L3 Informatique, UE DGINF363 (ALGO6, Algorithmique et modélisation)

Vendredi 3 février : utilisation de tables de hachage (jointure, soustraction,...)

Ce travail est à faire en binôme. Le compte rendu (un par binôme, indiquant vos noms et groupes de TD), est à déposer avant vendredi 3 février à 17h sur la page Moodle du cours.

Objectifs

Ce travail est une illustration de l'utilisation de tables de hachage pour gérer efficacement des ensembles de valeurs. On va comparer des versions naïves (à base de boucles imbriquées) à des versions basées sur l'utilisation de tables de hachage.

Ressources fournies

- `EssaiJoin.java`, programme principal.
- `Joiner.java`, implémentations de l'opérateur de jointure (cf. vos cours de *BDBC*).
- `tests.tar.gz`, une archive contenant des fichiers de tests. Cette archive contient :
  - Un fichier `LesVins` contenant les n-uplets d'une relation *LesVins(nom, degré, qualité)* (Rq : seules les valeurs de l'attribut *nom* sont "réelles", les autres valeurs ont été fixées aléatoirement).
  - Un fichier `LesAchats` contenant les n-uplets d'une relation *LesAchats(num, nom, quantité)*. Dans un tuple *<nu,no,q>* de cette table *nu* représente l'identifiant de l'achat qui concerne le vin *no* en quantité *q*.
  - Un fichier (pour vos tests) `LesAchats_petit` qui contient un petit nombre de n-uplets de la relation *LesAchats*.

Exercice 1. Compréhension du programme fourni

Le programme Java `EssaiJoin.java` contient l'implémentation d'un algorithme naïf de jointure (naturelle). Ce programme prend en argument une liste de fichiers. Les deux premiers pour indiquer les fichiers où se trouve les tables à joindre, les deux suivants pour indiquer dans quels fichiers les résultats des jointures doivent être écrits.

Lisez `EssaiJoin.java` et `Joiner.java`.

Essayez ce programme sur quelques exemples simples (à construire vous-même à partir des fichiers de données fournis, les commandes unix `tail` et/ou `head` peuvent être utiles).

Exercice 2. Coût de l'algorithme naïf

Ecrire sous une forme concise l'agorithme naïf. Donner sa complexité en fonction du nombre de n-uplets de chacune des tables.

Exercice 3. Algorithme de Jointure par hachage (Hash Join)

Implémentez la jointure par hachage dans la procédure `HashJoin`.

L'algorithme consiste à :

- En premier lieu, parcourir l'une des deux relations pour la stocker dans une table de hachage. Dans notre cas, la relation *LesVins* est la plus appropriée.
- Ensuite on parcours la deuxième relation (*LesAchats*), pour chacun de ses tuples, on cherche dans la table de hachage le tuple de la première relation auquel il doit être join.

Dans votre implementation, vous devez vous aider de la fonction `HashCode()` et de la class Hashtable de Java.

Exercice 4. Évaluation des performances de l'algorithme de HashJoin

Exécutez le programme complété pour différentes tailles de `LesAchats`. Notez (ou modifier le programme pour le faire) le temps obtenu en fonction de la taille de cette table.

Sur un même graphique, tracez le temps d'exécution en fonction de la taille de `LesAchats` pour les deux algorithmes.

Exercice 5. Soustraction et élimination des doublons.

Écrivez deux versions pour l'opération de projection sur l'attribut *nom* sans doublons (*select distinct nom from LesAchats*). La première version en utilisant deux boucles imbriquées, la deuxième en utilisant une table de hachage (même graphique que pour la question 4).

Écrivez deux versions pour l'opérateur de soustraction (les vins qui ne sont par dans les achats : *(select nom from LesVins) Minus (select nom from LesAchats)*). La première version en utilisant deux boucles imbriquées la deuxième en utilisant une table de hachage (même graphique que pour la question 4).

Compte-rendu

Le compte-rendu doit être déposé au format PDF. Il devra comporter, **sur 3 pages maximum** :

- une introduction résumant le travail effectué ;
- les réponses à l'exercice 2. ;
- les graphiques obtenus aux l'exercices 4 et 5., ainsi que leur interprétations.

Le fichier `Joiner.java` complété doit être joint à ce compte-rendu.

▶ Consignes pour le dépot des fichiers :

- Les fichiers doivent être déposés séparément sur Moodle, et non groupés dans une archive.
- Le nom du fichier du compte-rendu doit contenir le nom des deux étudiants du binôme.

- Les signatures des fonctions fournies ne doivent pas être modifiées.

Annexe : Gnuplot

Pour tracer les courbes, vous pouvez utiliser le logiciel `gnuplot`, **disponible sur le serveur `mandelbrot`**. On donne ci-dessous quelques indications pour utiliser ce logiciel, sachant que des informations complémentaires sont disponibles ici.

- Format des données : les couples de coordonnées des points à afficher doivent être fournies dans un fichier texte, un couple par ligne, séparés par un (ou plusieurs) espace(s). Ces coordonnées peuvent être entières ou réelles.
- Le logiciel gnuplot se lance simplement avec la commande `gnuplot` qui a pour effet de démarrer une session interactive. La commande `quit` permettra alors de terminer cette session.
- Une fois une session interactive démarrée (l'invite `gnuplot>` s'affiche) il est possible de tracer la courbe correspondant aux points fournis dans un fichier `data.txt` :

```
gnuplot> plot "data.txt" with linespoints
```

  On peut également afficher une seconde courbe dans le même repère, par exemple :

  ```
  gnuplot> replot "data2.txt" with linespoints
  ```

- Il est possible (et souhaitable...) de nommer les axes et légender les courbes :

```
gnuplot> set xlabel "nom de l'axe x"
gnuplot> set ylabel "nom de l'axe y"
gnuplot> plot "data1.txt" with linespoints title "Légende pour le fichier data1.txt'
             "data2.txt" with linespoints title "Légende pour le fichier data2.txt'
```

- Enfin, il est possible de diriger la sortie de gnuplot dans un fichier image PNG (ou PostScript, ou PDF, ...) :

```
gnuplot> set term png
gnuplot> set out "resultat.png"
gnuplot> plot "data1.txt" with linespoints title "Légende pour le fichier data1.txt'
              "data2.txt" with linespoints title "Légende pour le fichier data2.txt'
```