Cubes de Données Convexes Non-Dérivables Fermés

Hanen Brahmi, Sadok Ben Yahia

Faculté des Sciences de Tunis.
Département des Sciences de l'Informatique.
Campus Universitaire 1060.
hanenbrahmi@gmail.com, sadok.benyahia@fst.rnu.tn

Résumé. De nombreuses approches sont proposées pour pré-calculer des cubes de données afin de répondre efficacement aux requêtes OLAP. La notion de cube de données a été déclinée sous différentes appellations: cubes icebergs, cubes différentiels ou encore cubes émergents. Les cubes convexes permettent de focaliser l'attention de l'utilisateur sur un ensemble particulier de tuples intéressants. Dans cet article, nous étudions les représentations concises des cubes convexes. À cet effet, nous introduisons une nouvelle structure d'un cube de données: le Cube Convexe Non-Dérivable Fermé ($\mathcal{CCND}$-$\mathcal{C}$ube). Ce dernier permet de capturer tous les tuples d'un cube de données satisfaisant une combinaison de contraintes monotones et/ou antimonotones. Les expériences montrent que notre proposition fournit la représentation la plus compacte d'un cube de données de manière à optimiser à la fois le temps de calcul ainsi que l'espace de stockage nécessaire.

1 Introduction

La gestion des grandes masses de données est devenue une tâche difficile et assez coûteuse à maintenir. Ce problème a conduit aux développements *des entrepôts de données*. Ces derniers ont apporté une solution adéquate et efficace au problème de la croissance continuelle des données. Le contenu d'un entrepôt est analysé par les applications *On Line Analytical Processing* (OLAP), qui fournissent aux utilisateurs des moyens pour naviguer dans les données multidimensionnelles afin d'y découvrir des connaissances interprétables, exploitables et utiles à la prise de décision (Chaudhuri et Dayal, 1997).

En pré-calculant tous les agrégats possibles à différents niveaux de granularité, les cubes de données permettent une réponse efficace aux requêtes OLAP et sont donc un concept clef pour la gestion des entrepôts (Chaudhuri et Dayal, 1997). Étant donné un contexte d'extraction R contenant n attributs, le nombre de tuples dans un cube à k-attributs ($0 < k \leq n$), est le nombre de tuples dans R qui ont des valeurs d'attributs distinctes pour les k attributs. La taille d'un cuboïde est presque égale à la taille de R. Puisque le cube complet, construit à partir de R, consiste en 2^n cuboïdes, alors la taille de l'union des 2^n cuboïdes est beaucoup plus élevée que la taille de R.

Il est bien connu que le calcul des cubes de données est un problème combinatoire. En effet, la taille d'un cube augmente exponentiellement en fonction du nombre des dimensions.

En outre, le problème s'aggrave, puisque nous traitons des jeux de données volumineux. Dans ce cadre, Ross et Srivastava donnent un exemple de ce problème en calculant un cube de données complet englobant plus que 210 millions de tuples à partir d'une relation initiale ayant 1 million de tuples (Ross et Srivastava, 1997). Généralement, le problème est dû aux deux raisons suivants : le nombre exponentiel de combinaisons des dimensions et le nombre d'attributs par dimension. De plus, les cubes de données sont généralement épars (Ross et Srivastava, 1997). Ainsi, en calculant un cube de données complet, les combinaisons de valeurs infréquentes vont probablement être nombreuses. Dans un tel contexte, nous pouvons distinguer : (i) les approches favorisant l'efficacité des requêtes OLAP malgré l'espace de stockage ; et (ii) ceux favorisant des représentations optimales des cubes de données au lieu d'améliorer les performances des requêtes.

Bien que les contraintes liées à la taille des cubes de données aient attiré l'attention des chercheurs et divers algorithmes ont été développés visant un calcul rapide des cubes de données volumineux, relativement peu de travaux se sont intéressés à la résolution du problème de la complexité de calcul des cubes de données à partir de la racine : la réduction de la taille d'un cube de données. Dans cet article, nous examinons une autre façon d'attaquer le problème. D'abord, nous introduisons une nouvelle structure appelée *cube convexe non-dérivable fermé*, qui s'appuie sur l'espace de recherche du treillis cube. Notre représentation prend en compte des combinaisons de contraintes monotones et/ou antimonotones. Ensuite, nous introduisons un algorithme pour calculer efficacement ces cubes convexes non-dérivables fermés. Enfin, nous montrons l'efficacité de notre approche à travers une étude expérimentale approfondie menée sur des bancs d'essai et des bases réelles. Ces expérimentations portent à la fois sur l'efficacité de algorithme de calcul de la représentation et sur l'espace de stockage qui lui est nécessaire comparée aux approches s'inscrivant dans la même tendance.

Le reste de l'article est organisé comme suit. La section 2 introduit les fondements mathématiques qui seront utilisés. Dans la section 3, nous passons en revue les travaux antérieurs. Nous définissons les concepts de notre représentation dans la section 4. Nous introduisons l'algorithme dfCLOSND dans la section 5. Les résultats des expérimentations montrant l'utilité de l'approche proposée sont présentés dans la section 6. La conclusion et les travaux futurs font l'objet de la section 6.

2 Fondements mathématiques

L'ensemble des motifs fermés, basé sur le concept de fermeture, constitue une représentation concise de l'ensemble des motifs fréquents (Pasquier et al., 1999).

Définition 1 (Motif Fermé) . *La fermeture γ d'un motif X est le sur-ensemble maximal de X ayant la même valeur du support que celui de X. Un motif X est fermé si seulement si $X = \gamma(X)$.*

Le concept de générateur minimal (Bastide et al., 2000) est défini comme suit.

Définition 2 (Générateur Minimal) . *Un motif g est dit générateur minimal ($\mathcal{GM}$) d'un motif fermé f, ssi $\gamma(g) = f$ et $\nexists\ g_1 \subset g$ tel que $\gamma(g_1) = f$. Pour un seuil minimal de support, MinSeuil donné par l'utilisateur, l'ensemble des générateurs minimaux fréquents inclut tous les générateurs qui sont fréquents.*

La collection des motifs fréquents non-dérivables, notée $\mathcal{MND}$, est une représentation concise et exacte des motifs fréquents basée sur le principe d'inclusion-exclusion (Calders et Goethals, 2007).

Définition 3 (Motif Non-Dérivable) . *Soit X un motif et Y un sous-ensemble de X. Si $|X \backslash Y|$ est impair, alors la règle de déduction correspondante à une borne supérieure pour le Supp(X) est :*

$$Supp(X) \leq \sum_{Y \subseteq I \subset X} (-1)^{|X \backslash I| + 1} Supp(I)$$

Si $|X \backslash Y|$ est pair, le sens de l'inégalité est inversé et la règle de déduction donne une borne inférieure au lieu d'une borne supérieure du support de X. Étant donné tous les sous-ensembles de X et leurs supports, nous obtenons un ensemble de bornes supérieures et inférieures pour X. Dans le cas où la plus petite borne supérieure est égale à la borne inférieure la plus élevée, le support de X est exactement dérivé. Un tel motif s'appelle *dérivable*. Par la suite, les bornes inférieures et supérieures du support d'un motif X seront respectivement notées par $X.l$ et $X.u$.

Brahmi et al. (2012) ont introduit le concept des générateurs minimaux non-dérivables.

Définition 4 (Générateurs Minimaux Non-Dérivables) . *Étant donné un motif $I \subseteq \mathcal{I}$, l'ensemble des générateurs minimaux non-dérivables est défini comme suit : $\mathcal{GMND} = \{I \subseteq \mathcal{I} \mid I.l \neq I.u$ et I est un $\mathcal{GM}\}$.*

L'ensemble des motifs fermés fréquents non-dérivables, noté $\mathcal{MFND}$, a été introduit par Muhonen et Toivonen (2006). Récemment, Brahmi et al. (2012) ont introduit une nouvelle définition qui allie à la fois la notion du motif fermé fréquent et celle du générateur minimal non-dérivable.

Définition 5 (Motif Fermé Non-Dérivable) . *Soit $\mathcal{GMND}$ l'ensemble des générateurs minimaux fréquents non-dérivables. L'ensemble des motifs fermés fréquents non-dérivables basés sur les générateurs minimaux est $\mathcal{MFND} = \{\gamma(X) \mid X \subset \mathcal{GMND}\}$.*

Casali et al. (2003) ont fixé un cadre théorique et général pour l'OLAP et la fouille de bases de données multidimensionnelles.

Définition 6 (Treillis Cube) . *L'espace multidimensionnel d'un contexte d'extraction R est noté et défini comme suit : Space$(R) = \{X_{A \in D}(Dim(A) \bigcup \texttt{ALL}) \bigcup (\emptyset \dots \emptyset)\}$, où X symbolise le Produit cartésien, $(\emptyset \dots \emptyset)$ le majorant universel et Dim(A) la projection de R sur A. Toute combinaison de Space(R) est un tuple et représente un motif multidimensionnel.*

L'espace multidimensionnel de R, Space(R), est structuré par la relation de généralisation/spécialisation entre tuples, notée $\preceq$. Soient u, v deux tuples de l'espace multidimensionnel de R. Si $u \preceq v$, alors nous disons que u est plus général que v dans Space(R). Les deux opérateurs de base définis pour la construction de tuples sont la Somme (notée $+$) et le Produit (noté $\bullet$). La somme de deux tuples retourne le tuple le plus spécifique généralisant les deux opérandes. Le produit de deux tuples retourne le tuple le plus général spécialisant les deux opérandes. En dotant l'espace multidimensionnel Space(R) de la relation de généralisation entre tuples et en utilisant les opérateurs Produit et Somme, on obtient un treillis cube noté par : CL(R) = (Space(R), $\preceq$).

Nous rappelons les définitions de (Cicchetti et al., 2012) pour la notion d'espace convexe, des contraintes monotones et antimonotones selon l'ordre de généralisation $\preceq$.

Définition 7 (Espace Convexe) . *Soit $(\mathcal{P}, \leq)$ un ensemble partiellement ordonné, $\mathcal{C} \subseteq \mathcal{P}$ est un espace convexe (Vel, 1993) si et seulement si $\forall x, y, z \in \mathcal{P}$ tel que $x \leq y \leq z$ et $x, z \in \mathcal{C}$ alors $y \in \mathcal{C}$.*

Définition 8 (Contrainte Monotone/antimonotone) .

1. *Une contrainte Cont est dite monotone pour l'ordre de généralisation si et seulement si : $\forall t, u \in CL(R) : [t \preceq u$ and $Cont(t)] \Rightarrow Cont(u)$.*
2. *Une contrainte Cont est dite antimonotone pour l'ordre de généralisation si et seulement si : $\forall t, u \in CL(R) : [t \preceq u$ and $Cont(u)] \Rightarrow Cont(t)$.*

3 État de l'art

De nombreux travaux de recherche ont abordé la problématique d'une représentation concise (ou condensée) des cubes de données permettant d'en réduire notablement la taille. Les approches, qui choisissent de ne pas restituer les données exactes ou complètes se basent sur le fait que l'utilisateur d'un entrepôt s'intéresse aux grandes tendances générales dans la "population" examinée (Casali et al., 2009b). Certaines s'appuient sur la structure statistique des données pour calculer des distributions de densité et répondre de manière approximative aux requêtes OLAP, l'enjeu étant bien sûr d'obtenir la meilleure approximation possible. L'idée sous-jacente de cette tendance d'approches est d'intégrer, dès le calcul du cube, les contraintes antimonotones des utilisateurs potentiels, de manière à ne calculer et ne stocker que les agrégats correspondant à des tendances suffisamment générales pour être pertinentes.

En s'inspirant des motifs fréquents, les algorithmes BUC (Beyer et Ramakrishnan, 1999) et HCUBING (Han et al., 2001) calculent des résultats exacts mais incomplets : le cube de données partiel ou iceberg cube. Ces derniers sont présentés comme des sous-ensembles de tuples du cube de données satisfaisant, pour les valeurs de la mesure, une contrainte de seuil minimum. Pour ce faire, ils ajoutent une clause HAVING dans la requête. Il s'agit dans ce cas d'exhiber les tendances suffisamment générales pour être pertinentes pour le décideur, il en découle deux intérêts techniques importants : ne pas calculer ni matérialiser la totalité du cube d'où un gain notable à la fois de temps d'exécution et d'espace disque.

De plus, les nouvelles tendances apparaissant (ou les tendances avérées disparaissant) lors du rafraîchissement d'un entrepôt sont mises en évidence par le calcul de cubes différentiels (Casali, 2004). Ceux-ci peuvent être perçus comme la différence ensembliste entre deux cubes : l'un stocké dans l'entrepôt et l'autre calculé à partir des données de rafraîchissement. Suivant l'ordre des deux opérandes, les tendances disparaissantes ou apparaissantes sont exposées. Ils mettent en évidence des tuples pertinents dans un des cubes et inexistants dans l'autre. Leur intérêt est donc de pouvoir comparer des tendances entre deux jeux de données.

Dans le même contexte, les cubes émergents (Nedjar et al., 2011) exhibent des tendances non pertinentes pour l'utilisateur ou au contraire des tendances significatives, qui s'atténuent sans forcément disparaître. Les cubes émergents permettent donc d'élargir les résultats des cubes différentiels en affinant les comparaisons entre deux cubes. Ils sont aussi intéressants

dans un contexte OLAP pour l'analyse des flots de données puisqu'ils mettent en évidence des renversements de tendances.

Le cube convexe (Cicchetti et al., 2012) s'appuie sur l'espace de recherche du treillis cube. Il prend en compte des combinaisons de contraintes monotones et/ou antimonotones. Cette structure est un espace convexe (Vel, 1993) qui peut être représenté par ses bordures. D'où, le cube convexe reflète un cadre formel générique permettant de caractériser, de manière simple et solide, différentes variantes de cubes de données, trop souvent perçus comme les résultats de requêtes ou d'algorithmes et non comme des concepts.
Finalement, il existe deux approches ayant choisi de caractériser des représentations condensées du cube convexe :
- **Le cube convexe quotient** (Cicchetti et al., 2012) peut être efficacement construit et réalise une réduction significative de la taille du cube convexe. L'idée principale derrière un cube convexe quotient est de créer un résumé en partitionnant soigneusement l'ensemble des cellules d'un cube de données selon des classes d'équivalences. Une classe d'équivalence contient l'ensemble des cellules ayant la même fermeture et respectant une conjonction de contraintes monotones et/ou antimonotones. Le partitionnement des cellules se fait tout en gardant la sémantique ROLLUP et DRILLDOWN et la structure de treillis.
- **Le cube convexe fermé** (Cicchetti et al., 2012) offre une représentation de taille réduite d'un cube convexe comparée au cube convexe quotient. Il est composé, uniquement, de cellules fermées. Une cellule c, est dite une *cellule fermée* s'il n'y a aucune cellule d, telle que d est une spécialisation (descendante) de c, qui a la même mesure que c. Sachant que cette mesure satisfait une combinaison de contraintes monotones et/ou antimonotones.

Vu son importance, la réduction de l'espace de stockage d'un cube de données sur le disque ne cesse de présenter une problématique faisant l'objet de plusieurs recherches. À cet égard, l'intérêt principal de cet article est de proposer une nouvelle structure, appelée *cube convexe non-dérivable fermé* et notée $\mathcal{CCND}$-$\mathcal{C}$ube. En effet, nous essayons d'extraire un $\mathcal{CCND}$-$\mathcal{C}$ube, qui permet d'obtenir la représentation des données multidimensionnelles la plus petite afin de réduire considérablement l'espace de stockage sur le disque. En outre, nous appliquons un mécanisme simple qui réduit, d'une manière significative, la taille des agrégats à stocker. Notre but est de calculer la plus petite représentation concise d'un cube convexe, comparée aux approches existantes (*i.e.*, cube convexe fermé, cube convexe quotient). Pour construire le $\mathcal{CCND}$-$\mathcal{C}$ube, nous introduisons un nouvel algorithme appelé dfCLOSND. Ce dernier adopte une stratégie de recherche en profondeur (diviser-pour-régner) en vue d'extraire des motifs non-dérivables fermés en se basant sur les générateurs minimaux.

4 Structure du cube convexe non-dérivable fermé

L'idée de base de notre représentation est d'éliminer les redondances existantes au sein des cubes convexes. Effectivement, certains tuples véhiculent la même sémantique que d'autres qui sont plus agrégés. En fait, les uns et les autres sont construits en agrégeant exactement les mêmes tuples de la relation originale mais à des niveaux de granularité différents. Par conséquent, un seul tuple (le plus spécifique de tous) peut représenter tout l'ensemble. L'opérateur de fermeture (*cf.* Definition 1) permet de calculer ce tuple représentatif.

Le cube non-dérivable fermé, englobant l'ensemble des tuples fermés non-dérivables, est actuellement une des représentations les plus réduites pour le cube de données (Brahmi et al.,

2012). Il est donc intéressant de proposer, pour le cube convexe, une structure s'appuyant sur les concepts associés au cube non-dérivable fermé. Nous introduisons une représentation condensée du cube convexe avec un double objectif : (i) définir d'une manière compacte l'espace de solutions et décider si un tuple t appartient, ou pas, à cet espace ; et (ii) obtenir une représentation condensée du cube convexe en présence d'une conjonction de contraintes.

RowID	Modèle	Ville	Client	Couleur	Quantité
1	Ford	Sousse	Auto	Gris	400
2	Ford	Sousse	LeMoteur	Noir	100
3	Peugeot	Beja	LeMoteur	Gris	100
4	Peugeot	Sousse	Auto	Noir	300
5	Peugeot	Beja	Auto	Gris	200

TAB. 1 – *Relation exemple* VOITURE

Nous prenons en compte les contraintes monotones et/ou antimonotones les plus couramment utilisées en fouille de base de données. Celles-ci portent sur : (1) des mesures d'intérêts comme la fréquence de motifs, la confiance, la corrélation : dans ce cas, seuls les attributs dimensions de R sont nécessaires ; (2) des agrégats selon des attributs mesures calculés en utilisant des fonctions statistiques additives (COUNT, SUM, MIN, MAX) ; (3) des motifs respectant deux contraintes antimonotones "*être non-dérivable*" (*cf.* Definition 3) et "*être un générateur minimal*" (*cf.* Définition 2).

Tuples Non-Dérivables Fermés	**COUNT(*Quantité*)**	**Tuples Non-Dérivables Fermés**	**COUNT(*Quantité*)**
(ALL, ALL, ALL, Gris)	700	(Peugeot, ALL, Auto, ALL)	500
(ALL, Sousse, ALL, ALL)	800	(Peugeot, Beja, ALL, Gris)	300
(ALL, ALL, LeMoteur, ALL)	200	(Peugeot, Sousse, Auto, Noir)	300
(ALL, ALL, Auto, ALL)	900	(Peugeot, Beja, LeMoteur, Gris)	100
(ALL, ALL, Auto, Gris)	600	(Peugeot, Beja, Auto, Gris)	200
(ALL, Sousse, ALL, Noir)	400	(Ford, Sousse, ALL, ALL)	500
(ALL, Sousse, Auto, ALL)	700	(Ford, Sousse, Auto, Gris)	400
(Peugeot, ALL, ALL, ALL)	600	(Ford, Sousse, LeMoteur, Noir)	100

TAB. 2 – *Cube non-dérivable fermé issu de la relation* VOITURE

Définition 9 (Cube de Données Non-Dérivable Fermé) . *Soit $\mathcal{GMND}$ l'ensemble des motifs générateurs minimaux non-dérivables associés à une relation de données R. L'ensemble des fermetures des $\mathcal{GMND}$s est un treillis complet et gradué, appelé cube de données non-dérivable fermé, structuré par la relation de généralisation/spécialisation entre tuples, notée par $\preceq$:*
$\mathcal{CND}$-$\mathcal{C}ube(R)$ = $(\gamma(\mathcal{GMND}(R)), \preceq) = (\mathcal{MFND}, \preceq)$.

Exemple 1 *Considérons la relation* VOITURE (*cf. Tableau 1*) *répertoriant les quantités vendues par modèle, par ville, par client et par couleur. Le Tableau 2 illustre le cube non-dérivable fermé, obtenu à partir de cette relation. En effet, nous avons* (*Peugeot, Beja, Auto, Gris*) + (*Peugeot, Beja, LeMoteur, Gris*) = (*Peugeot, Beja, ALL, Gris*). *Ceci implique que le tuple* (*Peugeot, Beja, ALL, Gris*) *est construit à partir des tuples* (*Peugeot, Beja, Auto, Gris*) *et* (*Peugeot, Beja, LeMoteur, Gris*). *En outre, nous avons* (*Ford, Sousse, ALL, ALL*) $*$ (*ALL, ALL, Auto, Gris*) = (*Ford, Sousse, Auto, Gris*). *Ainsi,* (*Ford, Sousse, ALL, ALL*) *et* (*ALL, ALL, Auto, Gris*) *généralisent* (*Ford, Sousse, Auto, Gris*) *et ce dernier tuple participe à la construction de* (*Ford, Sousse, ALL, ALL*) *et* (*ALL, ALL, Auto, Gris*) (*directement ou non*). *Les tuples* (*Ford,*

ALL, ALL, ALL) *et* (*Peugeot, ALL, ALL, ALL*) *n'ont d'autre point commun que le tuple de valeurs vides* (*i.e. le tuple* (∅,∅,∅)).

Tuples Convexes	Qté	Tuples Convexes	Qté
(ALL, ALL, ALL, Noir)	400	(Peugeot, ALL, Auto, ALL)	500
(ALL, ALL, ALL, Gris)	700	(Peugeot, ALL, ALL, Gris)	300
(ALL, ALL, Auto, ALL)	700	(Peugeot, Sousse, ALL, Noir)	300
(ALL, ALL, LeMoteur, ALL)	200	(Peugeot, ALL, Auto, Noir)	300
(ALL, Beja, ALL, ALL)	300	(Peugeot, Beja, Auto, ALL)	200
(ALL, ALL, Auto, Noir)	300	(Peugeot, Beja, ALL, Gris)	200
(ALL, ALL, Auto, Gris)	600	(Peugeot, Sousse, Auto, ALL)	300
(ALL, Sousse, ALL, Noir)	400	(Peugeot, Auto, ALL, Gris)	200
(ALL, Beja, ALL, Gris)	300	(Peugeot, Sousse, Auto, Noir)	300
(ALL, Sousse, Auto, ALL)	700	(Peugeot, Beja, Auto, Gris)	200
(ALL, Sousse, ALL, Gris)	400	(Ford,ALL, ALL, ALL)	500
(ALL, Beja, Auto, ALL)	200	(Ford, ALL, ALL, Gris)	400
(ALL, Beja, Auto, Gris)	200	(Ford, ALL, Auto, ALL)	400
(ALL, Sousse, Auto, Noir)	300	(Ford, ALL, Auto, Gris)	400
(ALL, Sousse, Auto, Gris)	400	(Ford, Sousse, ALL, ALL)	500
(Peugeot, ALL, ALL, ALL)	600	(Ford, Sousse, ALL, Gris)	400
(Peugeot, ALL, ALL, Noir)	300	(Ford, Sousse, Auto, ALL)	400
(Peugeot, Beja, ALL, ALL)	300	(Ford, Sousse, Auto, Gris)	400
(Peugeot, Sousse, ALL, ALL)	300		

(Cube Convexe)

Tuples Convexes Non-Dérivables Fermés	COUNT (*Qté*)
(ALL, ALL, ALL, Gris)	700
(ALL, ALL, LeMoteur, ALL)	200
(ALL, ALL, Auto, Gris)	600
(ALL, Sousse, ALL, Noir)	400
(ALL, Sousse, Auto, ALL)	700
(Peugeot, ALL, ALL, ALL)	600
(Peugeot, ALL, Auto, ALL)	500
(Peugeot, Beja, ALL, Gris)	300
(Peugeot, Beja, Auto, Gris)	200
(Peugeot, Sousse, Auto, Noir)	300
(Ford, Sousse, ALL, ALL)	500
(Ford, Sousse, Auto, Gris)	400

(Cube Convexe Non-Dérivable Fermé)

FIG. 1 – *Cube convexe vs. cube convexe non-dérivable fermé issus de la relation* VOITURE *sous les contraintes 200* ≤ COUNT*(Quantité)* ≤ *700.*

Définition 10 (Cube de Données Convexe Non-Dérivable Fermé) . *Tout treillis cube non-dérivable fermé avec contraintes monotones et/ou antimonotones est un espace convexe qu'on appelle cube convexe non-dérivable fermé, noté par :*
$\mathcal{CCND}$-$\mathcal{C}ube(R)$ = { $t \in \mathcal{CND}$-$\mathcal{C}ube(R)$ | $cont(t)$}*, où cont peut être une contrainte monotone, antimonotone ou hybride. N'importe quel tuple appartenant au* $\mathcal{CCND}$-$\mathcal{C}ube(R)$ *s'appelle un tuple convexe non-dérivable fermé.*

Exemple 2 *Le* $\mathcal{CCND}$-$\mathcal{C}ube$(VOITURE) *est représenté par le tableau à droite dans la Figure 1 donnant l'ensemble des tuples convexes non-dérivables fermés. Ces tuples respectent la conjonction de contraintes monotones et antimonotones 200* ≤ COUNT*(Quantité)* ≤ *700. Dans l'espace du treillis cube non-dérivable fermé de la relation* VOITURE (*cf. Tableau 2*)*, nous voulons connaître tous les tuples dont la valeur de la mesure* COUNT(*Quantité*) *est supérieure ou égale à 200. La contrainte "*COUNT(*Quantité*) ≥ *200" est une contrainte antimonotone. Si le total des ventes par modèle, par ville et par client est supérieur à 200, il l'est a fortiori pour un niveau plus agrégé de granularité, e.g. par modèle et par client* (*toutes villes confondues*) *ou par ville* (*tous modèles et clients confondus*)*. De même, si nous voulons connaître tous les tuples dont les valeurs pour l'attribut "Quantité" sont inférieures ou égales à 700, la contrainte exprimée "*COUNT(*Quantité*) ≤ *700" est monotone. Considérons que le total des ventes par modèle* (*les attributs ville, client et couleur ont pour valeur ALL*) *respecte cette contrainte, la même information observée à un niveau de détail plus fin satisfait forcément la même condition et donc les ventes par modèle et par ville sont inférieures à 700 de même que les ventes par modèle, par ville et par client.*

Définition 11 (Fonction agrégative relative) . *Soit R une relation, $t \in \mathcal{CCND}\text{-}\mathcal{C}ube(R)$ un tuple et $f \in \{$*COUNT, SUM*$\}$ une fonction agrégative. On appelle $f_{val}(., R)$ la fonction agrégative relative de la fonction f pour la relation R. $f_{val}(t, R)$ est le ratio entre la valeur de f pour le tuple t et la valeur de f appliquée sur toute la relation R (donc pour le tuple (ALL,...,ALL)).*

$$f_{val}(t, R) = \frac{f(t,R)}{f_{val}((ALL,..,ALL),R)}$$

Exemple 3 *La fonction* $\text{COUNT}_{val}(t, R)$ *correspond simplement à la fréquence d'apparition du motif multidimensionnel t dans la relation R. Si on considère les ventes des voitures de couleur "Noir" à "Sousse", pour tous les modèles et tous les clients, i.e. le tuple (ALL, Sousse, ALL, Noir) de $\mathcal{CCND}\text{-}\mathcal{C}ube$(*VOITURE*) on a : la fréquence d'apparition du motif multidimensionnel (ALL, Sousse, ALL, Noir) dans la relation* VOITURE *est* COUNT_{val}*((ALL, Sousse, ALL, Noir),* VOITURE*) = 0.4.*

Définition 12 (Tuple Convexe Non-Dérivable Fermé) . *Un tuple $t \in \mathcal{CCND}\text{-}\mathcal{C}ube(R)$ est dit convexe non-dérivable fermé si et seulement s'il satisfait les deux contraintes suivantes :*

(C_1) $f_{val}(t, R) \geq MinSeuil_1$ *;*

(C_2) $f_{val}(t, R) \leq MinSeuil_2$.

où $MinSeuil_1$ et $MinSeuil_2 \in$]0,1[.

Exemple 4 *Soient les seuils $MinSeuil_1$ = 0.4 pour la relation* VOITURE *(cf. Tableau 1) et $MinSeuil_2$ = 0.7. Parmi les tuples non-dérivables fermés illustrés dans le Tableau 2, le tuple t_1 = (Ford, Sousse, ALL, ALL) est convexe car* $\text{COUNT}_{val}(t_1, R) = \frac{500}{1100}$ *= 0.5. Par contre, le tuple t_2 = (Peugeot, Sousse, Auto, Noir) ne l'est pas car* $\text{COUNT}_{val}(t_2, R) = \frac{300}{1100}$ *= 0.3.*

Remarque 1 *Dans le cas où on prend en compte la contrainte antimonotone (contrainte d'iceberg) et la fonction d'agrégation* COUNT *seulement, le cube convexe non-dérivable fermé (comme le cube convexe fermé) capture la base Max-Max (Zaki, 2000) des règles d'association dans un contexte* OLAP. *Le graphe de couverture de cette structure offre à l'utilisateur un outil de visualisation pour la base de règles d'association multidimensionnelles. Nous rappelons que le cube convexe quotient capture la base Min-Max (Bastide et al., 2000) des règles d'association dans un contexte* OLAP. *De plus, le cube convexe quotient permet de fournir aux utilisateurs des outils pour naviguer dans les cubes convexes.*

Finalement, il est possible de retrouver les valeurs des mesures associées aux différents tuples à partir du $\mathcal{CCND}\text{-}\mathcal{C}ube$. De plus, cette représentation englobe l'ensemble des tuples convexes non-dérivables fermés. Ainsi, le résultat de toute requête OLAP peut en être dérivé. Par exemple, nous pouvons répondre à des questions du type : "Pour quels modèles de voiture les ventes évoluent entre 600 et 1000 ?".

5 Calcul du $\mathcal{CCND}\text{-}\mathcal{C}ube$: l'algorithme dfCLOSND

Dans ce qui suit, nous introduisons l'algorithme dfCLOSND permettant le calcul du cube convexe non-dérivable fermé. Son pseudo-code est illustré par l'algorithme 1.

Algorithme 1 : dfCLOSND

Données :

1. $\mathcal{D}$: Une base de données,
2. $MinSeuil_1$, $MinSeuil_2$: Les seuils du support.

Résultat : $\mathcal{MFND}$: Ensemble des motifs fermés non-dérivables.

début
  $\mathcal{GMND} := \emptyset$; $\mathcal{MFND} := \emptyset$;
  /* $\mathcal{D}$ est dans un ordre croissant. */
  pour chaque $c \in \mathcal{D}$ **faire**
    /* $c = i$ ou $c = \bar{i}$ pour un item i */
    $\mathcal{GMND} := \mathcal{GMND} \cup \{(X \cup Y \cup \{i\})\}$;
    /* Créer $\mathcal{D}_c$ */
    $\mathcal{D}_c := \emptyset$;
    pour chaque $k \in \mathcal{D}$ *situé après* c **faire**
      /* k=j ou k=$\bar{j}$ pour un item j */
      /* Soit $J = X \cup Y \cup \{i, j\}$;*/
      Calculer la borne inférieure l et la borne supérieure u du support de J ;
      si $J.l \neq J.u$ *et* $MinSeuil_1 \leq J.u \leq MinSeuil_2$ **alors**
        /* J est non-dérivable.*/
        C[k] := couverture([c]) $\cap$ couverture([k]) ;
        C[$\bar{k}$] := couverture([c]) $\setminus$ couverture([k]) ;
        /* Calculer le support *Supp*(J) *i.e*, COUNT$_{val}$(J,$\mathcal{D}$)*/
        /*Calculer le support estimé*/
        Supp-estimé(J) := *min*{*Supp*(K) | $K \subset J$ et $|K|$ = $|J|$-1} ;
        si *Supp-estimé*(J) $\neq$ *Supp*(J) **alors**
          /* J est un générateur minimal non-dérivable. */
          Stocker J dans un arbre séparé ;
          si $MinSeuil_1 \leq$ *Supp*(J) $\leq MinSeuil_2$ *et* *Supp*(J) $\neq J.l$ *et* *Supp*(J) $\neq J.u$ **alors**
            si |C[j]|$\leq$|C[$\bar{j}$]| **alors**
              $\mathcal{D}_c$:= $\mathcal{D}_c \cup$ {(j,C[j])} ;
            else
              $\mathcal{D}_c := \mathcal{D}_c \cup \{(\bar{j}$,C[$\bar{j}$])$\}$;
    $\mathcal{GMND} := \mathcal{GMND} \cup$ dfCLOSNDMG($\mathcal{D}_c$, *MinSeuil*$_1$, *MinSeuil*$_2$) ;
  $\mathcal{MFND} := \{\gamma(I) \mid I \in \mathcal{GMND}\}$;
  retourner $\mathcal{MFND}$
fin

D'une part, nous profitons de la conclusion tirée par Casali et al. (2003). Ainsi, nous utilisons l'isomorphisme (Casali et al., 2003) et le théorème de Birkhoff (Ganter et Wille, 1999) afin d'appliquer notre algorithme. En effet, les auteurs ont prouvé qu'il y a un isomorphisme de treillis entre le treillis cube et le treillis de Galois (treillis de concept) calculés à partir d'un contexte d'extraction R. Cet isomorphisme est important puisqu'il permet l'utilisation des algorithmes de la fouille de données dans le calcul des représentations concises d'un cube de

données. Par exemple, le calcul du *cube convexe fermé* et du *cube convexe quotient* est basé sur l'algorithme CLOSE (Pasquier et al., 1999). Plus précisément, à partir d'un contexte d'extraction R, nous extrayons les motifs fermés non-dérivables en calculant les fermetures des générateurs minimaux non-dérivables. Ensuite, en se basant sur l'isomorphisme, nous employons le théorème de Birkhoff pour passer d'un treillis de concepts à un treillis $\mathcal{CCND}$-$\mathcal{C}$ube.

D'autre part, dfCLOSND respecte la technique "Diviser-pour-régner" (recherche en profondeur). En effet, l'idée motivant ce choix est d'éviter le goulot d'étranglement de l'approche "tester-et-générer" (recherche en largeur) proposée dans (Brahmi et al., 2012), à savoir la génération d'un nombre prohibitif de candidats. Dans la littérature, on retrouve essentiellement un seul algorithme implémentant cette technique, à savoir l'algorithme dfNDI (Calders et Goethals, 2005). Ce dernier permet d'extraire les motifs non-dérivables.

L'algorithme dfCLOSND effectue le processus d'extraction des $\mathcal{MFND}$ en deux étapes successives :
- La **première étape** extrait les motifs générateurs minimaux non-dérivables respectant une conjonction de contraintes monotones et/ou antimonotones. Cette étape est basée sur les deux principes suivants :

(a) Outre l'élagage des candidats ne respectant pas la conjonction de contraintes sur le support (*i.e.* $MinSeuil_1$, $MinSeuil_2$), dfCLOSND adopte une stratégie d'élagage basée sur le support estimé. Ce dernier est calculé du moment qu'on est sûr qu'un candidat *J* est non-dérivable (*cf.* ligne 13-14). Ainsi, si le support estimé du candidat *J* est égal à son support réel, alors *J* n'est pas un générateur minimal. Il sera élagué en conséquence (*cf.* ligne 18-22). Par ailleurs, tout générateur minimal non-dérivable fréquent *J* admettant un support égal soit à la borne supérieure ou à la borne inférieure ne sera pas utilisé pour générer les candidats de plus grande taille (*cf.* ligne 23-27). En effet, dans ce cas, tout sur-ensemble de *J* est prouvé être un motif dérivable (Calders et Goethals, 2007). Nous notons que l'ensemble des $\mathcal{GMND}$s est stockés dans une structure de données avancée. L'idée motivant cette structure de donnée compacte, basée sur la notion de trie [1] (Kruse et Ryba, 1999), est que lorsque plusieurs transactions se partagent un item, alors elles peuvent être fusionnées en prenant soin d'enregistrer le nombre d'occurrences des items (consultation rapide des supports) (*cf.* ligne 22).

(b) Au lieu d'une exploration en largeur des motifs générateurs minimaux non-dérivables candidats, dfCLOSND effectue une partition de l'espace de recherche pour effectuer ensuite une exploration en profondeur d'abord. Ainsi, il commence par considérer les 1-motifs fréquents, et examine seulement leur sous-contextes conditionnels $\mathcal{D}_c$. Un sous-contexte conditionnel ne contient que les items qui co-occurrent avec le 1-motif en question. Récursivement, les bases de données conditionnelles $\mathcal{D}_c$ sont construites (*cf.* ligne 4-8). Le choix du type de couverture [2] à employer n'est pas statique. Comme dans dfNDI, ce choix dépend du temps d'exécution : les deux couvertures sont calculées (*cf.* lignes 15-16) et celle avec la taille minimale sera choisie (*cf.* lignes 24-27). De cette façon, on garantit que : (i) la taille du couverture diminue de la moitié allant de $\mathcal{D}$ à $\mathcal{D}_c$ et ; (ii) la base de données peut être compressée pour arriver à achever le processus de fouille de données.

1. de reTRIEval.
2. La couverture d'un motif X dans $\mathcal{D}$ comprend l'ensemble des identifients des tuples dans $\mathcal{D}$ qui supportent X.

- Dans **la deuxième étape**, nous soulignons que le calcul des motifs fermés non-dérivables peut être optimisé si nous utilisons les générateurs minimaux. Par conséquent, au lieu de calculer l'ensemble des motifs non-dérivables puis leurs fermetures associées tel que le cas dans (Muhonen et Toivonen, 2006), nous utilisons seulement l'ensemble des générateurs minimaux non-dérivables. Par conséquent, l'introduction des générateurs minimaux dans l'algorithme optimisera l'étape de génération des candidats et les étapes de calcul de la fermeture. Motivé par cette idée, l'algorithme dfCLOSND parcourt l'ensemble des générateurs minimaux non-dérivables (*cf.* ligne 30). L'ensemble des motifs fermés non-dérivables correspond à l'ensemble des fermetures des générateurs minimaux non-dérivables. Pour chaque générateur minimal non-dérivable, sa fermeture et son support sont insérés dans l'ensemble des motifs fermés non-dérivables.

6 Évaluation expérimentale

Toutes les expérimentations ont été réalisées sur un PC équipé d'un Pentium 4 avec une fréquence d'horloge de 3 GHz et une mémoire RAM de 2 Go, utilisant la distribution de Linux Fedora Core 6 comme système d'exploitation. Nous avons choisi de comparer notre approche aux deux autres représentations concises s'inscrivant dans la même tendance, *i.e.*, cube convexe quotient, cube convexe fermé et au cube convexe. Pour calculer le cube convexe quotient et le cube convexe fermé, nous utilisons l'algorithme CLOSE considéré efficace dans la recherche de motifs fermés fréquents et pour lequel nous disposons des sources.

Durant les expérimentations effectuées, nous avons utilisé deux bases synthétiques [3] : (i) MUSHROOM est un banc d'essai dense ; et (ii) T10I4D100K est un banc d'essai éparse. De plus, nous avons utilisé une base réelle fréquemment exploitée dans le contexte des cubes de données (Casali et al., 2009a; Cicchetti et al., 2012) : SEP85L [4] décrit les conditions atmosphériques à de diverses stations météorologiques à partir de décembre 1981 à novembre 1991. Les caractéristiques de ces bases sont résumées par le Tableau 3.

Bases	# Attributs	# Tuples
SEP85L	7 871	1 015 367
MUSHROOM	119	8 124
T10I4D100K	1 000	100 000

TAB. 3 – *Descriptif des jeux de données.*

Dans les expérimentations, le paramètre qui change est le seuil minimal pour la contrainte antimonotone (les valeurs des mesures sont au-dessus de $MinSeuil_1$). Le seuil minimal pour la contrainte monotone (les valeurs des mesures sont au-dessous de $MinSeuil_2$) reste constant.

Nous avons effectué des évaluations expérimentales avec une double intention. D'une part, nous comparons le temps d'exécution consommé par dfCLOSND contre celui de FIRM [5] et celui de CLOSENDMG (Brahmi et al., 2012) pour calculer le $\mathcal{CCND}$-$\mathcal{C}$ube. D'autre part, nous nous concentrons sur l'évaluation de la compacité, en termes de l'espace de stockage sur disque, de notre approche contre celles proposées dans la littérature et s'inscrivant dans la même tendance.

3. Disponibles à l'adresse suivante : *http ://fimi.cs.helsinki.fi/data/.*
4. Disponible à l'adresse suivante : *http ://cdiac.esd.ornl.gov/cdiac/ndps/ndp026b.html.*
5. Disponible à l'adresse suivante : *http ://www.cs.helsinki.fi/u/jomuhone/.*

6.1 Les performances de dfCLOSND

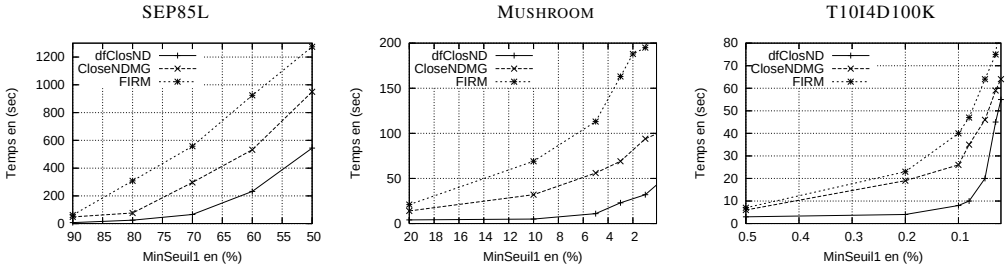

FIG. 2 – *Le temps d'extraction des $\mathcal{CCND}$-Cubes en utilisant les algorithmes* FIRM*,* CLOSENDMG *et df*CLOSND*.*

La figure 2 illustre le temps d'exécution consommé afin de générer le $\mathcal{CCND}$-$\mathcal{C}$ube pour les bases de données considérées, en utilisant les algorithmes dfCLOSND, CLOSENDMG et FIRM.

L'algorithme dfCLOSND s'avère performant sur les bases dense, éparse et réelle pour toutes les valeurs de $MinSeuil_1$. La différence entre les performances de dfCLOSND et celles des deux autres algorithmes atteint son maximum pour la base MUSHROOM. Pour un seuil minimal égal à 1%, dfCLOSND est 6 fois plus rapide que FIRM, alors qu'il est 3 fois plus rapide que CLOSENDMG. Pour la base SEP85L, les performances de dfCLOSND sont largement meilleures que celles des autres algorithmes. Par exemple, pour une valeur de $MinSeuil_1$ égale à 50%, les différences sont 730 et 406 secondes, respectivement, par rapport à FIRM et CLOSENDMG. Pour la base éparse T10I4D100K, l'algorithme dfCLOSND consomme des temps d'exécution beaucoup moins importants par rapport aux autres algorithmes. Pour cette base, dfCLOSND est en moyenne 30 fois plus rapide que FIRM et 16 fois plus rapide que CLOSENDMG.

Ainsi, nous constatons que dfCLOSND est plus performant que FIRM. Le mécanisme de comptage des générateurs minimaux adopté par dfCLOSND s'avère plus efficace que FIRM. Ceci peut être expliqué par le nombre réduit des générateurs minimaux fréquents non-dérivables à prendre en considération lors du calcul de la fermeture. Cependant, l'algorithme FIRM est handicapé par un calcul redondant des fermetures. De plus, nous remarquons que dfCLOSND est plus performant que CLOSENDMG. La stratégie de recherche en profondeur (diviser-pour-régner) adoptée par dfCLOSND permet d'éviter le goulot d'étranglement de l'algorithme CLOSENDMG. En effet, dfCLOSND essaie de diviser le contexte d'extraction en des sous-contextes et d'appliquer le processus de découverte des $\mathcal{MFND}$ récursivement sur ces sous-contextes. Ce processus de découverte repose sur un élagage du contexte basé sur une conjonction de contraintes. Cependant, CLOSENDMG est entravé par la génération d'un nombre prohibitif de candidats.

6.2 La taille du $\mathcal{CCND}$-$\mathcal{C}$ube

Dans ce qui suit, nous comparons la taille du $\mathcal{CCND}$-$\mathcal{C}$ube à stocker, respectivement contre la taille d'un *cube convexe*, d'un *cube convexe fermé* et d'un *cube convexe quotient*. La figure 3 illustre la cardinalité des tuples obtenus pour chaque représentation concise.

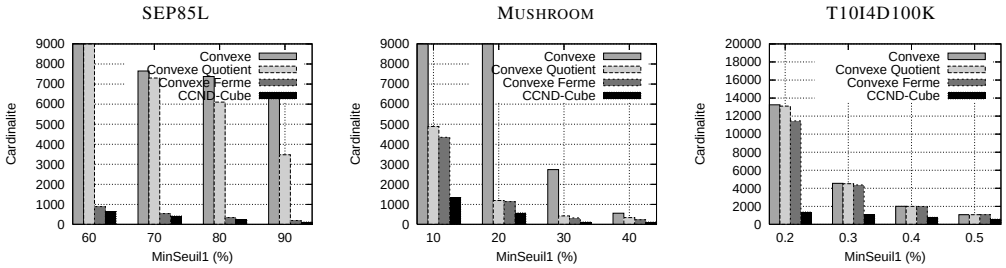

FIG. 3 – *La taille des cubes de données générés.*

Pour les trois bases, nous observons que plus le $MinSeuil_1$ s'accroît plus la taille des représentations concises décroît. Effectivement, quand le seuil minimal est élevé, il est logique que le nombre de tuples convexes soit moindre. Cependant, le $\mathcal{CCND}$-$\mathcal{C}$ube est toujours plus réduit que les autres cubes avec un gain appréciable.

Les taux de compression obtenus pour les bases de données denses et réelles, MUSHROOM et SEP85L, par le $\mathcal{CCND}$-$\mathcal{C}$ube sont significatifs. En outre, les taux de compression sont beaucoup plus élevés que ceux obtenus respectivement par le *cube convexe fermé* et le *cube convexe quotient*. Pour la base MUSHROOM, la taille du $\mathcal{CCND}$-$\mathcal{C}$ube est en moyenne 3 fois plus petite que le *cube convexe quotient* et 2.5 fois plus petite que le *cube convexe fermé*. De plus, pour la base SEP85L, la taille de notre représentation est en moyenne 29 fois plus petite que le *cube convexe* alors que les tailles du *cube convexe quotient* et du *cube convexe fermé* sont, respectivement, 1.35 et 19.8 fois plus petites. Nous concluons que pour les bases de données denses et réelles, la compression est plus élevée en utilisant le $\mathcal{CCND}$-$\mathcal{C}$ube.

Pour la base T10I4D100K, les tailles des *cubes convexes* et *convexes quotients* sont comparables ou légèrement inférieures à la taille du *cube convexe fermé*. La raison de cette très faible réduction de l'espace est la suivante : les données éparses sont très faiblement corrélées (Pasquier et al., 1999) et donc ne contiennent que très peu de redondances. Cependant, on remarque que les taux de compression obtenus pour le $\mathcal{CCND}$-$\mathcal{C}$ube dépassent ceux obtenus par les autres représentations. En moyenne, les cubes *convexes fermés* et *quotients* sont 1 fois plus petits que le *cube convexe* alors que le $\mathcal{CCND}$-$\mathcal{C}$ube est 4.5 fois plus petit. D'où, notre représentation permet une réduction effective.

Considérons les quatre représentations concises (cube convexe, cube convexe fermé, cube convexe quotient et $\mathcal{CCND}$-$\mathcal{C}$ube), nous concluons que les meilleurs taux de réduction d'un cube convexe sont obtenus pour les bases de données denses et réelles, *i.e*, MUSHROOM et SEP85L. En outre, les taux de compression obtenus avec la base de données éparse, *i.e.*, T10I4D100K sont plus petits et modestes mais ils sont toujours inférieurs à ceux obtenus par le *cube convexe fermé* et le *cube convexe quotient*.

7 Conclusion et perspectives

Dans ce papier, nous nous sommes concentrés sur les approches de réduction des cubes de données convexes, utilisant les algorithmes de la fouille de données afin d'attaquer les défis suivants : un temps d'exécution coûteux aussi bien qu'un grand espace de stockage sur le disque. Ainsi, nous avons introduit un cube convexe non-dérivable fermé appelé le $\mathcal{CCND}$-$\mathcal{C}$ube basé

sur un algorithme d'extraction efficace appelé dfCLOSND. Les expériences, que nous avons réalisées, ont montré que le $\mathcal{CCND}$-$\mathcal{C}$ube est plus performant que les représentations de réduction des cubes de données convexes existantes dans la littérature pour les différents types de contextes testés.

Les perspectives de travaux futurs concernent : (1) Les cubes convexes peuvent être représentés par les bordures classiques (Cicchetti et al., 2012). Ces dernières ont des avantages particuliers : éviter de calculer les cubes sous-jacents à comparer, offrir un outil de classification et focaliser l'attention de l'utilisateur sur un ensemble particulier de tuples intéressants ; et (2) les hiérarchies présentent plusieurs complications dans la construction d'un cube de données. Particulièrement les hiérarchies constituent un défi principal. En effet, le nombre de tuples qui doit être matérialisé dans le cube devient très élevé ce qui augmente la consommation de l'espace de stockage sur le disque. Nous proposons de prendre en compte les hiérarchies des dimensions dans le cube convexe non-dérivable fermé de la même manière que pour le cube CURE (Morfonios et Ioannidis, 2006).

Remerciements

Ce travail a été partiellement financé par le projet CMCU 11G1417.

Références

Bastide, Y., N. Pasquier, R. Taouil, G. Stumme, et L. Lakhal (2000). Mining Minimal Non-Redundant Association Rules Using Frequent Closed Itemsets. In *Proceedings of the First International Conference on Computational Logic, London, UK*, pp. 972–986.

Beyer, K. et R. Ramakrishnan (1999). Bottom-Up Computation of Sparse and Iceberg CUBEs. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data, Pennsylvania, USA*, pp. 359–370.

Brahmi, H., T. Hamrouni, R. B. Messaoud, et S. B. Yahia (2012). A New Concise and Exact Representation of Data Cubes. In *Advances in Knowledge Discovery and Management*, Volume 2, pp. 27–48. Springer-Verlag.

Calders, T. et B. Goethals (2005). Depth-First Non-Derivable Itemset Mining. In *In Proceedings of SIAM International Data Mining Conference,*, pp. 250–261.

Calders, T. et B. Goethals (2007). Non-Derivable Itemset Mining. *Data Mining and Knowledge Discovery, 14*(1), 171–206.

Casali, A. (2004). Mining Borders of the Difference of Two Datacubes. In *Proceedings of the 6th International Conference on Data Warehousing and Knowledge Discovery, Zaragoza, Spain*, pp. 391–400.

Casali, A., R. Cicchetti, et L. Lakhal (2003). Extracting Semantics from Data Cubes using Cube Transversals and Closures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, Washington, USA*, pp. 69–78.

Casali, A., R. Cicchetti, et L. Lakhal (2009a). Closed Cubes Lattices. *Annals of Information Systems 3*, 145–165. Special Issue on New Trends in Data Warehousing and Data Analysis.

Casali, A., S. Nedjar, R. Cicchetti, L. Lakhal, et N. Novelli (2009b). Lossless Reduction of Datacubes Using Partitions. *International Journal of Data Warehousing and Mining (IJDWM) 4*(1), 18–35.

Chaudhuri, S. et U. Dayal (1997). An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record 26*(1), 65–74.

Cicchetti, R., L. Lakhal, et S. Nedjar (2012). Constrained Closed and Quotient Cubes. In *Advances in Knowledge Discovery and Management*, Volume 2, pp. 3–26. Springer-Verlag.

Ganter, B. et R. Wille (1999). F*ormal* C*oncept* A*nalysis*. Springer-Verlag.

Han, J., J. Pei, G. Dong, et K. Wang (2001). Efficient Computation of Iceberg Cubes with Complex Measures. In *Proceedings of the International Conference on Management of Data, SIGMOD, Santa Barbara, California*, pp. 441–448.

Kruse, R. L. et A. J. Ryba (1999). D*ata* S*tructures and* P*rogram* D*esign in* C^{++}.

Morfonios, K. et Y. E. Ioannidis (2006). Cure for Cubes : Cubing Using a ROLAP Engine. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea*, pp. 379–390.

Muhonen, J. et H. Toivonen (2006). Closed Non-Derivable Itemsets. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany*, pp. 601–608.

Nedjar, S., R. Cicchetti, et L. Lakhal (2011). Extracting Semantics in OLAP Databases Using Emerging Cubes. *Information Science 181*, 2036–2059.

Pasquier, N., Y. Bastide, R. Taouil, et L. Lakhal (1999). Efficient Mining of Association Rules Using Closed Itemset Lattices. *Journal of Information Systems 24*(1), 25–46.

Ross, K. et D. Srivastava (1997). Fast Computation of Sparse Data Cubes. In *Proceedings of the 23rd International Conference on Very Large Databases, Athens, Greece*, pp. 116–125.

Vel, M. (1993). T*heory of* C*onvex* S*tructures*. North-Holland, Amsterdam.

Zaki, M. J. (2000). Generating Non-Redundant Association Rules. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, Boston, USA*, pp. 34–43.

Summary

In various approaches, data cubes are pre-computed in order to answer efficiently OLAP queries. The notion of data cube has been declined in various ways: iceberg cubes, differential cubes or emergent cubes. In this paper, we investigate reduced representations for the convex cubes. That is why we introduce a new and reduced structure of data cube: the Constrained Closed Non-Derivable Data Cube ($\mathcal{CCND}$-$\mathcal{C}$ube). The latter captures all the tuples of a datacube fulfilling a combination of monotone/ antimonotone constraints. It can be represented in a very compact way in order to optimize both computation time and required storage space. The results of our experiments confirm the relevance of our proposal.