TD6 : Optimisation de requêtes

UCBL - Département Informatique de Lyon 1 – BDW - automne 2024

Objectif du TD : optimiser une requête algébrique, comparer des plans d'exécution

Ce jeu de données décrit des réservations dans les refuges de montagne par des randonneurs. Ci-dessous le schéma relationnel et quelques caractéristiques des instances et des attributs (tableaux 1 et 2). Les tailles sont données en octets. Enfin, les hypothèses suivantes s'appliquent :

- La distribution des valeurs d'un attribut est uniforme dans son domaine ;
- Les valeurs d'attributs sont indépendantes (il n'y a pas de corrélation entre les attributs).

Schéma relationnel :

RANDONNEURS (idRa, *nomRa*, *prénomRa*, *adresseRa*, *tél*)
REFUGES (idRe, *nomRe*, *adresseRe*, *catégorie*, *coordonnées*, *nbLits*)
RÉSERVATIONS (#*idRa*, #*idRe*, dateR, *durée*)

Instances	Nombre	Taille
Refuges	1000	302
Randonneurs	2000	230
Réservations	5000	22

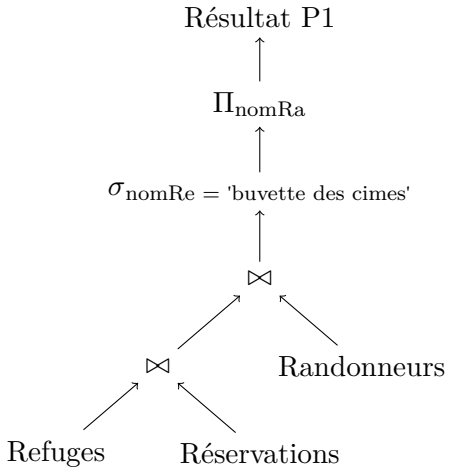
Tableau 1 : caractéristiques des instances

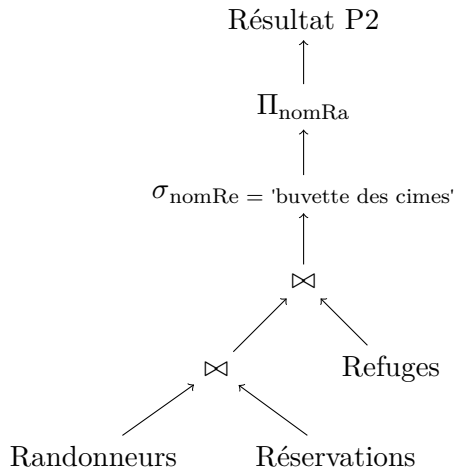
Attribut	Nombre valeurs distinctes	Taille
idRa	2000	5
nomRa	1950	50
idRe	1000	5
nomRe	1000	50

Tableau 2 : caractéristiques des attributs

Exercice 1 Optimisation de requête

1. Traduire la requête suivante en SQL (avec des jointures naturelles) : *le nom des randonneur·euse·s qui ont fait une réservation dans le refuge "la buvette des cimes".*
2. Le SGBD a compilé la requête SQL précédente en deux arbres algébriques P1 et P2 (ci-dessous). Cette traduction vous semble-t-elle correcte ?

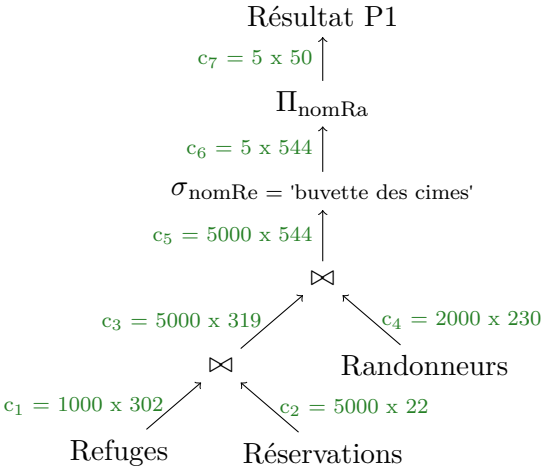




3. L'une des intuitions pour optimiser consiste à réduire le nombre d'attributs (projection) et réduire le nombre de n-uplets (sélection) avant de réaliser des jointures. On peut aussi réordonner les jointures. Choisissez l'un des deux arbres et optimisez le.

Exercice 2 Calcul du coût d'exécution d'un plan

Nous nous intéressons maintenant à la sélection du plan à appliquer, ce qui nécessite de calculer et comparer le coût de chaque plan d'exécution. Pour mettre en avant les avantages de l'optimisation, nous allons comparer le coût du plan P1 et celui du plan optimisé.
L'arbre P1 ci-contre est annoté avec des coûts intermédiaires (c_1 à c_7). Il s'agit ici de coûts bruts (théoriques) sans considérer d'autres mécanismes (comme la composition d'opérations ou le cache).
Dans la suite, les hypothèses suivantes s'appliquent : la taille d'un bloc disque est de 1000 octets, la mémoire peut contenir 100 blocs, et le système peut lire 10000 blocs par seconde.



Pour la suite de l'exercice, les jointures sont effectuées avec l'algorithme de boucles imbriquées avec bloc (*"block nested loops"*). Pour rappel, la formule du coût d'une jointure avec cet algorithme est $\frac{k_R}{m-1} \times (m - 1 + k_S)$ où m est le nombre de blocs en mémoire, k_R le nombre de blocs pour la plus petite table, et k_S pour la plus grande table. La sélection s'effectue par itération (*"scan"*) de la table, et les projections sont réalisées en même temps qu'une autre opération (coût nul).

1. Expliquez comment ont été calculés les coûts intermédiaires c_1 à c_7 du plan P1.
2. Pour calculer le coût du plan P1, il faut calculer le coût des 2 jointures et de la sélection. Voici le calcul du coût de la première jointure :
   - Nombre de blocs pour lire les n-uplets de Refuges : $\frac{1000 \times 302}{1000} = 302$ blocs
   - Nombre de blocs pour lire les n-uplets de Réservations : $\frac{5000 \times 22}{1000} = 110$ blocs
   - Application de la formule : $\frac{110}{99} \times (99 + 302) \approx 446$ blocs pour exécuter la première jointure

   Terminez le calcul du coût du plan d'exécution P1 en nombre de blocs puis en secondes.
3. Calculez le coût du plan d'exécution optimisé dans l'exercice 1.
4. Les deux arbres étant équivalents (résultat identique), que peut-on en conclure ?