

# Algorithmes pour la sélection des vues à matérialiser avec garantie de performance<sup>1</sup>

Nicolas Hanusse\*, Sofian Maabout\*  
Radu Tofan\*

\*LaBRI. Université Bordeaux 1  
nom@labri.fr

**Résumé.** Nous considérons dans cet article le problème du choix des vues à matérialiser dans le contexte des cubes de données. Contrairement à la plupart des autres travaux, la contrainte que l'on prend en compte ici est relative à la performance des requêtes non pas l'espace mémoire disponible. Nous montrons l'avantage de notre approche avec des arguments théoriques et nous le confirmons par des expériences menées sur des jeux de données différents.

## 1 Introduction

Dans les applications OLAP, l'optimisation des requêtes analytiques est primordial. Ainsi, on a souvent recours à un pré-calcul ou d'une manière équivalente à la matérialisation des requêtes. Ces dernières peuvent être trop nombreuses pour être toutes pré-calculées. Ceci pour deux raisons : soit à cause du temps qu'il faut pour toutes les calculer ou bien de l'espace mémoire nécessaire pour toutes les stocker. Ainsi, on a besoin d'algorithmes permettant de choisir les "*meilleures*" vues à stocker. Les performances des algorithmes de sélection sont évalués selon trois critères : (1) l'espace requis pour stocker les vues sélectionnées, (2) le temps d'évaluation des requêtes et (3) le temps nécessaire pour exécuter l'algorithme, i.e sa complexité. La plupart des solutions proposées jusqu'à présent consistent à choisir un sous-ensemble de vues permettant d'optimiser l'évaluation des requêtes sans dépasser une limite d'espace mémoire fixée par l'utilisateur. Certaines variantes de ce problèmes ont été étudiées : (1) la nature des données qu'on peut stocker, e.g seulement des vues ou des index aussi, (2) le modèle de coût e.g minimiser non seulement le temps d'évaluation des requêtes mais aussi la coût de mise à jour ou finalement (3) l'ensemble des requêtes susceptibles d'être posées e.g toutes ou bien un sous-ensemble. Dans ce dernier cas, le problème peut être raffiné en considérant la fréquence ou *charge* des requêtes qu'on veut optimiser. En général, le problème peut être décrit par un problème d'optimisation sous contrainte : la contrainte étant l'espace disponible qu'il ne faut pas dépasser et la fonction à optimiser est le coût moyen des requêtes qu'il faut minimiser. A notre connaissance, aucune des solutions proposées dans la littérature n'offre un bon compromis entre complexité de l'algorithme de sélection, minimalité du coût moyen de la solution et garantie de performance.

L'article est organisé de la manière suivante : nous présentons d'abord quelques notations et définitions nous permettant de formaliser le problème à traiter. Nous décrivons par la suite quelques travaux relatifs. Dans la section 3, nous présentons notre solution pour cas où tous