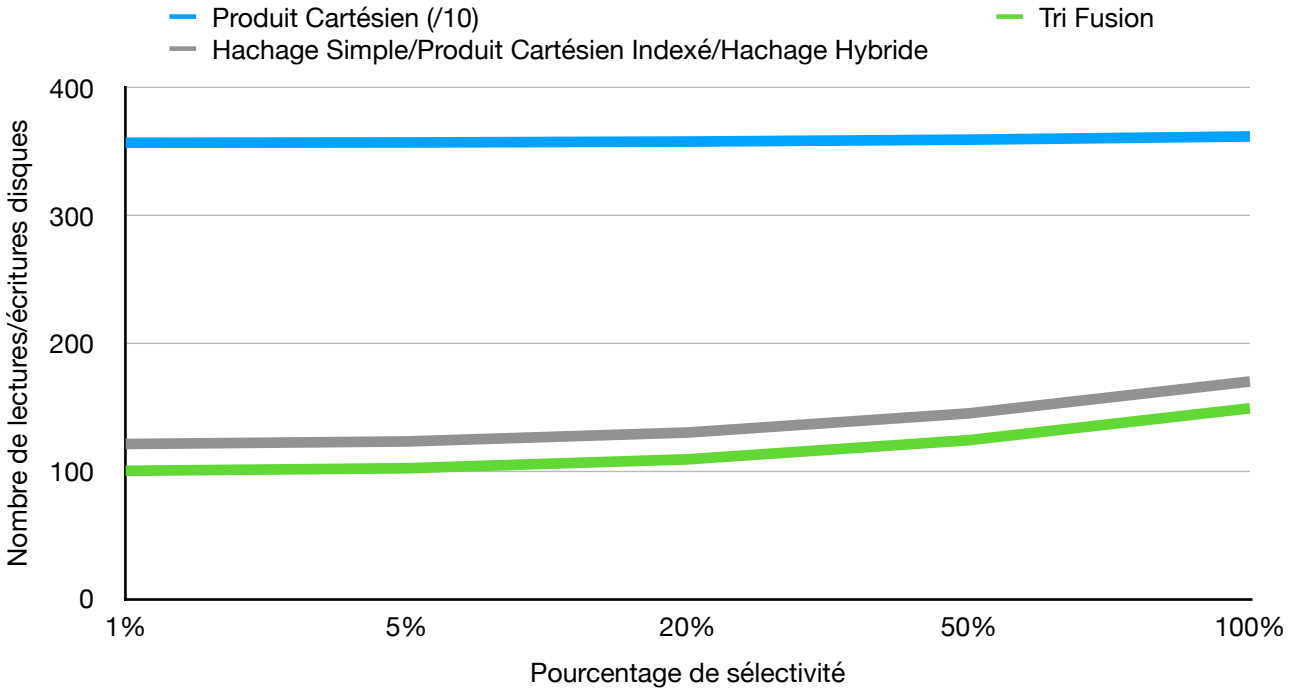
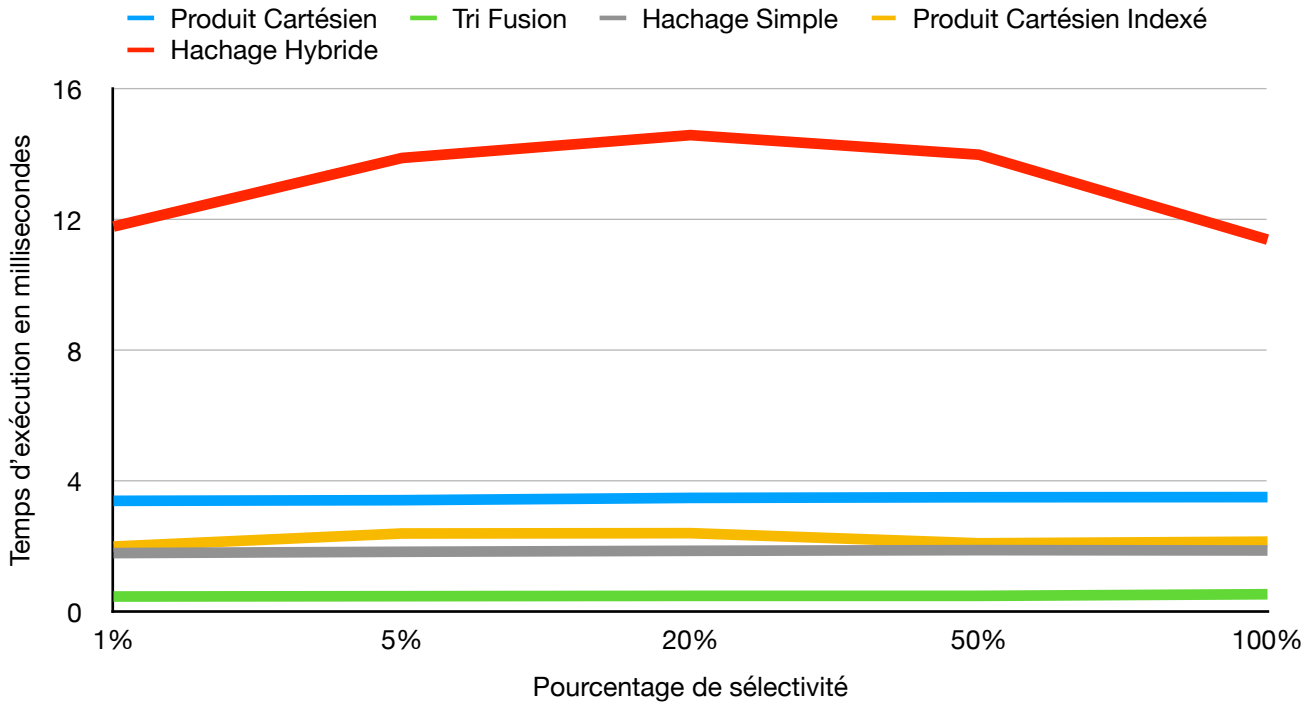
Analyse expérimentale des algorithmes de jointure:

Graphiques des performances des différents algorithmes:





Remarques:

I. Jointure par Produit Cartésien:

Le temps d'exécution est élevé et le coût de calcul peut être important pour de grandes relations, car le nombre de tuples dans le produit cartésien croît rapidement avec la taille des relations.

II. Jointure par Tri-Fusion:

Avantages:
- Bonne performance pour des relations déjà triées ou des petites relations.
- Peut fonctionner efficacement même avec des données non triées.

Inconvénient:
- Le tri effectué en pré-traitement peut être coûteux en temps et en ressources en présence de grandes relations.

III. Jointure par Hachage Simple:

Avantages:
- Les performances sont généralement bonnes peux importe la taille des données.

Inconvénient:
- Les performances peuvent être affectées si les données ne sont pas uniformément distribuées et entraînent des collisions de hachage (que deux clés distinctes ont la même position dans la table de hachage).

IV. Jointure par Produit Cartésien Indexé:

Avantages:
- Les performances sont généralement bonnes peux importe la taille des données.
- Peut gérer efficacement des relations non triées.

Inconvénients:
- La construction de l'index peut être coûteux en termes de ressources.
- Les performances peuvent être affectées s'il y a des collisions d'index.

V. Jointure par Hachage Hybride:

Avantages:
- Permet mieux gérer les jointures entre des ensembles de données de tailles différentes, car il utilise des techniques spécifiques à chaque jeu de données pour optimiser le processus de jointure.
- L'utilisation de plusieurs fonctions de hachage peut contribuer à réduire le nombre de collisions, ce qui améliorerait la précision et l'efficacité de la jointure.

Inconvénients:
- La mise en œuvre est plus complexe que celle d'un seul algorithme de hachage.
- L'utilisation de plusieurs fonctions de hachage peut augmenter la consommation de mémoire et de puissance de calcul.
- La configuration des différents algorithmes de hachage pour une jointure spécifique peut nécessiter des essais et des ajustements.

Conclusion:

Comme représentés par ces graphiques et dans mes remarques, c'est le Tri-Fusion qui semble être le plus efficace en termes de lectures/écritures disques ainsi qu'au niveau du temps d'exécution. Néanmoins plus les relations sont grandes plus celui ci augmente à cause du tri qui doit être effectué en pré-traitement.

L'autre algorithme qui semble se détacher des autres est le Hachage Simple qui engendre un peu plus de lectures/écritures disques et qui a un temps d'exécution plus élevé avec les données testées.

Vient ensuite le Produit Cartésien Indexé et le Hachage Hybride qui engendrent le même nombre de lectures/écritures disques que le Hachage Simple mais qui ont un temps d'exécution plus long.

Nous pouvons donc en conclure qu'en présence de grandes relations il faut généralement se tourner vers le Hachage Simple et qu'au contraire pour de petites relations ou des relations déjà triées il sera plus avantageux d'utiliser le Tri-Fusion.