TD Mécanismes Internes SGBD

Exercices : Implémentation d'opérateurs

Exercice 1 : Fonction AVG

1) Donnez le pseudo-code, composé de la méthode open(), next() et close(), permettant d'implémenter la fonction AVG. On supposera qu'on passera en paramètre un opérateur quelconque OP1 sur lequel on peut appeler les méthodes open(), next() et close, et pour lequel la méthode next() retourne le prochain n-uplet disponible.
   Est-ce que votre opérateur est bloquant ?

Exercice 2 : Fonction DISTINCT en mémoire

2) Donnez le pseudo-code de l'opérateur `DISTINCT`, qui de même prend en paramètre un opérateur quelconque `OP1` sur lequel on peut appeler les méthodes open(), next() et close(). Dans un premier temps, vous pourrez faire l'hypothèse que l'ensemble des données produites par `OP1` tient en mémoire. Pouvez-vous réaliser cet opérateur de manière non bloquante ?

Exercice 3 : Fonction DISTINCT sans contrainte de mémoire

3) Dans un deuxième temps, on fait l'hypothèse que ce n'est pas le cas. On suppose qu'on dispose d'un opérateur déjà implémenté `TriExterne`, comme dans le cours. Proposez une implémentation de l'opérateur `DISTINCT` à l'aide de `TriExterne` et qui fonctionnera même si l'ensemble des résultats produits par `OP1` ne tient pas en mémoire.

Exercice 4 : Implémentation d'opérateurs de jointure : Boucle Imbriquée

4) On souhaite réaliser un opérateur de jointure simple, qui prend comme paramètre 2 tables T1 et T2, et qui produit en sortie (pour chaque appel de `next()`) un n-uplet dans la jointure. On propose d'utiliser l'implémentation Double Boucle Imbriquée, c'est-à-dire qu'on parcourt l'une des tables T1, puis pour chaque entrée dans T1 on parcourt l'intégralité de la table T2. On utilisera dans un premier temps l'opérateur FullScan pour parcourir les tables. Donnez l'implémentation en pseudo-code de cet opérateur. Est-ce que votre opérateur est bloquant ? Donnez sa complexité en nombre de lectures sur le disque.
5) Supposons que T1 soit indexée sur l'attribut T1.A. Il est donc possible de parcourir T1 en utilisant les opérateurs `IndexScan` et `DirectAccess` (voir cours), en supposant que la jointure soit entre T1.A et T2.A. Proposez une implémentation qui parcourt T2 en utilisant FullScan, et qui accède à T1 via son index. Est-ce que votre opérateur est bloquant ? Donnez sa complexité en nombre de lectures sur le disque (attention à bien évaluer le paramètre de complexité pertinent).

Exercice 5 : Implémentation d'opérateurs de jointure : SortMergeJoin

6) On souhaite réaliser un opérateur de jointure de la manière suivante : on utilise TriExterne pour trier les deux tables T1 et T2, qu'on supposera non indexées. Puis on parcourt les deux listes triées en produisant les n-uplets dans la jointure. Est-ce que votre opérateur est bloquant ? Est-ce qu'il y a une contrainte sur la mémoire nécessaire pour faire tourner votre algorithme, et si oui laquelle ? Donnez la complexité de votre opérateur en terme de lectures sur le disque.