Analyse théorique des algorithmes de jointure:

I. Jointure par Produit Cartésien:

Fonctionnement:

On suppose que l'on a deux relations: R(X,Y), S(Y,X), et que l'on applique la jointure suivante: T=Jointure(R,S,R.Y=S.Y)

1-lire la page i de R

2-lire la page j de S

3-effectuer les 4 matches possibles: R.Pi.Y0 ?= S.Pj.Y0

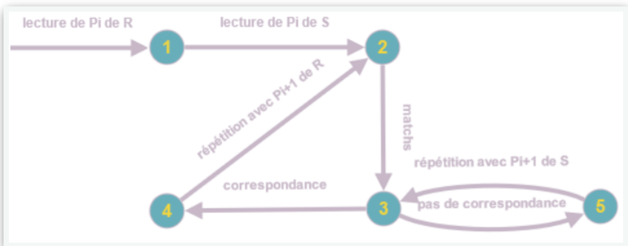
R.Pi.Y1 ?= S.Pj.Y0

R.Pi.Y1 ?= S.Pj.Y1

R.Pi.Y0 ?= S.Pj.Y1

4-s'il y a une correspondance alors l'écrire dans le résultat de la jointure (T.append(R.Pi.X0,R.Pi.Y0,S.Pi.Y0,R.Pi.X0) par exemple) et répéter les étapes 2,3,4,5 avec Pi+1 de R

5-si pas de correspondance alors lire Pj+1 de S et répéter les étapes 3,4,5



Graphe représentant le fonctionnement

de la jointure par Produit Cartésien

Nombre de lectures et écritures disques:

-taille(R) lectures pour R

-taille(R)*taille(S) lectures pour S

-min(taille(R),taille(S)) écritures pour T

Total=taille(R)+taille(R)*taille(S)+min(taille(R),taille(S)) lectures/écritures disque.

II. Jointure par Tri-Fusion:

On suppose que l'on a deux relations déjà triées (et sans doublon) par ordre croissant en fonction de à leur Y: R'(X,Y), S'(Y,X), et que l'on applique la meme jointure: T=Jointure(R',S',R'.Y=S'.Y)

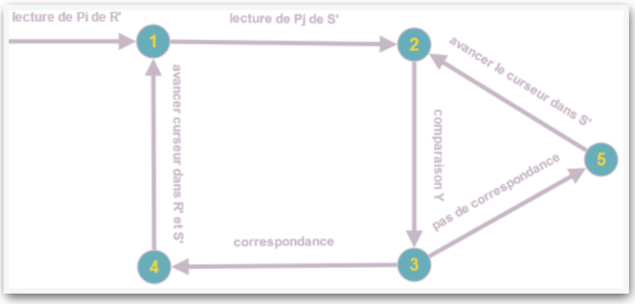
1-lire la page i de R'

2-lire la page j de S'

3- comparer R'.Pi.Y0 avec S.Pj.Y0 et R'.Pi.Y1 avec S.Pj.Y1

4-s'il y a une correspondance alors l'écrire dans le résultat de la jointure (T), avancer le curseur dans R' et dans S' et répéter les étapes 2,3,4,5.

5-s'il n y a pas de correspondance, si R'.Pi.Y0>S'.Pj.Y0 alors avancer le curseur dans S' (Y(1) ou passer à Pj+1 si déjà dans Y(1)) et répéter toutes les étapes.



Graphe représentant le fonctionnement

de la jointure par Tri-Fusion

Nombre de lectures et écritures disques:

-taille(R) lectures pour R

-taille(S) lectures pour S

-min(taille(R),taille(S)) écritures pour T

Total=taille(R)+taille(S)+min(taille(R),taille(S)) lectures/écritures disque.

III. Jointure par Hachage Simple:

Fonctionnement:

On suppose que l'on reprend les deux relations: R(X,Y), S(Y,X) utilisées pour le produit cartésien (par conséquent non triées). on suppose également que l'on a une table de hachage "hachage" avant autant d'entrée que de résultat possible de la fonction de hachage, et que l'on applique la meme jointure sur Y: T=Jointure(R,S,R.Y=S.Y)

Phase 1: Build

On suppose que l'on a choisit la fonction de hachage modulo 3.

1-lire Pi de R

2-pour chaque coupe (X,Y) calculer le modulo 3 de Y et l'écrire dans l'indice de la table de hachage correspondant (hachage[Y%3].append(R.Pi.X0,R.Pi.Y0,S.Pi.Y0,R.Pi.X0)) en formant des blocs de deux éléments, répéter ces étapes avec R.Pi+1

Le résultat obtenu est une nouvelle version de la relation R ou les Y ont été regroupés par la fonction de hachage.

Phase 2: Probe

1-lire la page i de S

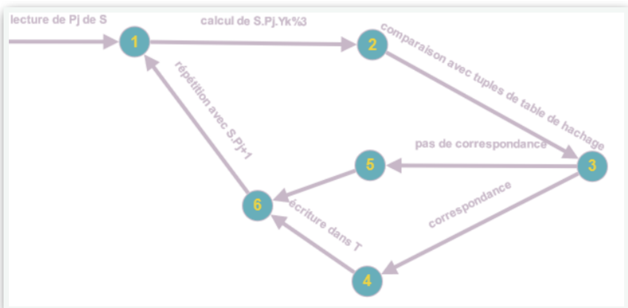
2-calculer S.Pj.Y0%3

3-chercher un tuple pour lequel Y vaut S.Pj.Y0%3 uniquement dans l'entrée S.Pj.Y0%3 de "hachage"

4-s'il y a une correspondance l'écrire dans T entrées

5-si pas de correspondance alors arrêter la recherche sans vérifier les autres entrées

6-reproduire les memes étapes avec S.Pj.Y1 puis avec S.Pj+1



Graphe représentant le fonctionnement

de la jointure par Hachage Simple

Nombre de lectures et écritures disques:

-taille(R) lectures pour R

-taille(R)*taille(S) lectures pour S

-min(taille(R),taille(S)) écritures pour T

Total=taille(R)+taille(R)*taille(S)+min(taille(R),taille(S)) lectures/écritures disque.

IV. Comparaison:

Après avoir effectué une analyse théorique des différents algorithmes de jointures nous pouvons en conclure qu'en terme de lectures/écritures disques ce sont les algorithmes Tri-Fusion et Hachage simple qui sont les plus efficaces.

En effet, leur exécution engendre le meme nombre de lectures/écritures alors que le Produit Cartésien en engendre plus.