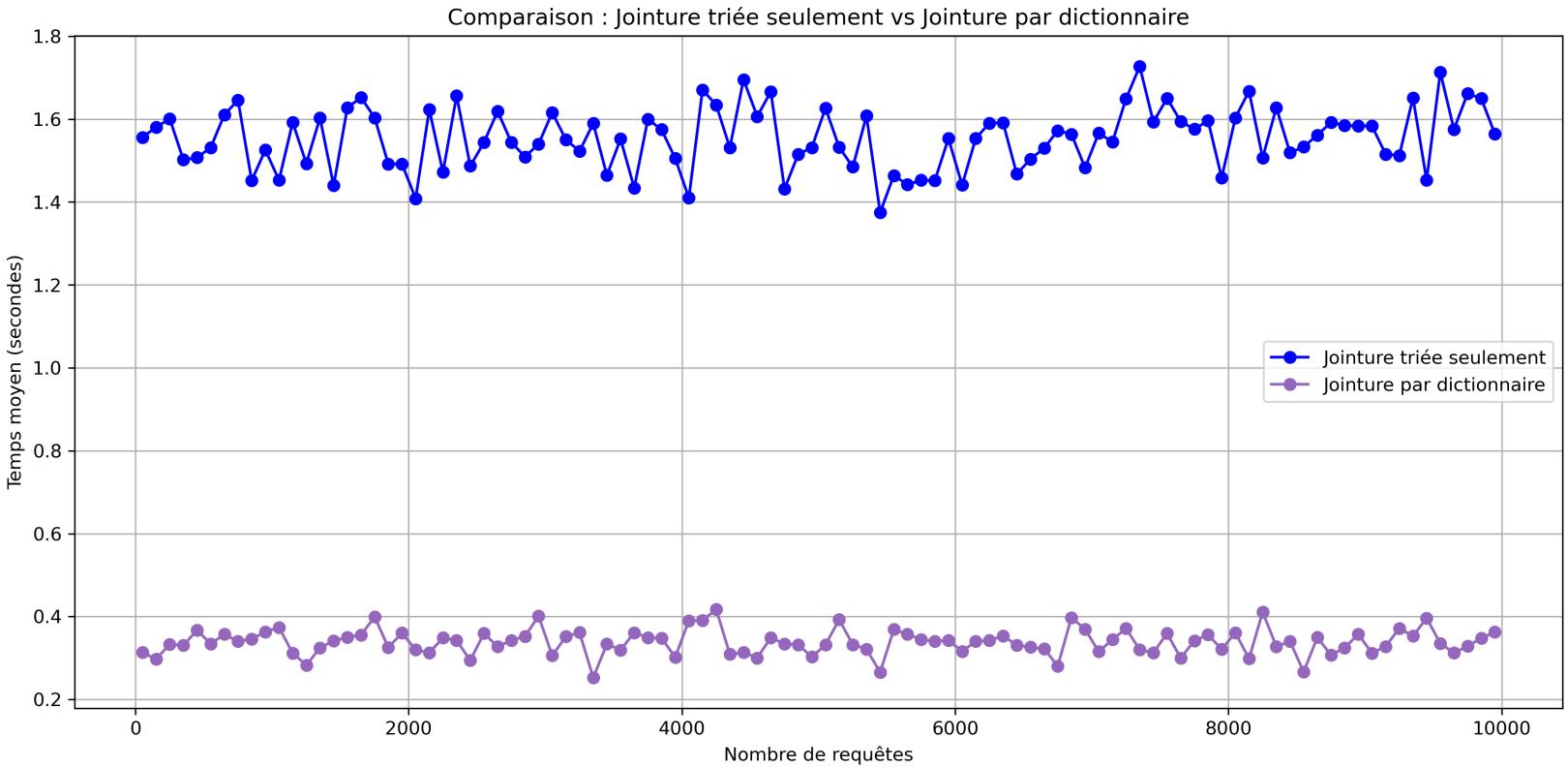
Insertion utile de la Projection




Comparaison : Jointure triée seulement vs En insérant des projections
Jointure triée seulement
En insérant des projections
Temps moyen (secondes)
1.70
1.65
1.60
1.55
1.50
1.45
1.40
0
2000
4000
6000
8000
10000
Nombre de requêtes

Les différents types de jointures possibles

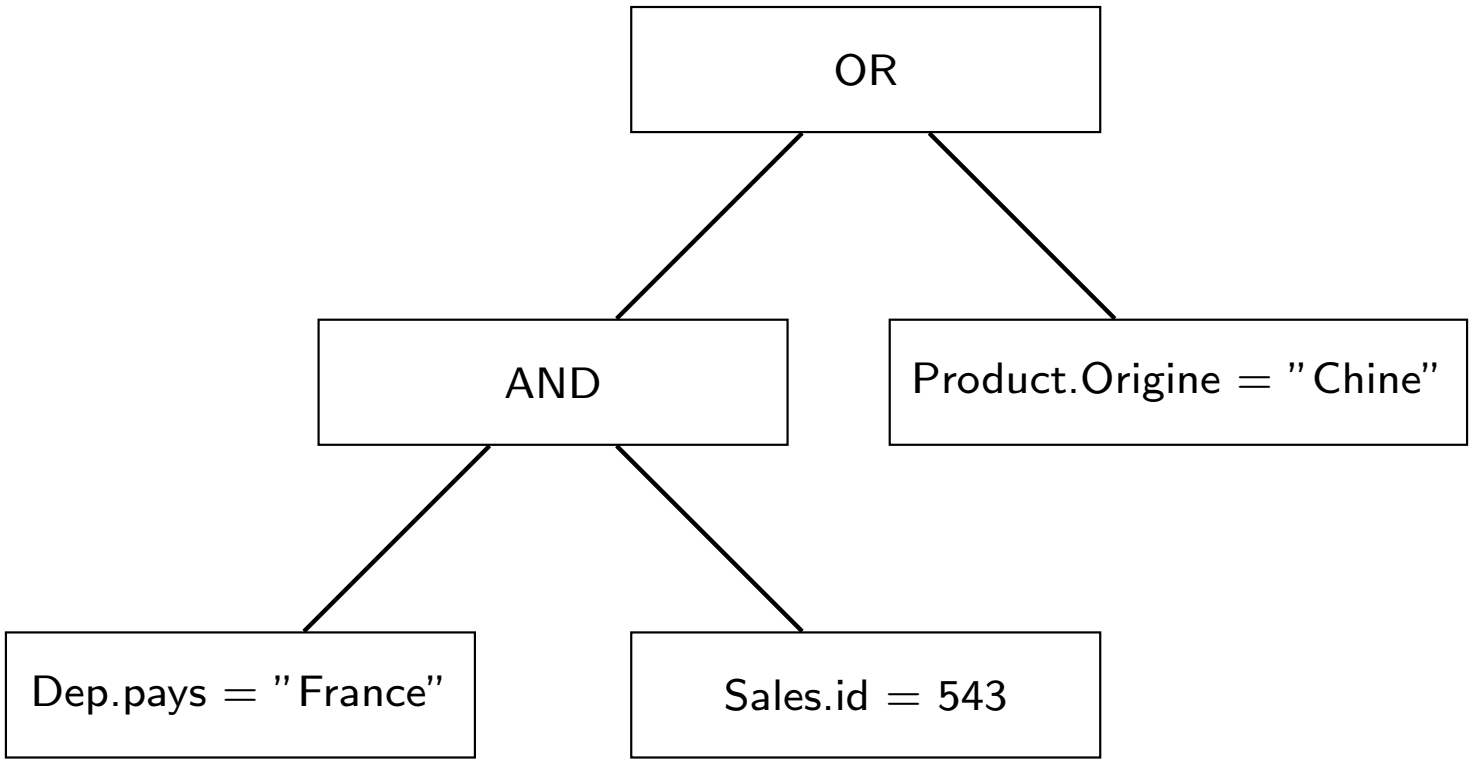
- Produit Cartésien: génère tous les couples possibles et teste la condition de jointure $\rightarrow O(n^2)$
- Jointure Triée: pré-trie les deux tables et lit les deux tables $\rightarrow o(2n\log(n) + 2n)$
- Jointure par dictionnaire: crée un dictionnaire et recherche les correspondances $\rightarrow O(2n)$
- Leap Frog Trie Join: procédé similaire à la jointure triée mais l'étend à k-table en simultané$\rightarrow o(k \times (n\log(n) + n))$.


Comparaison : Jointure triée seulement vs Jointure par dictionnaire
Temps moyen (secondes)
Nombre de requêtes
Jointure triée seulement
Jointure par dictionnaire
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
2000
4000
6000
8000
10000

Sommaire

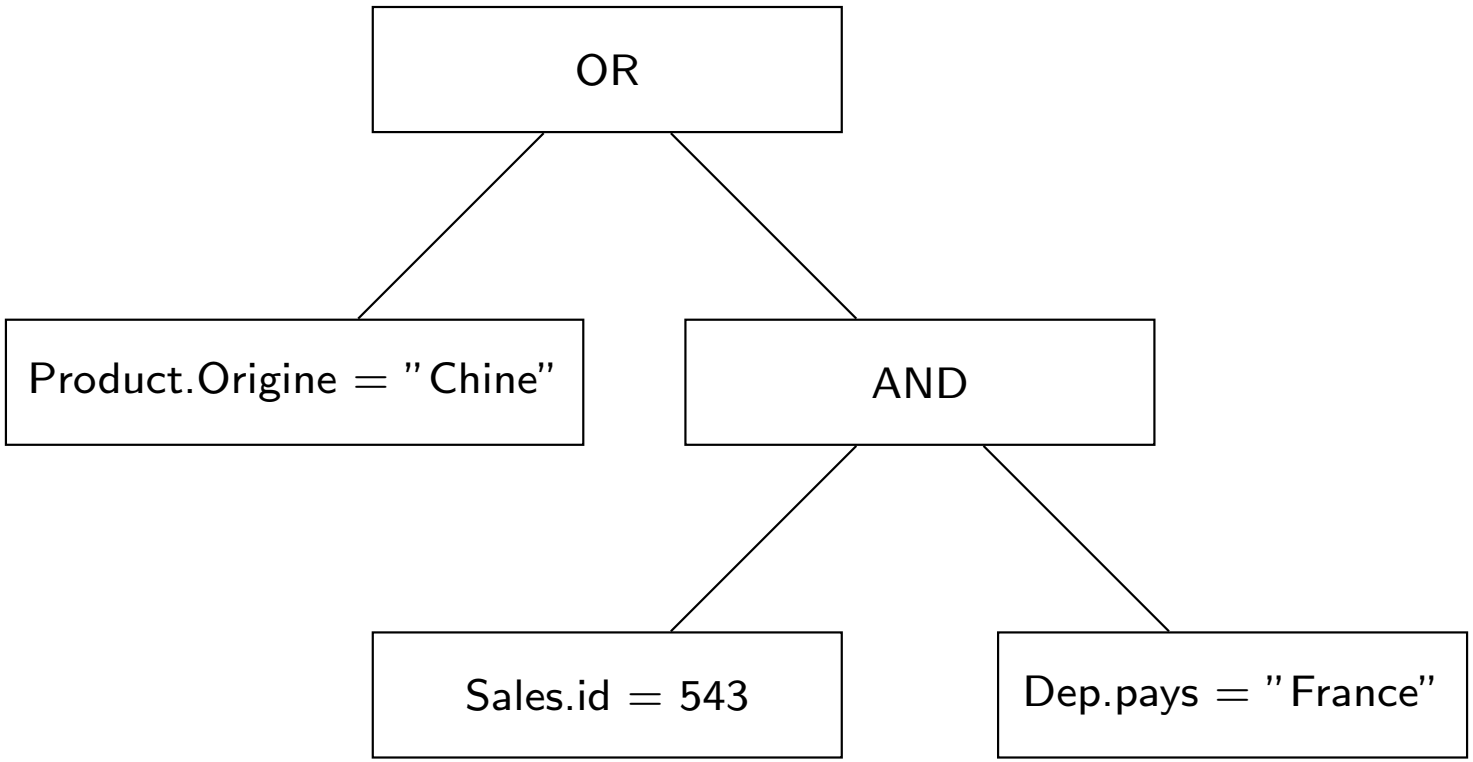
Analyse des expressions binaires

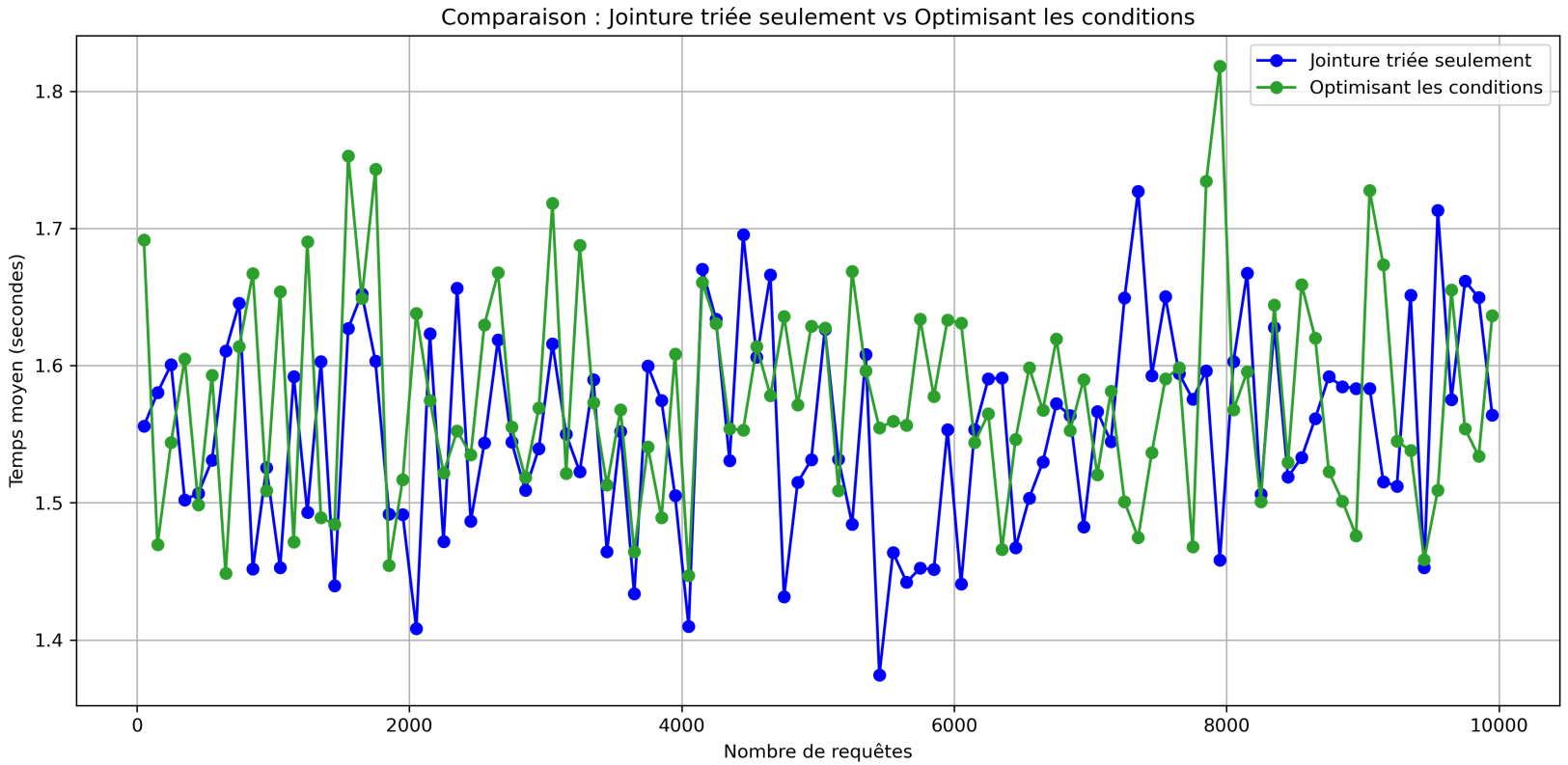
```sql
Where (Dep.pays = "France" AND Sales.id = 543) OR (Product.
    Origine = "Chine")
```



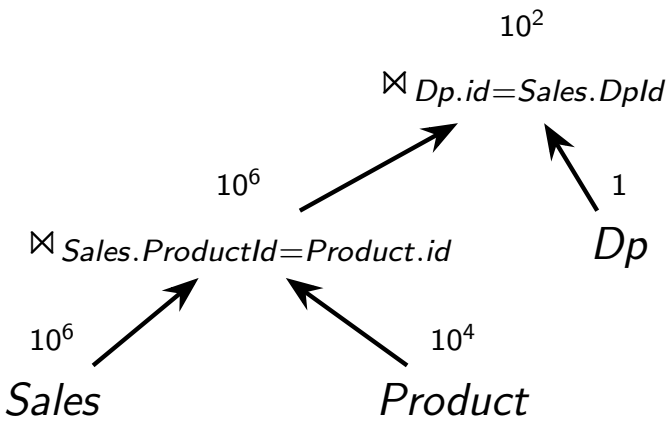
Optimisation des expressions binaires

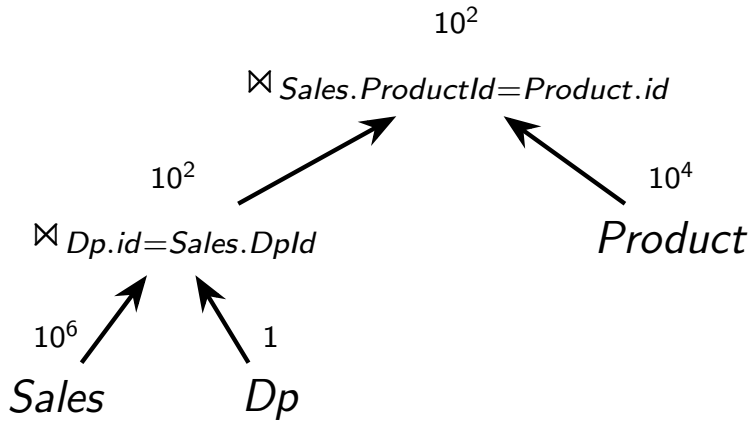
```sql
Where (Dep.pays = "France" AND Sales.id = 543) OR (Product.
    Origine = "Chine")
```



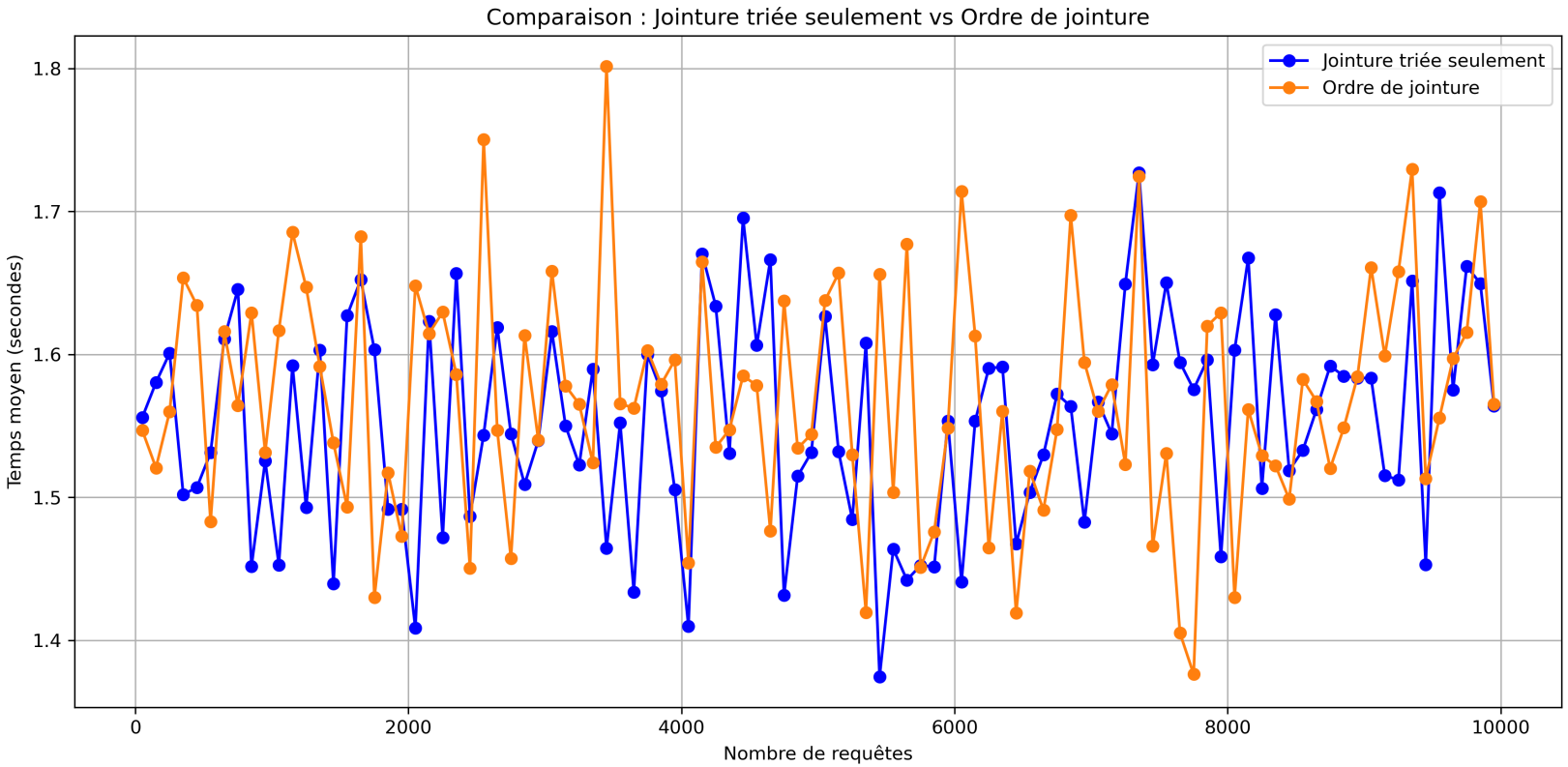

Comparaison : Jointure triée seulement vs Optimisant les conditions
Jointure triée seulement
Optimisant les conditions
Temps moyen (secondes)
1.4
1.5
1.6
1.7
1.8
0
2000
4000
6000
8000
10000
Nombre de requêtes

Résultats équivalent mais complexité différente





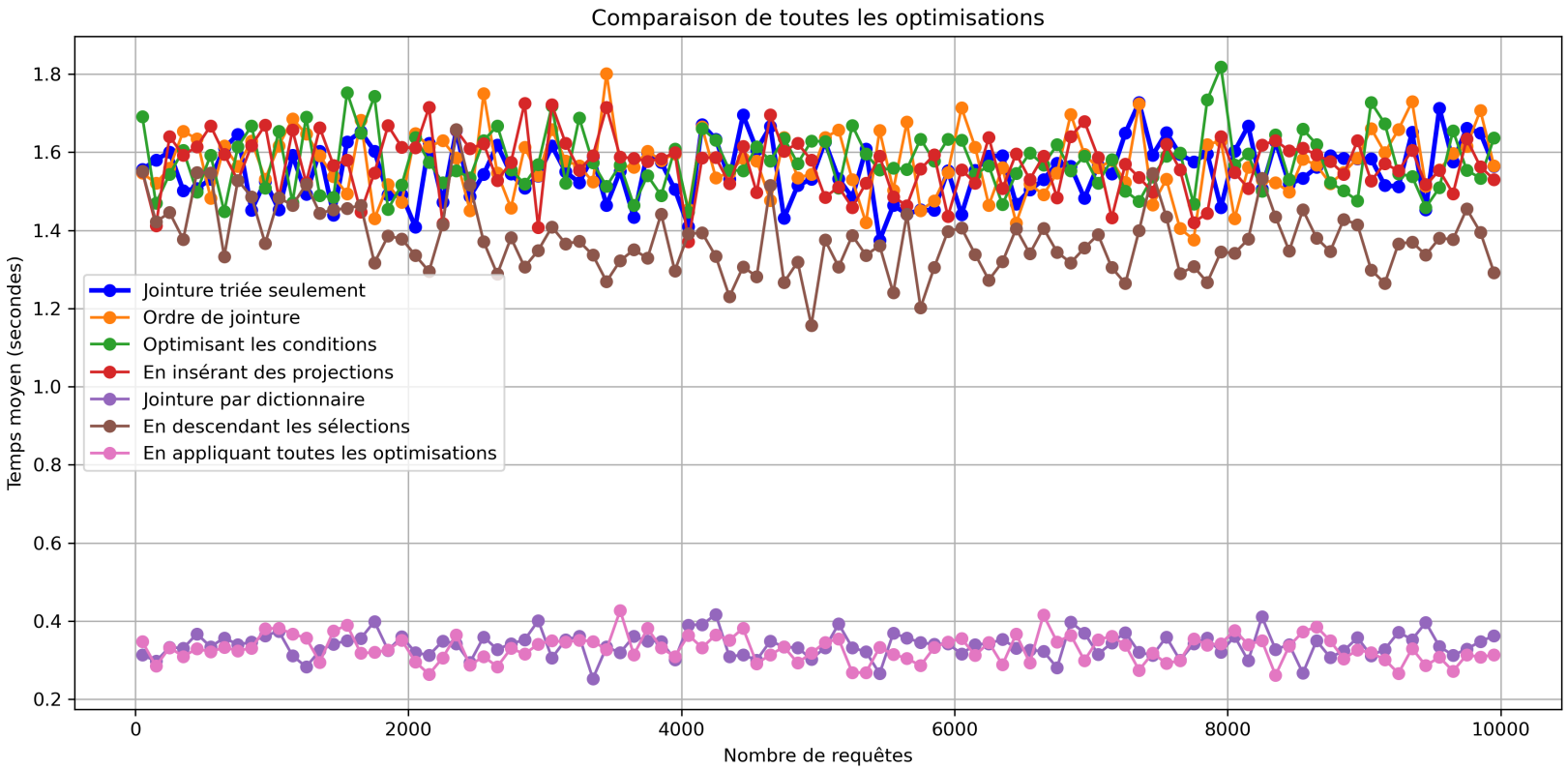
- $\bowtie_{Sales.ProductId=Product.id}$ d'abord $\rightarrow 10^4 \times 10^6 + 10^6 \times 1$ comparaisons
- $\bowtie_{Dp.id=Sales.DpId}$ d'abord $\rightarrow 1 \times 10^6 + 10^2 \times 10^4$ comparaisons


Comparaison : Jointure triée seulement vs Ordre de jointure
Jointure triée seulement
Ordre de jointure
Temps moyen (secondes)
Nombre de requêtes
1.8
1.7
1.6
1.5
1.4
0
2000
4000
6000
8000
10000

Comment choisir le meilleur plan rapidement?

- Choisir les optimisations avant de lancer la requête
- Utiliser l'historique des plans générés
- Donner un coût à chaque opération, énumérer tous les plans et prendre celui avec le plus faible coût

Sommaire


Comparaison de toutes les optimisations
Temps moyen (secondes)
Nombre de requêtes
Jointure triée seulement
Ordre de jointure
Optimisant les conditions
En insérant des projections
Jointure par dictionnaire
En descendant les sélections
En appliquant toutes les optimisations
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
0
2000
4000
6000
8000
10000

Des optimisations impactantes

- Le choix de la jointure est crucial
- Il existe un très grand nombre de plan possible et manière de le choisir.

Les raisons de la lenteur de notre modèle

- De nombreuses techniques n'ont pas été traitées, notamment les plus récentes ($\sim$ 20 ans)
- Les failles de notre code. 45 minutes $\rightarrow$ 22 secondes
- Les optimisations matérielles. Appel aux fonctions internes au processeur
- La compression des données. Exemple: *C-Store* 4 manière de compresser les données
- La vectorisation et le multi-threading

Sommaire

Résumé

- Les bases des Systèmes de Gestion de Base de Données
- Différentes optimisations naïve utilisant l'algèbre relationnelle
- Certaines optimisations dépendantes des données que notre SGBD possède
- Une analyse des résultats obtenus

Ouverture

Les pistes récentes:

- Analyse des données $\Rightarrow$ Machine Learning (Génération de plan, Choix de l'ordre des jointures,..)
- Une meilleure organisation des données en mémoire (Voire le travail d'Antoine)
- Les deux: *Hypothetical Index Benefit Estimation on Column-oriented Databases using Quantiles* 2025
- Base de donnée sur GPU.SIGMOD de Microsoft

Nos ressources

Nos ressources:

- *The Design and Implementation of Modern Column-Oriented Database Systems* 2013
- *C-Store: A Column-oriented DBMS* 2005
- *A Survey on Advancing the DBMS Query Optimizer: Cardinality Estimation, Cost Model, and Plan Enumeration* 2021
- *Extensible Query Optimizers in Practice* 2024
- *Foundation of Databases* 1995