UNIVERSITE SAAD DAHLAB DE BLIDA
Faculté des Sciences
Département d'Informatique

N° D'ordre :.....

Mémoire Présenté par :

DJOUALIL Fatiha

HABARKA Ibtissam

En vue d'obtenir le diplôme de master
Domaine : Mathématique et informatique
Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Thème : Une nouvelle approche pour le problème d'extraction des règles d'association.

Soutenu le : juin 2016, devant le jury composé de :

Mme. FAREH M	Présidente
Mme. OUAHRANI L	Examinatrice
Me. DJENOURI Youcef	Promoteur

Promotion
2015/ 2016

Remerciement

A fin de ce travail nous remercions premièrement notre bon DIEU qui nous réussites en notre carrière d'étude, nos parents qui nous réservent tous les besoins de vie.

C'est avec un grand plaisir, que nous exprimons ici notre reconnaissance à tous ceux qui nous ont apporté leur aide scientifique, matérielle ou morale pour mener à terme ce travail :

Mme: chef de département informatique, Chaque enseignant à son nom.

Notre promoteur Mr: Djenouri Youcef, Il nous guidés de début du projet à sa fin.

Mr: Touitti Mohammed, qui nous a beaucoup aidés dans notre projet.

Mr: Dahmani Abdelaziz, qui nous donne le coup de main.

C'est bien sûr n'oublions pas de remercie monsieur Makfouldji.

Et tous les employeurs de la universitaire Saad Dahlab de Blida.

Enfin encore merci à toutes et pour toutes…

Dédicace

Je voudrais remercier en premier le bon Dieu tout puissant, et dédié ce modeste travail a

Mes très chères ma mère aicha et mon père

Abdelkader pour leur amour et leur encouragement et aussi mes sœurs Fatima, Hacena, Bothina, mon frère mohamed blkhair

Et a mes adorables anges Aicha, Naila, zineb.

A mes très chére tantes et oncles.

A ma chére binôme fatiha et tous mes amis zineb, selma, fadila, sawsan, rekia, amina.

Et toutes personnes que font partie de ma vie.

IBTISSAM

Résumé:

Grâce au développement technologique, les bases de données deviennent très coûteuses. Cela conduit à confronter des difficultés d'extraction des connaissances. Tandis que la fouille de données se propose plusieurs techniques, comprenant parmi elles classification, estimation, prédiction, Clustering et association, nous nous s'intéressons dans ce projet à l'extraction des règles d'association qui se base sur l'algorithme d'Apriori. Ce dernier a rencontré un problème de lentement à cause du parcours récursif de la base de transactions. Pour cela, nous proposons deux approches : l'une permet d'éliminer la transaction qui ne contient pas un itemset fréquent et l'autre permet de réaliser un seul parcours de la base de transaction afin de construire une table de hachage contenant tous les itemsets possibles et leurs supports. Après notre étude théorique et expérimentale, nous déduisons que l'approche1 est rapide par rapport au Apriori quel que soit le minSup et le minConf mais l'approche2 fonctionne selon le type de base, par rapport à une base non condensée, mieux qu'une base condensé.

Abstract:

Thanks to technological development, the database became contain a large amount of data of those who make the difficult of extraction of knowledge, data mining display many techniques such as classification, prediction, clustering, estimation, association. In this project we care about extraction association rules, Apriori is a main algorithm for extracting association rules but is consume many times for this we proposes two approaches, the first permit to delete the transaction that not contain a frequent itemset and other based on only one browse of transaction base. After our theoretical and experimental study we deduce that approche1 is fast compared with Apriori whatever minSup and minConf but the approach2 walk according to the base type, compared to a non-condensed base better than base condensed.

ملخص:

مع التطور التكنولوجي أصبح من السهل نقل المعلومات مما ادى إلى تضخم قاعدة المعطيات و بالتالي مواجهة صعوبات اكثر في استخراج المعلومات. لذلك اقترحت تكنولوجيا التنقيب عن المعلومات عدة تقنيات منها: التصنيف, التقييم, التنبؤ, التقسيم, الربط. نهتم في هذا المشروع باستخراج قواعد الربط التي تقوم على خوارزمية ابغيوغي. هذا الاخير يأخذ وقت كبير في تصفح قاعدة التبادلات لاستخراج قواعد الربط لذلك قمنا باقتراح خوارزميتين جديدتين لهما نفس مبدأ ابغيوغي الاولى تقوم بالتنقيب عن الحدود المتواترة بطريقة رجعية مع ازالة الحدود الغير متواترة. اما الاخرى فتعتمد على مبدأ تصفح قاعدة التبادلات مرة واحدة لإنشاء جدول الحدود المحتملة مع تكراراتها.

نتيجة لدراستنا النظرية والتطبيقية استنتجنا ان الخوارزمية الاولى تبقى وبشكل مستمر افضل من ,اما الثانية فحسب قاعدة التبادلات بحيث تستغرق وقت اقل اذا كانت القاعدة غير مكثفة.

Table des matières

Table des figures

Liste des tables

Introduction générale

Avec le développement technologique, la collection des données est devenue très facile mais notre puissance à extraire l'information à partir d'une grande base de données reste limitée. Afin d'exploiter ces masses importantes des données stockées dans les systèmes décisionnels, la fouille de données se propose des techniques pour extraire cette connaissance. La fouille de données ou bien data mining est le domaine de recherche au sein duquel coopèrent statisticiens, spécialistes en bases de données et en intelligence artificielle, ou encore chercheurs en conception d'interfaces homme-machine [1]. Les informations extraites, suite à l'application d'un processus de fouille de données, peuvent prendre plusieurs formes, telles que les règles d'association, arbre de décision, réseaux de neurones. Dans ce travail, nous allons nous intéresser spécialement à la technique d'extraction des règles d'association. Cette technique a été introduite par Agrawal et al. en 1993 [2]. En 1994, Agrawal et al. ont proposé un des premiers algorithmes, appelé Apriori [3], pour l'extraction des règles d'association à partir des grandes bases de données. L'inconvénient majeur de cet algorithme est le parcours de la base de transaction et cela pour chaque k-itemset généré, ce qui nécessite un temps de fouille considérable. Le problème d'extraction des règles d'association propose plusieurs variantes séquentielles et parallèles. Dans notre mémoire nous allons nous intéresser aux algorithmes séquentiels exacts. Pour cela, et dans le but de diminuer le temps d'exécution d'APRIORI, on propose deux algorithmes intelligents basés sur APRIORI qui parcourt la base de transactions seulement une et une seule fois. Ces algorithmes permettent dans un premier temps d'organiser et de résumer l'espace de base de transaction et dans un deuxième temps de réduire le temps d'exécution.

Le plan du mémoire est comme suit : Dans le premier chapitre, nous présentons le processus de découverte de connaissances(KDD) et définissons les étapes qui le composent et aussi les tâches et les techniques de fouille de données. Dans le deuxième chapitre, nous nous intéressons aux algorithmes séquentiels exacts ainsi qu'aux classifications qui existent dans l'état de l'art des algorithmes d'extraction des règles d'association. Ensuite, nous introduisons notre contribution ainsi que les résultats obtenus, les tests et l'explication respectivement dans les troisième et quatrième chapitres.

CHAPITRE 01 :

Etat de l'art sur la fouille de données

I. Introduction :

Avec le développement des outils informatique durant ces dernières années, des informations massives qui sont stockées dans une grande base de données scientifique, économique, médicale et financière, le besoin d'interpréter et d'analyser ces grandes masses de données a suscité beaucoup d'intérêt et la mise au point de nouvelles techniques d'analyse est devenue un réel défi pour la communauté scientifique. Pour répondre à cette insuffisance de connaissances sur les données, de nouvelles méthodes d'extraction de l'information ont vu le jour, regroupées sous le terme générique de fouille de données [4].

La fouille de données consiste à rechercher et extraire des connaissances à partir de grands volumes de données stockées dans des bases ou des entrepôts de données. Le développement de la fouille de données est lié à plusieurs facteurs: une puissance de calcul importante, un volume sans cesse croissant des bases de données, un accès plus simple aux réseaux et un accroissement du débit ; ces facteurs rendent possible le calcul distribué et la distribution d'information sur un réseau d'échelle mondiale mais aujourd'hui la fouille de données hétérogènes par exemple l'extraction de connaissances à partir de texte, d'image, de vidéo, l'analyse des pratiques et stratégies commerciales et leurs impacts sur les ventes, la classification d'objets (astronomie, ...).

La fouille de données s'intègre dans le processus d'extraction des connaissances à partir des données ECD ou (KDD : Knowledge Discovery from Data en anglais).

Ce domaine en pleine expansion est souvent appelé le data mining.

La fouille de données est souvent une étape difficile à mettre en œuvre, car elle est très coûteuse et les résultats doivent en être interprétés et relativisés.

Dans ce chapitre, nous montrerons le processus de découverte de connaissances(KDD) et les différentes étapes le composant ainsi que les techniques de fouille de données.

II. Processus de découverte de connaissances(KDD) :

Le processus d'extraction de connaissances à partir de données ECD ou KDD (Knowledge Discovery from Data) est un processus qui permet d'interpréter un grand ensemble de données afin d'extraire des connaissances exploitables par l'utilisateur. Il gère automatiquement des connaissances implicites dans des bases de données. Il est composé de six étapes [5]: Sélection de données, nettoyage de données, transformation de données, modélisation, validation, intégration des résultats. (*Figure 1.1*)

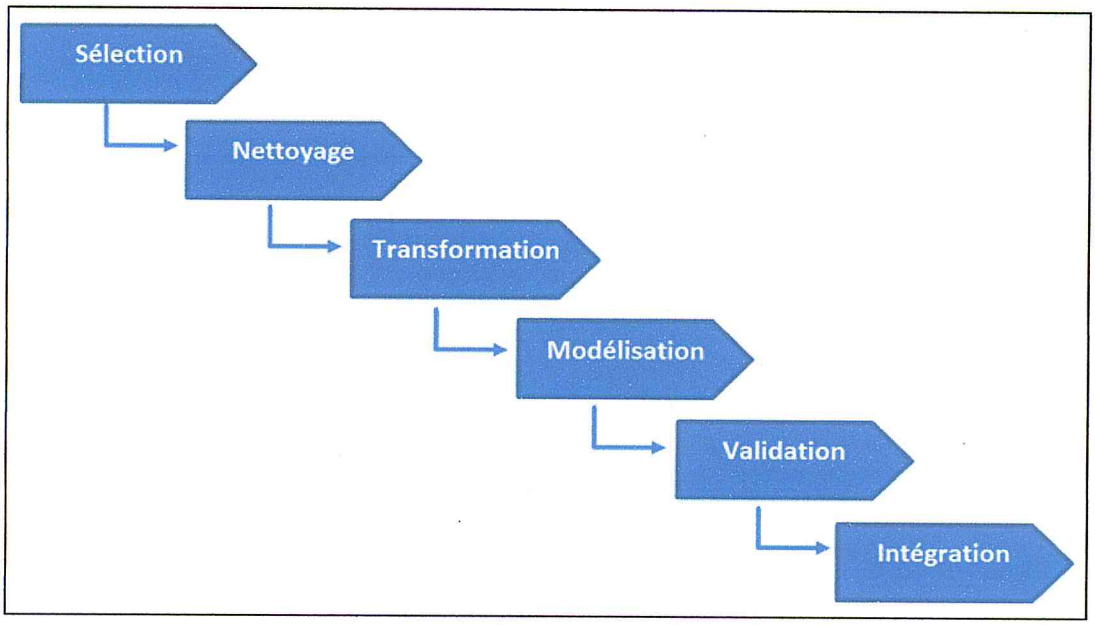


Figure 1.1 Processus de KDD.

1. Sélection :

Consiste dans un premier temps à obtenir des données en accord avec les objectifs que l'on s'impose. Ces données proviennent le plus souvent de bases de production ou d'entrepôts. Plusieurs traitements peuvent être utiles dans cette étape :

- Ignorer certaines variables absolument non pertinentes ou non discriminantes par rapport à l'objectif à atteindre, au phénomène à détecter.
- Rassembler plusieurs variables en une seule.
- Réduire le nombre de variables au moyen de l'analyse factotielle.

2. Nettoyage :

Cette étape consiste à résoudre le problème de consistance des données : cette inconsistance peut être locale à un enregistrement (une erreur de frappe, valeur manquante, valeur aberrante), locale à une source (une même personne a deux adresses différentes) ou bien entre sources (différents codages de même donnée ex : M=1 ou F=0 pour le sexe de personne, différence de granularité, différence d'unités, utilisation de synonymes) [5].

Correction des erreurs :

Pour les valeurs aberrantes on exclut les valeurs se trouvant à l'extérieur de [Moy-EcartType, Moy+EcatType], mais pour les valeurs manquantes, soit on exclut les enregistrements incomplets ou bien on remplace les données manquantes par la moyenne calculée ou héritée [5].

3. Transformation :

Il faut transformer les données pour qu'elles soient exploitables par des outils de modélisation. Il peut être soit par modification de type par exemple l'attribut 'ancienneté' est mieux que 'Date du 1er achat' et 'Date du dernier achat' ou bien normalisation (échelle uniforme) [5].

Il y a deux phases de normalisation :

1- Trouver la déviation moyenne s_{fN}pour tous les objets : on dispose de n objet et de k attribut, m_f est la moyenne de l'attribut f.

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|)$$

2- Calculer la mesure normalisée Z.

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

Exemple :

Soit la table suivante :

	Attribut 1 (brut)	Attribut 1 (normalisé)
Objet 1	23	-1,49
Objet 2	55	1,026
Objet 3	48	0,47

Table 1.1 Normalisation de données.

m_1= (23+55+48)/3=42.

S_1=1/3(|23-42|+|55-42|+|48-42|) =12,67.

Z_{11}= (23-42)/12,67= -1,49.

Z_{12}= (55-42)/12,67= 1,026.

Z_{11}= (48-42)/12,67= 0,47.

Il existe deux types de représentation de données verticale et horizontale, par exemple :

	Attribut
Objet1	Val 1
Objet2	Val 2, Val 3
Objet3	Val 3

Table 1.2 Représentation verticale de données.

	Val 1	Val 2	Val 3
Objet 1	1	0	0
Objet 2	0	1	1
Objet 3	0	0	1

Table 1.3 Représentation horizontale de données.

4. **Modélisation (fouille de données):**

C'est un processus itératif et interactif de découverte dans les bases de données larges de modèles de données valides, utiles et compréhensibles. Elle est considérée comme le cœur du processus de KDD [5].

5. **Validation :**

Pour Permettre de valider le processus de fouille, il y a deux modes de validation :

1. Validation par l'expertise telle que le diagnostic médical : pour que le modèle produit soit compréhensible, l'expert (médecin) va donc valider le modèle fouillage.
2. Validation automatique (statistique) utilisée pour le problème d'apprentissage, les données sont divisées en trois ensembles :
   - Apprentissage : générer le modèle du fouillage.
   - Validation : permet de valider le résultat de l'ensemble apprentissage au cours du fouillage.
   - Test : permet de valider le modèle après le fouillage.

6. **Intégration :**

Ce processus se fait par l'implémentation des modèles et ses résultats ou par la prise des décisions et appliquer les actions correspondantes dans le processus de l'entreprise [5].

III. La fouille de données:

La fouille de données ou « Data Mining » est un ensemble de techniques et de méthodes destinées à l'exploration et l'analyse de grandes BDD de façon automatique ou semi-automatique en vue de détecter des règles, des tendances inconnues ou cachées ou des structures particulières [5].

1. Les mesures de similarité :

La mesure de similarité présente par une distance avec une fonction dépend du type de données binaires, nominales ou continues. La distance doit satisfaire les propriétés suivantes :

- d (x, y) ≥ 0
- d (x, x) = 0 ↔ identité
- d (x, y) = d (y, x) 0 ↔ symétrie
- d (x, y) ≤ d (x, z) + d (z, y) ↔ inégalité triangulaire

On peut représenter la distance dans une matrice symétrique appelée matrice de distance. (*Table 1.4*)

0				
d(2,1)	0			
d(3,1)	d(3,2)	0		
…	….	….	…	
d(n,1)	…	…	d(n,n-1)	0

Table 1.4 Matrice de distance.

1.1. Les mesures de similarité entre données :

La fonction de distance pour les attributs numériques est calculée de plusieurs manières, parmi elles :

- ***la distance de Minkowsky :***

$$d(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{1/p}$$ (Pour des vecteurs de dimension *n*).

- ***la distance Euclidienne :*** est un cas particulier pour p = 2 de la distance de Minkowsky. La distance la plus connue.

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

- ***la distance de Manhattan*** **:**aussi appelée distance « city-block » ou métrique absolue, c'est un cas particulier de pour p = 1 du Minkowsky.

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i|$$

Pour les attributs binaires, la distance est calculée selon le type de variable à partir de table de contingence. (*Table 1.5*)

RMQ :

d (0, 0)=d (1, 1)=0

d (0, 1)=d (1, 0)=1.

		Objet y	
		1	0
Objet x	1	A	B
	0	C	D

Table 1.5 Table de contingence.

- ***Variable symétrique :***

C'est-à-dire le nombre de (0) égale le nombre de (1).

$$d(x,y)=\frac{b+c}{a+b+c+d}$$

- ***Variable asymétrique :***

C'est-à-dire le nombre de (0) diffère du nombre de (1).

$$d(x,y)=\frac{b+c}{a+b+c}$$

Exemple :

Individu	Test1	Test2	Test3	Diabète	hypertension
Yacin	+	+	+	Oui	Non
Ali	+	+	-	Non	Oui
Mohammed	-	-	-	Oui	Non

Table 1.6 Résultat de tests des individus.

Les variables Yacin et Ali sont asymétriques, donc :
d (Yacin, Ali) = (2+1)/ (2+2+1) =3/5.
Les variables Yacin et Mohammed sont symétriques, donc :
d (Yacin, Mohammed) = (3+0)/ (1+3+0+1) =3/5.

Mais, pour les attributs nominaux, il existe plusieurs fonctions de calcul de distance, parmi elles :

- ***La distance de Jaccard, ou encore score de Jaccard :***

$$d(x,y)=\frac{|x\cap y|}{|x\cup y|}$$

Exemple :

Soient X1= (a1,b1,c1) et X2= (a1,b2,c1),
d(X1,X2)=|{a1,c1}|/|{a1,b1,b2,c1}|=1/2.

1.2. Les mesures de similarité entre clusters :

La distance entre deux clusters (classes) peut être la plus proche métrique, la métrique moyenne ou la plus loin métrique.

- ***Lien simple (singlelikage) :*** c'est la plus petite distance entre tous les éléments des deux clusters.

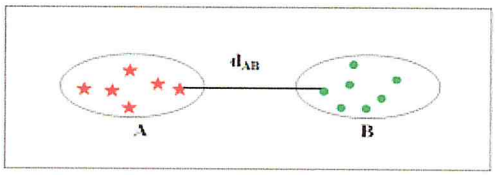


Figure 1.2 Lien simple entre les clusters A et B.

- ***Lien moyenne (averagelikage) :*** similarité moyenne entre les éléments des deux clusters. Consiste à calculer la distance entre les centroides des deux clusters.
  (Le centroide est le centre logique de cluster).

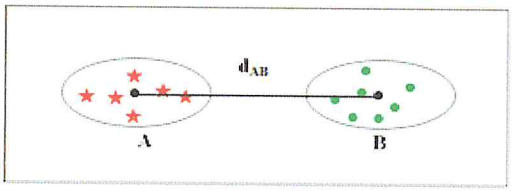


Figure 1.3 Lien moyenne entre les clusters A et B.

- ***Lien complet (completelikage) :*** c'est la plus grande distance entre tous les éléments des deux clusters.

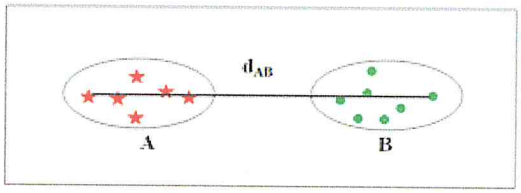


Figure 1.4 Lien complet entre les clusters A et B.

La construction de modèle se fait par deux étapes, le choix de type de modèle (par exemple classification) et choix de méthode pour construire ce modèle (par exemple arbre de décision) [6].

2. Les tâches de la fouille de données :

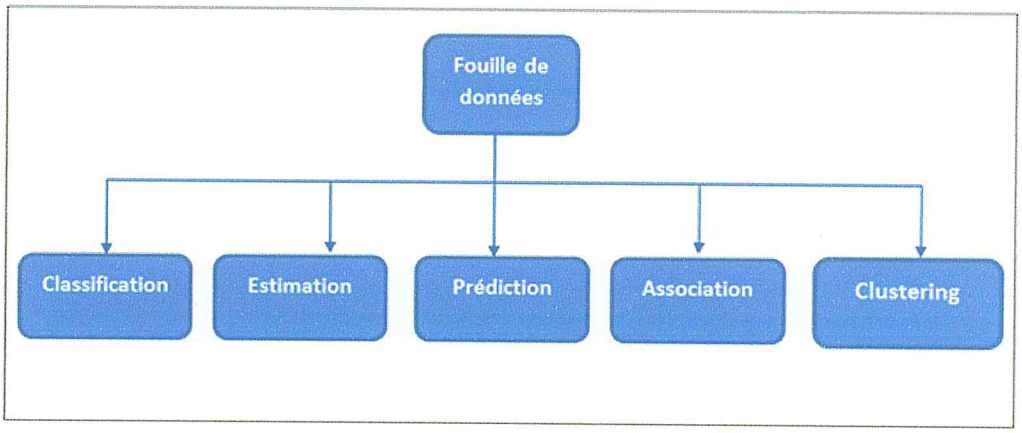


Figure 1.5 les taches du fouille de données.

2.1. Classification :

La classification est la tâche la plus commune de la fouille de données qui semble être une tâche humaine primordiale. Afin de comprendre notre vie quotidienne, nous sommes constamment obligés à classer, catégoriser et évaluer.

La classification consiste à étudier les caractéristiques d'un nouvel objet pour l'attribuer à une classe prédéfinie. Les objets à classifier sont généralement des enregistrements d'une base de données, la classification consiste à mettre à jour chaque enregistrement en déterminant la valeur d'un champ de classe [6].

2.2. Estimation :

L'estimation est similaire à la classification à part que la variable de sortie est numérique plutôt que catégorique. En fonction des autres champs de l'enregistrement, l'estimation consiste à compléter une valeur manquante dans un champ particulier [6].

2.3.Prédiction :

Consiste à prédire la valeur future d'un attribut en fonction d'autres attributs. Elle se base sur le passé et le présent et le résultat se situe dans le futur. Par exemple, prédire la qualité d'un client en fonction de son revenu, de son nombre d'enfants,… [6]

2.4.Clustering:

Le Clustering(ou classification non supervisé) est le regroupement d'enregistrements ou des observations en classes d'objets similaires. Un cluster est une collection d'enregistrements similaires l'un à l'autre, et différents de ceux existants dans les autres clusters. La différence entre le Clustering et la classification est que dans le Clustering, il n'y a pas de variables sortantes [6].

2.5.Association :

L'association consiste à déterminer quels attributs "vont ensemble". La tâche la plus répandue dans le monde du business, est celle appelée l'analyse d'affinité ou l'analyse du panier du marché, elle permet de rechercher des associations pour mesurer la relation entre deux ou plusieurs attributs. Les règles d'associations sont, généralement, de la forme Si "cause" alors "conséquence" [6].

3. Les techniques de la fouille de données :

Chaque tâche de la fouille de données possède une ou plusieurs techniques. Parmi ces techniques, nous avons :

3.1.Les techniques utilisées pour la classification:

Pour la classification, il existe KPPV, Arbre de décision, … etc. Nous nous intéressons à l'algorithme suivant :

L'algorithme de k-plus proche voisin :

L'objectif de l'algorithme est de classer un nouvel individu (non étiqueté sur la base de leur similarité) avec les individus déjà classés.

Code Algorithme K plus proches voisins :

Début
Pour chaque individu (x',c)∈ *L faire*
| Calculer la distance d(x',x).
Fin pour
Pour chaque {x' ∈ K plus proche (x)} *faire*
| Calculer le nombre d'occurrence de chaque classe.
Fin pour
Attribuer à x la classe la plus fréquente ;
Fin.

3.2.Les techniques utilisées pour le Clustering :

Pour le Clustering, il existe K-means, CHA, etc… Nous nous intéressons aux deux algorithmes suivants :

L'algorithme de CHA (Clustering HierarchiAscud) :

Cet algorithme fait la classification à partir de la décomposition hiérarchique d'ensembles d'objets par les étapes suivantes :

Nécessité de définir une distance entre groupes d'individus et de choisir le nombre de classes à retenir.

1- Initialisation : les classes initiales sont les singletons-individus. Calculer la matrice de leurs distances deux à deux.
2- Itérer les deux étapes suivantes jusqu'à l'agrégation en une seule classe :
   - Regrouper les deux classes les plus proches au sens de la distance entre groupes choisis.
   - Mettre à jour la matrice de distance en remplaçant les deux classes regroupées par la nouvelle et en calculant sa distance avec chacune des autres classes.

L'algorithme de k-means (Clustering de partitionnement) :

K-means est un algorithme de minimisation alternée qui, étant donné un entier *K*, va chercher à séparer un ensemble de points en *K* clusters. Par les étapes suivantes :

1- Initialisation : initialiser K centres μ1,…, μk.
2- Répéter les étapes suivants jusqu'aux *K* clusters pour qu'ils soient stables :
   - Affecter chaque élément à son cluster *Ci* le plus proche.
   - Recalculer le centre μi de chaque cluster tel que :

$$\mu i = \frac{1}{|Ci|} \sum xj \ (j = 1 \dots |ci|$$

IV. Conclusion :

Dans ce chapitre, nous avons donné un aperçu sur le processus de découverte de connaissances(KDD) et la fouille de données. Nous avons également introduit les différentes tâches et techniques de fouille de données. Dans notre travail, nous allons spécifier la technique de génération des règles d'association, et nous montrerons ses différents algorithmes dans le chapitre suivant.

CHAPITRE 02 :

Etat de l'art sur les algorithmes d'extraction des règles d'association

I. Introduction :

L'extraction des règles d'association est un des principaux problèmes d'extraction des connaissances à partir de données ECD. Elle est le processus de découverte d'un ensemble de règles pertinentes à partir d'une base de transactions. Chaque transaction est un ensemble d'items alors qu'un item est la donnée la plus élémentaire du problème rencontré. La plupart des algorithmes de recherche des règles d'association utilise un seuil minimum de confidence. Cependant, plusieurs règles intéressantes peuvent être trouvées en utilisant un seuil très faible. Bien que l'utilisation d'un support minimum empêche l'exploration d'un grand nombre de règles intéressantes, plusieurs règles ainsi trouvées restent inutiles pour l'utilisateur. Il existe plusieurs catégories des algorithmes tels que séquentiels et parallèles. Dans ces catégories, il y a des algorithmes exacts et approchés. Dans ce chapitre, nous nous intéressons aux algorithmes séquentiels exacts qui donnent toutes les règles pertinentes, ainsi que les classifications qui existent dans l'état de l'art des algorithmes d'extraction des règles d'association.

II. Concepts de base :

1. Définitions :

- ***Item et itemset :***

  Un item c'est l'élément de base de problème d'extraction des règles d'association. Un itemset peut être un seul item ou un ensemble d'item. On dit que k-itemset est un ensemble d'items de taille k.

- ***Base de transactions :***

  Est un ensemble d'itemset qui peut constituer une base de données, elle représente le problème à résoudre.

- ***Règle d'association :***

  C'est la relation entre deux itemsets appartenant à un ensemble d'items I dans une base de transactions T un représenté comme suite : $X \rightarrow Y$ (si X alors Y), X et Y sont des ensembles d'items inclus dans I "itemset" telle que $X \cap Y = \emptyset$.

  On dit règles pertinentes pour les règles qui ont un support et une confiance >= minSup et minConf respectivement (minSup et minConf choisis par l'utilisateur).

- ***Mesure d'évaluation :***

Nous avons premièrement deux mesures (support, confiance) :

- Le support d'un itemset I, noté Support(I), est la proportion de transactions de T contenant I. La valeur du support est comprise entre [0,1]. Il peut également être exprimé en pourcentage. Le support de règle $X \rightarrow Y$ est le support de $X \cup Y$.
- La confiance d'une règle, notée Confiance(R), correspond au rapport du nombre de transactions où apparaissent simultanément les items de la condition et de la conclusion sur le nombre de transactions où apparaissent les items de la condition. La confiance permet de mesurer la force de l'association.

La confiance représentée comme suite :

$$\frac{Support(X \cup Y)}{Support(X)}$$

D'autres mesures d'évaluations ont été proposées comme suit [7]:

$$Lift(X \rightarrow Y) = \frac{confidence(X \rightarrow Y)}{support(Y)}.$$
$$Leverage(X \rightarrow Y) = support(X \rightarrow Y) - (support(X) \times support(Y)).$$
$$Coverage(X \rightarrow Y) = support(X).$$
$$Conviction(X \rightarrow Y) = \frac{1 - support(Y)}{1 - confiance(X \rightarrow Y)}.$$
$$Information - Gain(X \rightarrow Y) = \log \frac{m \times support(X \rightarrow Y)}{support(X) \times support(Y)}.$$

- ***Les types de représentation de la base de transactions :***

Dans la littérature, il y a trois types de représentations de la base de transactions et qui sont [8][9]:

- La représentation horizontale : Elle est la plus utilisée dans les algorithmes d'extraction des règles d'associations, chaque transaction est représentée par un ensemble d'items.
- La représentation verticale (tidlists) : la base de transactions est représentée par un ensemble d'items, chaque item est défini par l'ensemble de transactions qui le contiennent.
- La représentation sous forme de bitmap : Elle est très utilisée dans le contexte parallèle, chaque ligne représente une transaction et dans chaque transaction on définit les items qu'elle contient tel que : Bitmap[i][j]=1 si l'item *j* est inclus dans la transaction *i*, 0, sinon.

2. Exemple :

Soient la base de transactions et l'ensemble d'items suivants :

T= {t_1, t_2, t_3, t_4, t_5} et I= {a, b, c, d, e} tel que :

TID	Items
t_1	{a, b}
t_2	{b, c, d}
t_3	{a, b, c}
t_4	{e}
t_5	{c, d, e}

Table 2.1 Exemple d'une base de transaction.

t_1	a	b	
t_2	b	c	d
t_3	a	b	c
t_4	e		
t_5	c	d	e

Table 2.2 La représentation horizontale de l'exemple (Table 2.1).

Items	t_1	t_2	t_3	t_4	t_5
a	1	0	1	0	0
b	1	1	1	0	0
c	0	1	1	0	1
d	0	1	0	0	1
e	0	0	0	1	1

Table 2.3 La représentation verticale de l'exemple (Table 2.1).

Transactions	a	b	c	d	e
t_1	1	1	0	0	0
t_2	0	1	1	1	0
t_3	1	1	1	0	0
t_4	0	0	0	0	1
t_5	0	0	1	1	1

Table 2.4 La représentation bitmap de l'exemple (Table 2.1).

Calcul des mesures pour l'exemple (*Table 2.1*):

L'item {a} apparaît 2 fois (t_1 et t_3) dans la base de transaction (t_1, t_2, t_3, t_4 et t_5) de taille 5, donc :

Support ({a})=2/5 ;

Et de même :

Support ({b})=3/5 ; Support ({c})=3/5 ; Support ({d})=2/5 ; Support ({e})=2/5

Soient minSup=2/5 et minConf=3/5 :

Donc {ab, ac, ad, ae, bc, bd, be, cd, ce, de}appartient à l'ensemble 2-itemset,

Confiance (a→b)=Support ({a, b})/Support ({a})=1 (c'est une règle sûre car leur confiance est égale à 1);

Confiance (b→d)=1/3 (c'est une règle partielle ou approximative car leur confiance < 1).

III. Les algorithmes d'extraction des règles d'association basés sur *générer* et *tester* :

1. L'algorithme Apriori :

Apriori [10], c'est le premier algorithme d'extraire des itemsets à partir d'une base de transaction [11], car il est à la base de la majorité des algorithmes servant à découvrir des règles d'association telles que les associations séquentielles. C'est un algorithme permettant de générer toutes les règles d'association satisfaisant aux seuils de support et de confiance choisis au départ à partir des deux étapes suivantes :

Génération des itemsets fréquents: La détermination de k-itemset fréquent (l'ensemble de itemsets avec un support >= minSup) se fait de manière récursive à partir de (k-1)-itemset et tout sous-itemset d'un itemset fréquent doit être fréquent. Pour construire l'ensemble de k-itemset candidat, on applique le jointeur sur (k-1)-itemset, ensuite on doit calculer le support de chaque itemset et éliminer les itemsets qui ont un support <minSup.

Génération des règles d'association : A partir de l'ensemble d'itemsets fréquents trouvés dans la première étape, pour chaque itemset fréquent, on génère toutes les règles possibles afin de trouver l'ensemble de règles fréquent (les règles qui ont une confiance >= minConf).

- ***Pseudo code d'Apriori :***

```
Entrée : L1= {items fréquents}
         minSupport, minConfiance
For (k=2 ; Lk-1 ; k++) {
  Ck=génér_candidat (Lk-1) ;
  Foreach c ϵ Ck {
    Count=0 ;
    Foreach t ϵT {
      If c ϵ t then count++ ;
                }
     If count >= minSupport then {
       Lk=LkU{c} ;
       Foreach subset s != Ø of c {
             Confiance(s →c-s)=Support(c)/Support(s) ;
         If confiance >=minConfiance then Règles=Règles U {s →c-s} ;
                                          }
                                        }
        L=LULk ;
}
}
Return Règles ;
```

2. Exemple de l'algorithme Apriori :

On utilise l'exemple précédent (*Table 2.1*) :

Soient minSup=2/5 et minConf=3/5:

Etape 1 : génération des itemsets fréquents :

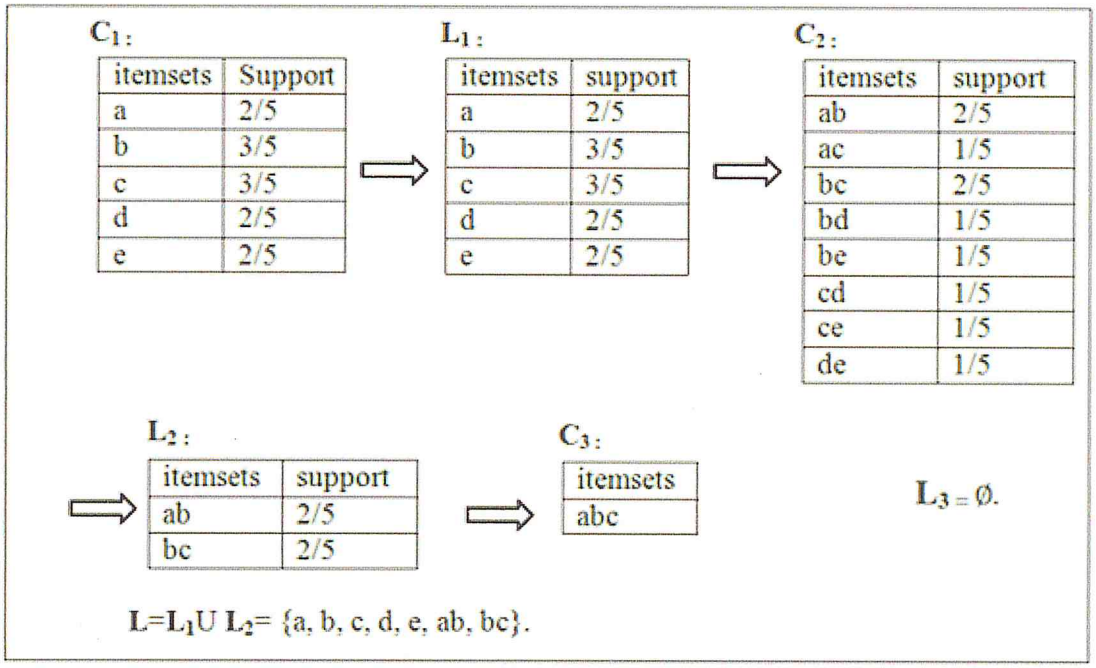


Figure2.1 Exemple de génération des règles d'association.

- On parcourt la base de transactions afin de calculer le support de chaque item dans C_1= {a, b, c, d, e}.
- On compare le support de chaque candidat avec le minSupport afin d'extraie L_1= {a, b, c, d, e}.
- On génère les candidats C_2 à partir de L_1, (L_1*L_1), C_2= {ab, ac, bd, bc, be, cd, de, ce}.
- On parcourt la base de transactions afin de calculer le support de chaque itemset dans C_2.
- On compare le support de chaque candidat avec le minSupport afin d'extraire L_2 = {ab, bc}.
- On génère les candidats C_3 à partir de L_2, (L_2*L_2), C_3= {abc}.
- On parcoure la base de transactions afin de calculer le support de chaque itemset dans C_3.
- On compare le support de chaque candidat avec le minSupport afin d'extraire L_3 = {}.
  Donc L=L_1UL_2 = {a, b, c, d, e, ab, bc}.

Etape 2 : génération de règles fréquentes

On a L= {a, b, c, d, e, ab, bc}

Confiance (a→b)=1>minConf
Confiance (b→a)=2/3>minConf
Confiance (b→c)=2/3>minConf
Confiance (c→b)=2/3>minConf
Donc :
R= {a→b, b→c, b→a, c→b} c'est l'ensemble des règles fréquents.

- ***La complexité :***

Apriori réduit proportionnellement la complexité de problème d'extraction des règles d'association, leur complexité au pire des cas est O (n^2*m) telle que m représente le nombre de transactions et n représente le nombre des items.

3. Algorithmes extension de l'algorithme Apriori :

DHP [12] (the Direct Hashing Pruning) qui est une extension de l'algorithme Apriori utilisant une mémoire supplémentaire afin de calculer à l'avance les 2-itemsets fréquents au cours de la première itération. Aussi, le DHP réduit, progressivement, la taille de la base de transactions en éliminant à chaque fois les itemsets non fréquents [13].

DIC [14] (Dynamic Itemsets Counting) est la généralisation d'Apriori où la base de transactions est partitionnée en plusieurs parties de même taille, telles que chacune d'elles est allouée en mémoire. Initialement, les supports d'items sont calculés pour la première partition. Les items trouvés localement, sont utilisés pour générer les candidats 2-itemsets. Ensuite, la deuxième partition est lue afin de trouver le support de tous les candidats actuels, ce processus est répété pour les autres partitions. Après le traitement de la dernière partition, le même processus se répétera jusqu'à ce que aucun candidat ne peut être généré [13].

Apriori partition [15] qui divise logiquement la base de transactions en partitions totalement disjointes. Dans la première passe, chaque partition est lue afin de construire les tidlists des items. Ensuite, on génère localement tous les itemsets fréquents en appliquant des intersections entre les tidlists. Dans la deuxième passe, les itemsets fréquents de toutes les partitions se fusionnent pour former un ensemble global d'itemsets fréquents [13].

SEAR [16] est identique à Apriori, sauf que SEAR stocke les candidats dans une structure appelée **prefixtree** au lieu de la structure **hash tree**. Dans **prefixtree** (également appelée Trie), chaque arête est masquée par des items : les préfixes communs sont représentés par des branches d'arbres, et les **sufixes** uniques sont stockés dans les feuilles. Les tests de Mueller ont montré l'efficacité de la structure **prefixtree** par rapport à la structure **hash tree** lors de calcul des supports des itemsets candidats [13].

ECLAT [17] utilise la représentation verticale des items. Les k-itemsets fréquent sont organisés dans des classes d'équivalence disjointes par les (k-1) **préfixes** communs, de telle sorte que les (k+1) itemsets candidats peuvent être générés en rejoignant les paires de k-itemsets fréquents dans les mêmes classes. Le support d'un itemset candidat peut, alors, être calculé simplement en s'intéressant juste aux tidlists de sa classe [13].

Tous les algorithmes expliqués précédemment, nécessitent plusieurs accès à la base de transactions et génère un nombre important des itemsets candidats.

IV. Les algorithmes d'extraction des règles d'association basés sur *diviser* pour *régner* :

1. L'algorithme FP-growth :

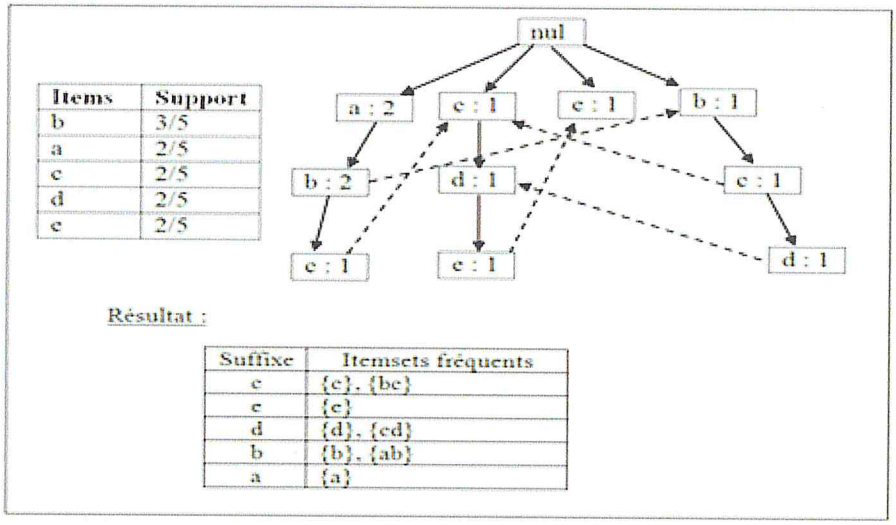
FP-growth [18] utilise une structure appelée **FP-tree** pour la compression de la base de transactions. L'algorithme est divisé en deux phases.

La construction de la structure FP-tree : FP-tree est un arbre dont chaque nœud contient un item avec sa fréquence. Initialement, l'arbre contient seulement la tête avec la valeur nulle. Pour chaque transaction *t*, on doit parcourir l'arbre avant d'insérer les items de *t* dans le bon endroit de l'arbre. En effet, si l'item courant existe dans le chemin actuel alors on incrémente sa fréquence par 1. Sinon, on crée un nouveau nœud pour cet item dans le chemin actuel avec une fréquence de valeur 1.

Extraction des itemsets fréquents : On produit les itemsets fréquents directement à partir de la structure **FP-tree**, en utilisant la stratégie profondeur d'abord.

2. Exemple de l'algorithme FP-growth :

On utilise l'exemple précédent (*Table 2.1*), soit minSup=2/5.



Items	Support
b	3/5
a	2/5
c	2/5
d	2/5
e	2/5

Suffixe	Itemsets fréquents
e	{e}, {be}
c	{c}
d	{d}, {cd}
b	{b}, {ab}
a	{a}

Figure 2.2 Exemple de construction de la structure FP-tree.

- Parcours la base de transactions, trouve le 1-itemset fréquent {a, b, c, d, e}.
- Extraire les items fréquents par ordre décroissant (b, a, b, d, e).
- Parcours la base de transactions afin de construire FP-tree.

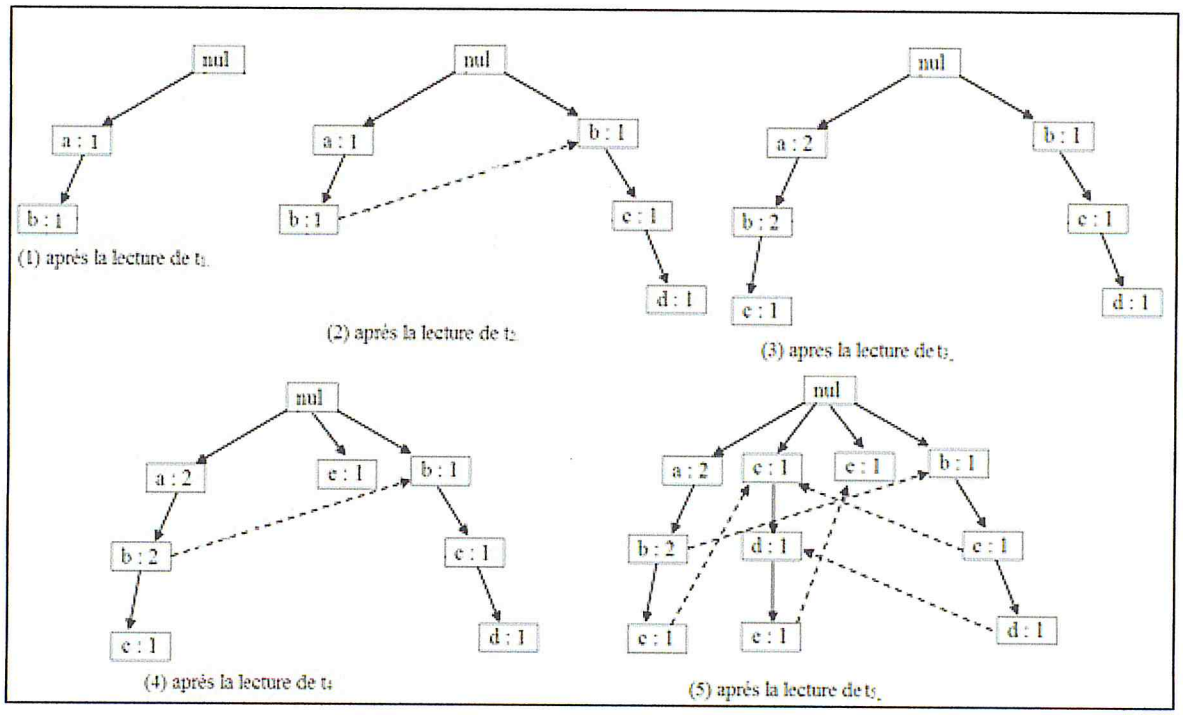


Figure 2.3 détail de construction de FP-tree.

3. Algorithmes extension de l'algorithme PF-growth :

FI-growth [19] représente l'ensemble de données dans une structure **appelée FI-tree**. Il fournit aussi une opération combinaison qui permet de fusionner deux sous-arbres de même tête de telle sorte qu'on additionne les fréquences de la tête et on fusionne les branches des sous-arbres. FI-growth se compose de deux phases : la construction de la structure **FI-tree** et l'extraction des itemsets fréquent à partir de FI-tree. La construction de **FI-tree** est réalisée comme dans FP-growth tandis qu'il existe principalement trois phases dans l'extraction des itemsets fréquents qui sont : La ramification, La recherche des branches restantes et La suppression [13].

V. Classification des algorithmes des règles d'association :

Il existe plusieurs classifications pour les algorithmes d'extraction des règles d'association selon des critères précisés. Parmi ces classifications, nous nous intéressons à deux catégories : séquentiel et parallèle.

Les algorithmes séquentiels sont divisés sur deux autres catégories exactes et approchées : Les algorithmes séquentiels exacts pouvant être basé sur (générer et tester) de manière récursive pour déterminer les itemsets fréquents, ou bien sur (diviser pour régner) à partir d'un arbre afin de trouver les itemsets fréquents. Les algorithmes séquentiels approchés permettent d'extraire une partie des règles aussi, Les algorithmes parallèles sont divisés sur deux catégories exactes et approchés : Les algorithmes parallèles exacts qui sont implémentés sur différentes architectures (mémoire distribuée, mémoire partagée, ou la technologie GPU).Les algorithmes parallèles approchés sont basés sur les algorithmes génétiques [13].

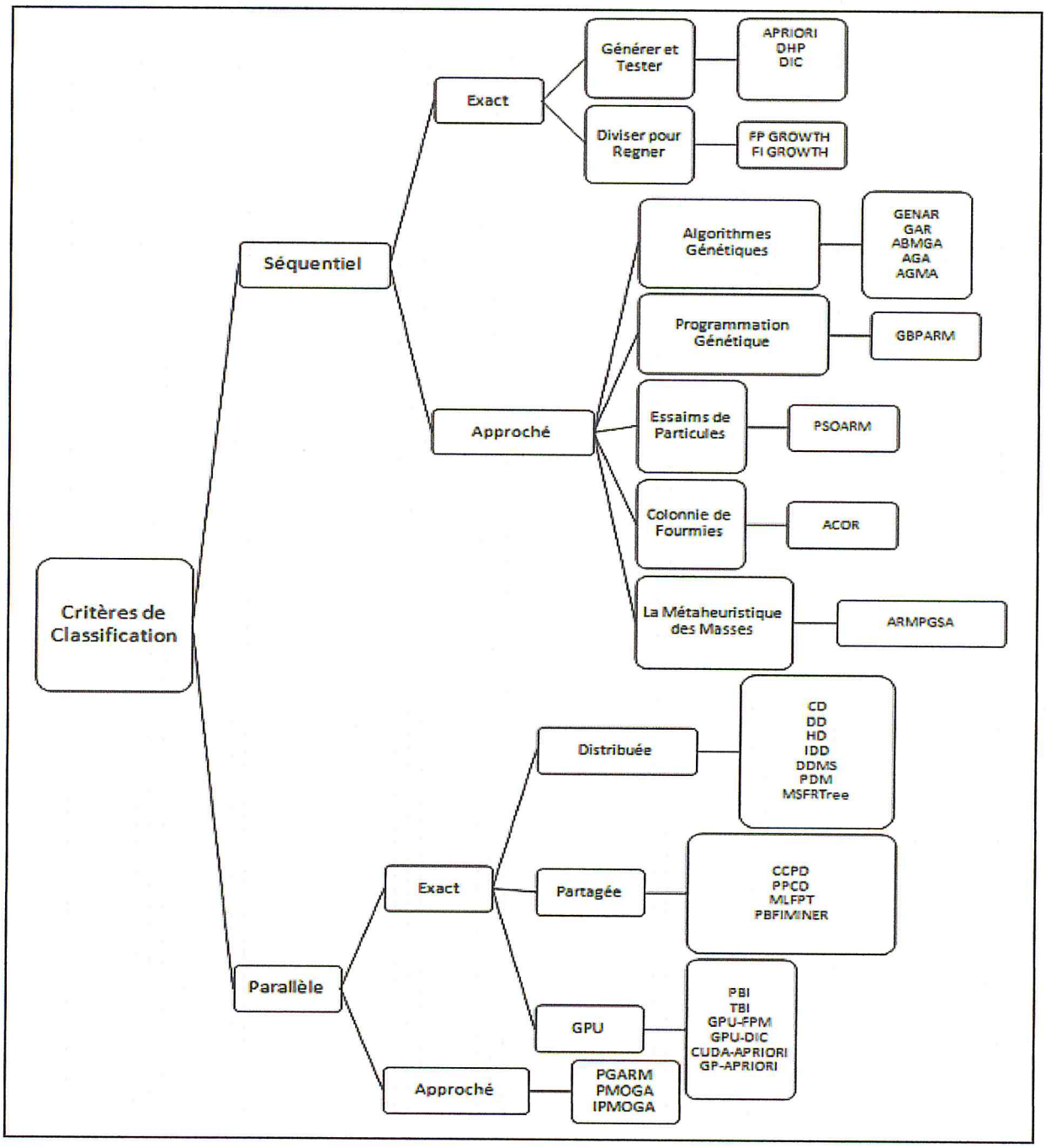


Figure2.4 Classification des algorithmes d'extractions des règles d'association [13].

VI. Conclusion :

Apriori et FP-growth sont les deux principaux algorithmes exacts d'extraction des règles d'associations. L'algorithme FP-growth est mieux que l'Apriori car il ne fait pas la génération des candidats, mais le FP-growth prend un grand espace mémoire lors de la construction du FP-tree. Malgré que l'algorithme Apriori devient lourd (coûteux en temps), production d'un nombre important des règles triviales et inutile aussi la méthode non efficace pour les articles rares mais il reste le plus utilisé.

En outre, dans les prochains chapitres nous allons améliorer l'Apriori qui est la base de tous les algorithmes d'extraction des règles d'association.

CHAPITRE 03 :

Contribution

I. Introduction :

Conjointement au développement technologique, les organisations ont pu élargir leurs champs de travail, par ce qu'il leur est possible de stocker des bases de transactions volumineuses.

L'algorithme d'Apriori est le premier algorithme d'extraction des règles d'association mais, avec les bases de transactions actuelles, il devient suffisant car il prend beaucoup de temps d'exécution ; d'autres algorithmes ont donc été développés pour la résolution de ce problème (FP-growth, DHP, DIC, etc…).

Dans ce chapitre, nous allons proposer deux nouvelles approches améliorant la technique d'Apriori.

II. Définitions :

1. Table de hachage :

Une table de hachage est une structure de données qui permet une association clé-élément, c'est-à-dire une implémentation du type abstrait tableau associatif.

2. Base de transactions condensée :

C'est une base de transactions qui contient plusieurs items par une transaction et un nombre de transactions limité.

3. Base de transactions non condensée :

C'est le contraire d'une base de transactions condensée, elle comprend un nombre très grand de transactions de petite taille.

III. La première approche :

Il est bien connu que l'étape la plus complexe et la plus consommatrice en temps d'exécution est celle de la découverte des itemsets fréquents. Cette étape peut générer un nombre important d'itemsets fréquents et par conséquence de règles d'association. Apriori est le principal algorithme de problème d'extraction des règles d'association, l'inconvénient de cet algorithme est le parcours de la base de transactions et cela pour chaque k-itemset généré, ce qui nécessite un temps d'exécution considérable. Plusieurs algorithmes sont développées pour le but de diminuer le temps d'exécution d'Apriori telle que DHP, DIC, Apriori partitionné, SEAR et ECLAT mais ces algorithmes ne sont pas parfaitement complets, dans ce chapitre nous avons proposée deux nouvelle algorithmes extension d'Apriori. La différence entre la première approche qui nous avons proposé et algorithmes existant comme suit :

L'algorithme DHP est identique avec cette approche sauf que le DHP utilise une mémoire supplémentaire ce qui prend une tempe d'exécution considérable.

L'algorithme DIC nécessite un espace mémoire, car il est partitionne la base de transactions en plusieurs parties de même taille, telle que chacune d'elles est allouée en mémoire contraire notre approche que permette de réduire l'espace de la base de transactions.

Apriori partitionné est presque DIC sauf que l'Apriori partitionné divise logiquement la base de transactions en partitions totalement disjointes. Dans la première passe, chaque partition est lue afin de construire les tidlists des items, ce qui besoins un grand espace mémoire et un temps d'exécution considérable.

1. Principe :

C'est un algorithme d'extraction des règles d'association, il est considéré comme une amélioration de l'Apriori, basé sur les deux étapes suivantes:

Génération des itemsets fréquents : La détermination de k-itemset fréquent (l'ensemble d'itemsets avec un support >= minSup) se fait de manière récursive à partir de (k-1)-itemset avec une base de transactions qui ne contient que les transactions qui ont des itemsets fréquents et tout sous-itemset d'un itemset fréquent doit être fréquent. Pour construire l'ensemble de k-itemset candidat, on applique le jointeur sur (k-1)-itemset,

ensuite on doit calculer le support de chaque itemset et éliminer les itemsets qui ont un support <minSup.

Génération des règles d'association : A partir de l'ensemble d'itemsets fréquents trouvés dans la première étape, pour chaque itemeset fréquent, on génère toutes les règles possibles afin de trouver l'ensemble de règles fréquents (les règles qui ont une confiance >= minConf).

L'avantage de cette approche est la réduction de la base de transactions lors du calcul, surtout si le minimum support est grand.

2. Pseudo code de l'approche 01:

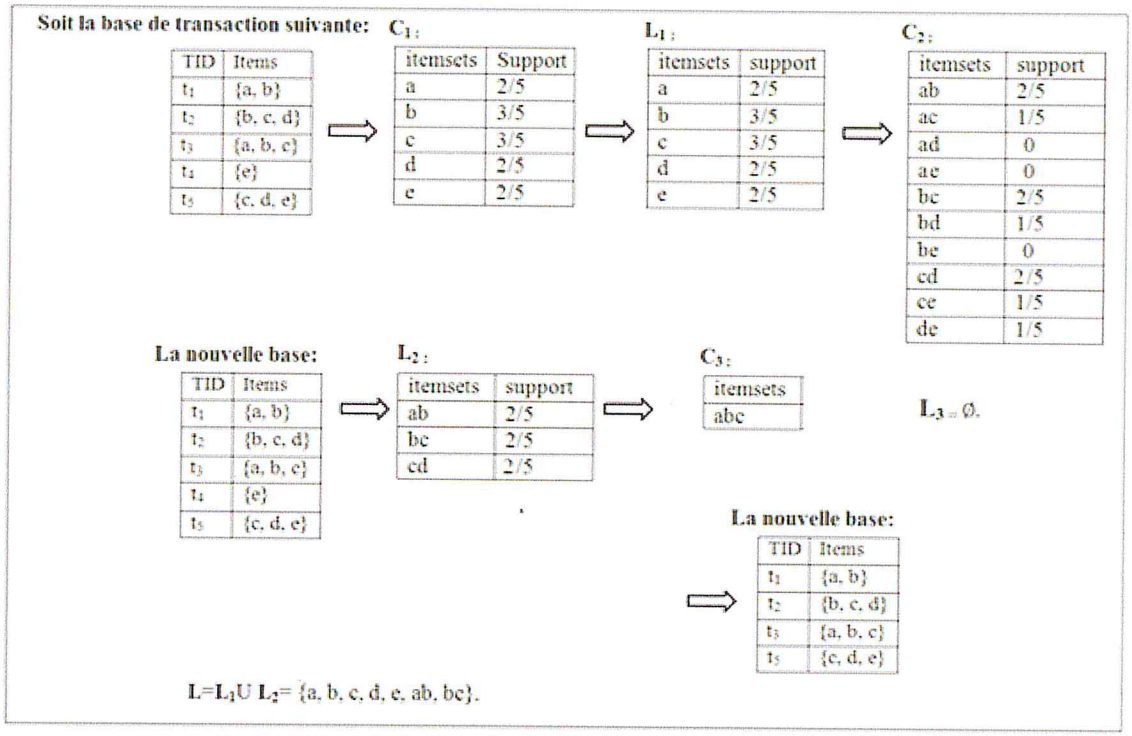
```
Entrée : L1= {items fréquents}, T
         minSupport, minConfiance
        Foreach t ϵ T {
          existe=false;
          Foreach item ϵ L1 {
            If t contain item then
              existe=true;
                }
            If existe==false then
              T=T-{t};
                }

For (k=2 ; Lk-1 !={} ; k++) {
  Ck=génér_candidat (Lk-1) ;
  Foreach c ϵ Ck {
    Count=0 ;
    Foreach t ϵT {
      If c ϵ t then count++ ;
            }
     If count >= minSupport then {
       Lk=LkU{c} ;
        Foreach subset s != Ø of c {
            Confiance(s →c-s)=Support(c)/Support(s) ;
          If confiance >=minConfiance then Règles=Règles U {s →c-s} ;
                                    }
                                  }
        L=LULk ;
       Foreach t ϵ T {
         existe=false;
        Foreach lk ϵ Lk{
          Foreach item ϵ lk {
            If t contain item then
              existe=true;
                        }
                    }
              If existe==false then
                T=T-{t};
                  }
}
}
Return Règles ;
```

3. Illustration:

Soient minSup=2/5 et minConf=3/5:

Etape 1 : génération des itemsets fréquents.



TID	Items
t_1	{a, b}
t_2	{b, c, d}
t_3	{a, b, c}
t_4	{e}
t_5	{c, d, e}

itemsets	Support
a	2/5
b	3/5
c	3/5
d	2/5
e	2/5

itemsets	support
a	2/5
b	3/5
c	3/5
d	2/5
e	2/5

itemsets	support
ab	2/5
ac	1/5
ad	0
ae	0
bc	2/5
bd	1/5
be	0
cd	2/5
ce	1/5
de	1/5

TID	Items
t_1	{a, b}
t_2	{b, c, d}
t_3	{a, b, c}
t_4	{e}
t_5	{c, d, e}

itemsets	support
ab	2/5
bc	2/5
cd	2/5

itemsets
abc

TID	Items
t_1	{a, b}
t_2	{b, c, d}
t_3	{a, b, c}
t_5	{c, d, e}

Figure3. 1Exemple de génération des items fréquents avec l'approche 01.

- On parcourt la base de transactions originale afin de calculer le support de chaque item dans C_1 = {a, b, c, d, e}.
- On compare le support de chaque candidat avec le minSupport afin d'extraire L_1 = {a, b, c, d, e}, puis on élimine toutes les transactions qui ne contiennent aucun itemset fréquent T= {t_1, t_2, t_3, t_4, t_5}.
- On Génère les candidats C_2 à partir de L_1, (L_1*L_1), C_2= {ab, ac, ad, ae, bd, bc, be, cd, de, ce}.
- On parcourt la nouvelle base de transactions afin de calculer le support de chaque itemset dans C_2.
- On compare le support de chaque candidat avec le minSupport afin d'extraire L_2 = {ab, bc}, puis on élimine toutes les transactions qui ne contiennent aucun itemset fréquent T= {t_1, t_2, t_3, t_5}.
- On Génère les candidats C_3 à partir de L_2, (L_2*L_2), C_3= {abc}.
- On parcourt la base de transactions afin de calculer le support de chaque itemset dans C_3.
- On compare le support de chaque candidat avec le minSupport afin d'extraire L_3 = {} on arrête.

  Donc L=L_1UL_2 = {a, b, c, d, e, ab, bc}.

Etape 2 : génération de règles fréquentes.

On a L= {a, b, c, d, e, ab, bc}

Confiance (a→b)=1> minConf
Confiance (b→a)=2/3> minConf
Confiance (b→c)=2/3> minConf
Confiance (c→b)=2/3> minConf
Donc :
R= {a→b, b→c, b→a, c→b} c'est l'ensemble des règles fréquents.

4. La complexité:

Cette approche réduit proportionnellement la complexité de problème d'extraction des règles d'association telle qu'elle diminue la taille de la base de transactions, leur complexité au pire des cas est comme suit :

```
Foreach t ϵ T {                    nombre d'itération =m
   existe=false;                   O(1)
  Foreach item ϵ L1 {              nombre d'itération au pire des cas =n
      If t contain item then       O(1)
        existe=true;               O(1)
                }
        If existe==false then      O(1)
            T=T-{t};               O(1)
                }
```

C1

Alors : C_1=m*(1+n+1)= 2m+mn=O(m*n)
(m représente le nombre de transactions et n représente les nombre des items).

```
For (k=2 ; Lk-1 !={} ; k++) {                    nombre d'itération au pire des cas=n-1
  Ck=génér_candidat (Lk-1) ;
  Foreach c ϵ Ck {                               nombre d'itération =K^(n-k)+1
    Count=0 ;                                    O(1)
    Foreach t ϵT {                               nombre d'itération =m
      If c ϵ t then count++ ;                    O(1)
                }
     If count >= minSupport then {               O(1)
       Lk=LkU{c} ;                               O(1)
        Foreach subset s != Ø of c {             nombre d'itération =2^k-1
             Confiance(s →c-s)=Support(c)/Support(s) ;                O(1)
          If confiance >=minConfiance then Règles=Règles U {s →c-s} ;  O(1)
                                    }
                                  }
          L=LULk ;                               O(1)
        Foreach t ϵ T {                          nombre d'itération =m
           existe=false;                         O(1)
          Foreach lk ϵ Lk{                       nombre d'itération =K^(n-k)+1
            Foreach item ϵ lk {                  nombre d'itération =K
              If t contain item then             O(1)
                existe=true;                     O(1)
                                  }
                          }
             If existe==false then               O(1)
                T=T-{t};                         O(1)
                }
}
}
```

C2

Alors : $C_2=\sum_{k=2..n}(k^{n-k}+1)*(1+m+(2^k-1)+mk^{n-k+1})=m\sum_{k=2..n}(k^{n-k}+1)+\sum_{k=2..n}(k^{n-k}+1)2^k+m\sum_{k=2..n}(k^{n-k}+1)k^{n-k+1}=m((1-k^n)(1-k^{-k})/(1-k)^2)+((1-k^n)(1-k^{-k})(2^k-1)/(1-k)^2)+m((1-k^n)(1-k^{-k})/(1-k)^2)^2*((n^2+n)/2)-1)=O(m*n^2)$

Donc :

$T=O(m*n^2)$.

(m représente le nombre de transactions et n représente les nombre des items).

IV. La deuxième approche :

Cette approche fait un seul parcours de la base de transactions dans l'étape de génération des itemsets fréquents, tandis que les algorithmes DHP, DIC et Apriori partitionné prennent un grand temps d'exécution en raison de le parcours itératif du base de transactions.

1. Principe :

L'algorithme de l'approche2 considéré comme un amélioration de l'Apriori, s'appuie sur un seul parcours de la base de transactions pour construire un tableau *n*2* (n présente le nombre les itemsets possibles) appelé *la table de hachage.* Cet algorithme passe par les trois étapes suivantes :

Construction de table de hachage : Le remplissage de la table de hachage se fait par un parcours de la base de transactions élément par élément. Pour chaque transaction, on extrait tous les itemsets possibles avec une fonction de hachage h(x)=h(x)+1 (On incrémente la fréquence si l'itemset existe déjà sinon on initialise la fréquence par 1).

Extraction des itemsets fréquents : A partir de la table de hachage qui est construite, on extrait les itemsets fréquents (qui ont un support >= minSup).

Génération de règles fréquentes : C'est exactement comme l'approche précédente.

Cette approche fait un parcours de la base de transactions lors de la construction de table de hachage seulement, contrairement par rapport à l'Apriori qui parcourt la base pour chaque génération d'itemset.

2. Pseudo code de l'approche 02 :

```
Entrée : minSupport, minConfiance, T
  Tab=Table_hachage(T, minSupport) ;
  Foreach itemset ϵ Tab {
    If (freq itemset)>=minSupport then
      L=L U{itemset} ;
    }
  Foreach l ϵ L /l.size>1 {
  Foreach subset s != Ø of l {
    Confiance (s→l-s)=Support(l)/Support(s) ;
    If Confiance>=minConfiance then
      Règles=Règles U{ s→l-s } ;
    }
  }
Return Règles ;
```

3. Illustration :

Soit la base de transaction suivante consiste à 5 transactions :

Etape 1 : construction de table de hachage.

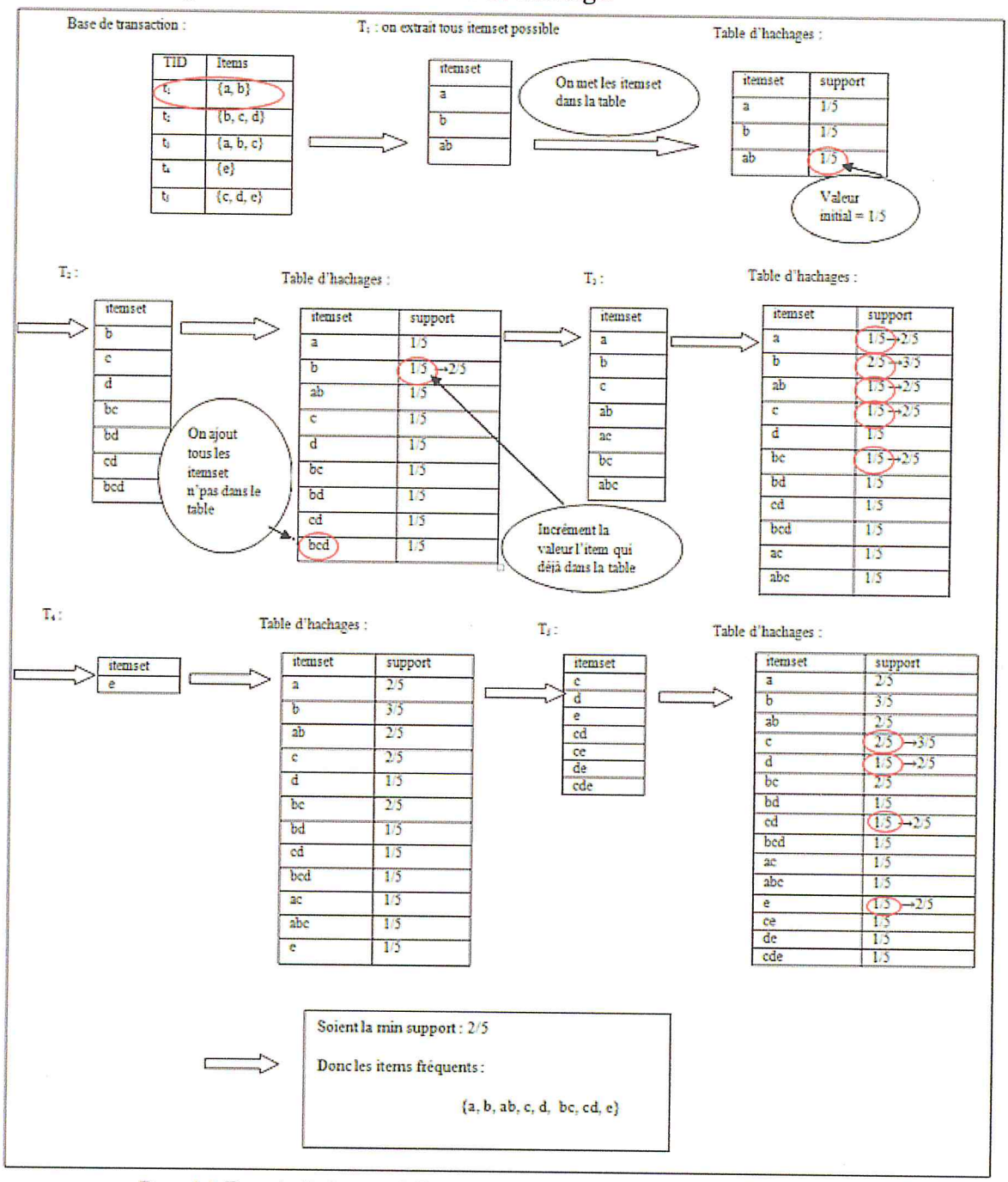


Figure 3.2 Exemple d'utilisation de l'approche 02 pour la construction de table de hachage.

- On parcourt chaque itemset dans la base de transactions on commence par le t_1, les itemset de $t_1 = \{a, b, ab\}$, on va mette chaque item dans la table de hachage avec une valeur initiale égale à 1/5.

- On parcourt le t_2 de base de transactions les itemset de t_2 = {b, c, d, bc, bd, cd, bcd} on va mettre chaque item danst_2 dans la table de hachage, l'item {b} existe déjà dans la table de hachage, on va incrément le support et les autres items {c, d, bc, bd, bcd} on va mettre1/5.
- On parcourt le t_3 de base de transactions les itemset de t_3 = {a, b, c, ab, ac, bc} on va mettre chaque item danst_3 dans la table de hachage, les items {a, b, c, ab} existent déjà dans la table de hachage ; on va incrément le support et l'item {ac, abc} on va mettre1/5.
- On parcourt le t_4 de base de transactions les itemset de t_4 = {e} onva mettre l'item dans la table de hachage ; l'item avec la valeur initiale égale 1/5.

Etape 2 : génération de règles fréquentes.

Soit minConfiance=3/5.

itemsets	les règles possibles	confiance	les règles fréquents
ab	a→b	(2/5)/(2/5)=1	a→b
	b→a	(2/5)/(3/5)=2/3	b→a
bc	b→c	(2/5)/(3/5)=2/3	b→c
	c→b	(2/5)/(3/5)=2/3	c→b
cd	c→d	(2/5)/(3/5)=2/3	c→d
	d→c	(2/5)/(2/5)=1	d→c

Table 3.1 Génération de règles fréquentes

4. La complexité :

La complexité de la deuxième approche se calcule comme suit :

```
Tab=Table_hachage(T, minSupport) ; au pire des cas = O(m*2^n )
Foreach itemset ϵ Tab {      nombre d'itération au pire des cas =m*(2^n-1)
  If (freq itemset)>=minSupport then          O(1)
    L=L U{itemset} ;                          O(1)
  }
Foreach l ϵ L /l.size>1 {    nombre d'itération au pire des cas =m*(2^n-1)-n*m
Foreach subset s != Ø of l { nombre d'itération au pire des cas = 2^n-1
  Confiance (s→l-s)=Support(l)/Support(s) ;   O(1)
  If Confiance>=minConfiance then             O(1)
    Règles=Règles U{ s→l-s } ;                O(1)
  }
}
Return Règles ;                               O(1)
```

Donc :

$T=m*2^n+m*(2^n-1)+m\times(2^n-1)^2-n*m*(2^n-1)=-2m+m2^{2n}-n*m2^n+n*m=O(m2^{2n})$.

V. Conclusion :

A partir de cette analyse théorique, nous déduisons que ces approches ne sont pas parfaitement efficaces, des fois l'algorithme d'Apriori et l'approche1 sont mieux que l'approche2 quand on a une base de transactions condensée mais si la base est non condensée, l'approche2 génère un nombre d'itemsets important alors, elle ne reste plus efficace par rapport à l'Apriori. Contrairement, l'approche1 il reste toujours mieux que l'Apriori.

Donc chaque amélioration proposée dépend des conditions, ces conditions suivent les cas les plus généraux.

Dans le chapitre suivant, nous allons essayer d'évaluer expérimentalement nos algorithmes sur la base d'une série de tests. Ces tests permettront de voir l'influence de quelques optimisations que nous introduisons, d'analyser les caractéristiques de nos algorithmes, ainsi que de comparer ses performances à l'algorithme d'Apriori.

CHAPITRE 04 :

Implémentation et expérimentation

I. Introduction :

Dans le chapitre précédent nous avons proposé deux nouveaux algorithmes qui améliorent l'Apriori pour l'extraction des règles d'association, l'implémentation s'appuiera sur ses algorithmes, l'approche1 réduira la base de transactions lors du calcul, surtout si le minimum support est plus grand par rapport aux algorithmes existants et l'approche2 est mieux que l'approche1 parce que généralement les bases de transaction ne contiennent pas beaucoup d'items.

Dans ce chapitre, nous présentons le parti de l'implémentation avec l'outil utilisé. Ensuite, nous présenterons une évaluation expérimentale appliquée sur des bases benchmark permettant d'analyser les caractéristiques des deux approches et de comparer ses performances à celle de l'algorithme Apriori. Ces derniers vont extraire les itemsets règles fréquents et comptabiliseront le temps d'exécution.

II. Implémentation :

1. Outils utilises:

NetBeans IDE 7.0.1 :

Le NetBeans IDE est un awardwinning environnement de développement intégré disponible pour Windows, Mac, Linux et Solaris. Le projet de NetBeans se compose d'un open-source IDE et d'une plate-forme d'application qui permet à des développeurs de créer rapidement le Web, entreprise, bureau, et applications mobiles utilisant la plate-forme de Java, aussi bien que PHP, JavaScript et Ajax, routinier et Graal, et C/C++.

Le projet de NetBeans est soutenu par la communauté vibrante de développeur et offre des ressources étendues de documentation et de formation aussi bien qu'une sélection diverse de tiers des logiciels [20].

2. Classes utilisées :

Dans notre application nous utilisâmes plusieurs classes (*Figure 4.1*) interactif entre eux.

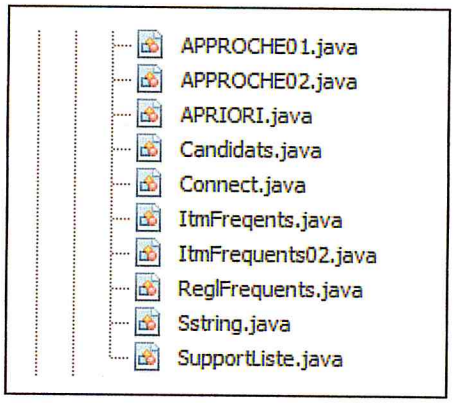


Figure 4.1 Présentation du classes utilisées.

2.1 Classe APPROCHE01 :

Cette classe présente le processus de la première approche, contient une méthode pour générer l'ensemble de 1-item candidat et une méthode pour générer les k-itemsets fréquents en utilisant la classe ItmFrequents02.

2.2 Classe APPROCHE02 :

La classe APPROCHE02 réalisant le processus de la deuxième approche, se compose de deux méthodes. La première méthode fait un parcours global de la base de transaction pour générer les itemsets possibles de chaque transaction avec leurs fréquences afin de construire la table de hachage. La deuxième méthode génère les k-itemsets fréquents en utilisant la classe ItmFreqents.

2.3 Classe APRIORI :

C'est une classe permettant d'implémenter le processus d'extraction des itemsets fréquents de l'algorithme Apriori.

2.4 Classe Candidats :

C'est une classe importante pour Apriori et Approche01. Elle permet de générer les k-itemsets candidats en s'appuyant sur (k-1)-itemsets fréquents.

2.5 Classe ItmFreqents :

Cette classe implémente le processus d'extraction des itemsets fréquents. Elle a besoin comme paramètres d'entrer la base de transaction, le minimum support et l'ensemble k-itemset candidat qui se parcourt afin de connaître les itemsets candidats qui ont un support supérieur ou égal au support minimal.

2.6 Classe ItmFrequents02 :

De même fonctionnalité de la classe précédente, cette classe permet d'extraire les itemsets fréquents sauf qu'elle élimine les itemsets qui ont un support inférieur au support minimal, celui qui permet de réduire la base de transaction pour le prochain parcours.

2.7 Classe ReglFrequents :

La classe ReglFrequents implémente le processus d'extraction des règles fréquents. Elle génère toutes les règles de taille 1 en partie *cause* et (n-1) en partie *conséquence* de chaque itemset fréquent. En outre, elle génère le côté le plus à gauche à partir de la $(n-3)^{ième}$ profondeur de la Treillis des itemsets fréquents ont une taille supérieure à trois dans la partie cause de la règle et met le reste en partie conséquence afin d'extraire les règles qui ont un support supérieur ou égal à la confiance minimale.

Elle a besoin d'entrer dans la base de transactions, tous les k-itemsets $_{(k\epsilon 1..n)}$ fréquents et la confiance.

2.8 Classe Sstring :

L'intérêt de cette classe est de convertir une liste à une chaîne de caractère ne contenant pas des '[', des ']' et des ','. Elle s'utilise par la classe ReglFrequents.

2.9 Classe SupportListe :

Elle est la classe la plus importante. Elle permet de calculer l'occurrence de l'itemset dans la base de transactions.

3. Lancement de l'application :

Accueil :

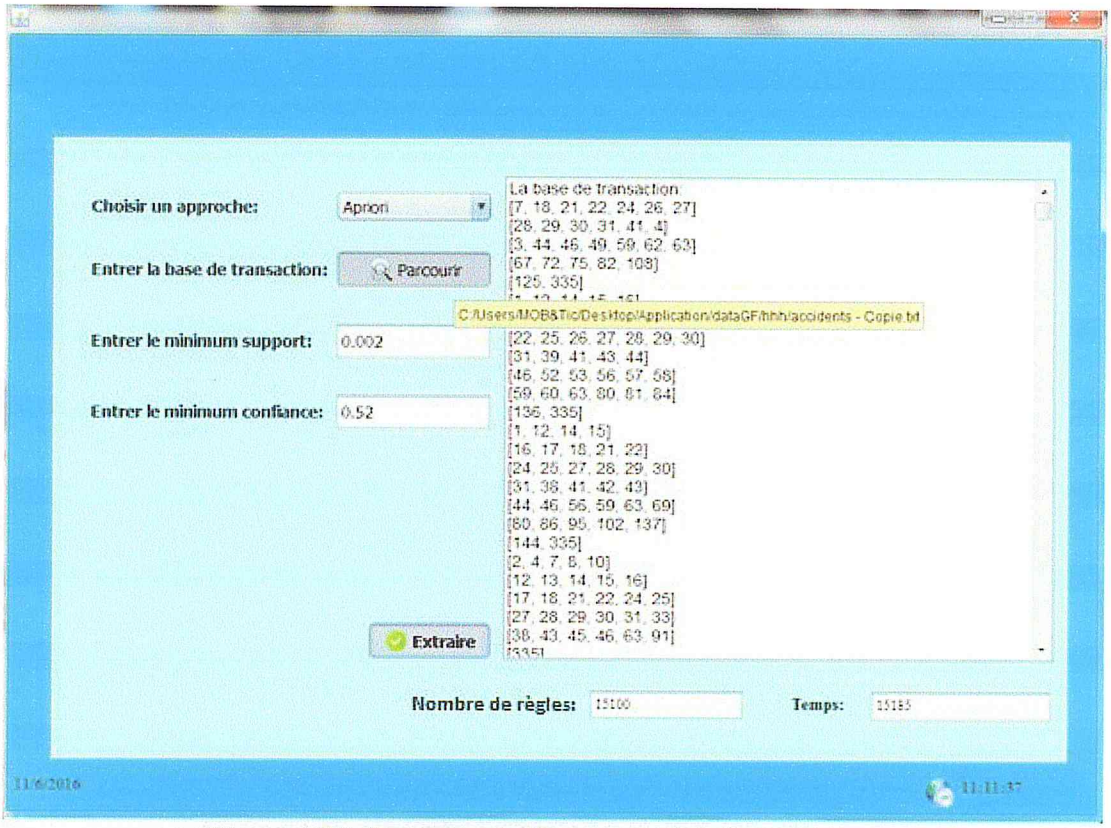


Figure 4.2 Présentation de l'accueil de l'application.

La commande Extraire :

Quand on choisit un approche, préciser la base de transactions et remplir les cases de support et confiance, on clique sur « Extraire » pour lancer le processus d'extraction des règles d'association.

III. Résultat d'expérimentation :

1. Spécificités des Bases de Transactions utilisées pour les tests:

Dans la partie expérimentations, nous allons calculer le temps d'exécution total et le nombre des règles d'association pour les tests (Apriori, approche1 et approche2) qui nous avons appliqué sur des bases benchmark [21].

Les caractéristiques de ces bases sont résumées dans un tableau défini pour chaque base : nom, type, le nombre de transactions, le nombre d'items et la taille (*Table 4.1*).

Nom de la base	Type de la base	Nombre de transaction	Nombre d'items	Taille moyenne
T10I4D100K	Eparse	100 000	1 000	3928 ko
T40I1D100K	Eparse	100 000	1 000	15116 ko
PUMSB	Dense	49046	7 117	16299 ko
CONNECT	Dense	67 557	129	9039 ko
CHESS	Dense	3 196	75	335 ko
MUSHROOM	Dense	8 124	119	558 ko
ACCIDENTS	Eparse	340 183	468	2131 ko
RETAIL	Eparse	88 162	16 470	4070 ko
KOSARAK	Eparse	990 002	41935	379 ko

Table 4.1. Caractéristique des bases utilisées [27].

1.1.Les définitions des bases utilisées :

La base PUMSB :

Cette base contient des données de recensement (c'est-à-dire des données statistiques).

La base MUSHROOM :

Cette base contient les caractéristiques de différentes espèces de champignons.

Les bases CONNECT et CHESS :

Sont dérivées à partir des étapes de jeux respectifs qu'elles représentent.

Ces bases sont fournies par U C Irvine MachineLearning Database Repository [22]. Ces bases sont très denses (c'est-à-dire, elles produisent plusieurs itemsets fréquents longs même pour des valeurs de supports élevées).

T10I4D100K et T40I10D100K :

Sont deux bases synthétiques générées par un programme développé dans le cadre du projet dbQUEST [23].

Ces bases simulent le comportement d'achats des clients dans des grandes surfaces [24].

La base RETAIL:

Elle est fournie par une grande surface anonyme située en Belgique [25].

La base ACCIDENTS:

Elle est obtenue de l'institut national belge de statistiques et représente des statistiques sur les accidents du trafic survenus dans la région de Flandre (Belgique) entre 1991 et 2000 [26].

La base PUMSB :

Cette base contient des données de recensement (c'est-à-dire des données statistiques).

2. Discussions :

L'ensemble des expérimentations que nous avons faites ont été effectuées sur un PC d'un processeur Intel CORE i3 sous Windows 2.20GHz et 2.00Go de RAM.

A partir des moyens existants, nous utilisons une partie de bases originales afin d'obtenir les résultats suivants :

Data	Support	Confiance	Apriori		Approche 1		Approche 2	
			taille	temps	Taille	temps	taille	Temps
P_Accidents	0,5%	1%	41882	37005ms	41882	36017ms	28192	19435 ms
		45%	20243	18726ms	20243	18081ms	13991	11958 ms
		9%	36423	33074ms	36423	29212ms	24923	18491 ms
	0,7%	10%	35239	28376ms	35239	27940ms	24097	17884 ms
		60%	14964	16422ms	14964	15458ms	10435	10782 ms
		8%	36945	32820ms	36945	30039ms	25278	19893 ms
	0,2%	8%	36945	33316ms	36945	32602ms	25278	20279 ms
		52%	15100	15282ms	15100	15282ms	10568	10525 ms
		30%	24537	19882ms	24537	19683ms	16968	12546 ms
P_Kosarak	1,3%	90%	4021	850ms	4021	810ms	13	158528 3ms
		4,5%	6146	1090ms	6146	1000ms	78	199058 5 ms
		50%	4978	1003ms	4978	980ms	36	162641 2ms
	5%	4%	166	30ms	166	30ms	12	157356 9ms
		0,05%	166	40ms	166	30ms	12	149959 5ms
		10%	154	30ms	154	30ms	12	157733 8ms
	2,5%	8%	5664	1160ms	5664	1075ms	68	177976 8ms
		10%	5514	1030ms	5514	1020ms	65	149797 8ms
		0,7%	6146	1160ms	6146	1150ms	78	182687 8ms

P_T10I4D100K	2,5%	40%	264	3354 ms	264	3261ms	23	231785 ms
		10%	282	4025	282	3276 ms	24	246231 ms
		90%	186	4165ms	186	3322 ms	24	246231 ms
	4%	1%	282	4056 ms	282	3447ms	10	239632 ms
		40%	264	4150ms	264	3276ms	23	260692 ms
		0,05%	282	4196ms	282	3339ms	23	215827 ms
	2,5%	40%	264	3354 ms	264	3261ms	23	231785 ms
		10%	282	4025	282	3276 ms	24	246231 ms
		90%	186	4165ms	186	3322 ms	24	246231 ms
P_Retail	2,5%	1,5%	369	203ms	369	109 ms	178	197965 ms
		40%	182	141 ms	182	125 ms	79	192863 ms
	4%	90%	112	109 ms	112	109 ms	37	191882 ms
		2%	92	16ms	92	31 ms	48	192505 ms
	1,99%	30%	56	31ms	56	16ms	33	205499 ms
		0,075%	92	31ms	92	15ms	48	194876 ms

Table 4.2 Résultats d'exécution des BDT.

Figure 4.3 Résultat d'execution de P_Accidents.

Nous remarquons que l'approche1 et l'Apriori sont presque identiques en utilisant un support inférieur à 0.7%, mais par rapport à l'approche2, elle a un temps d'exécution moindre de l'approche1 et Apriori. Alors, dans ce cas l'approche2 est la meilleure.

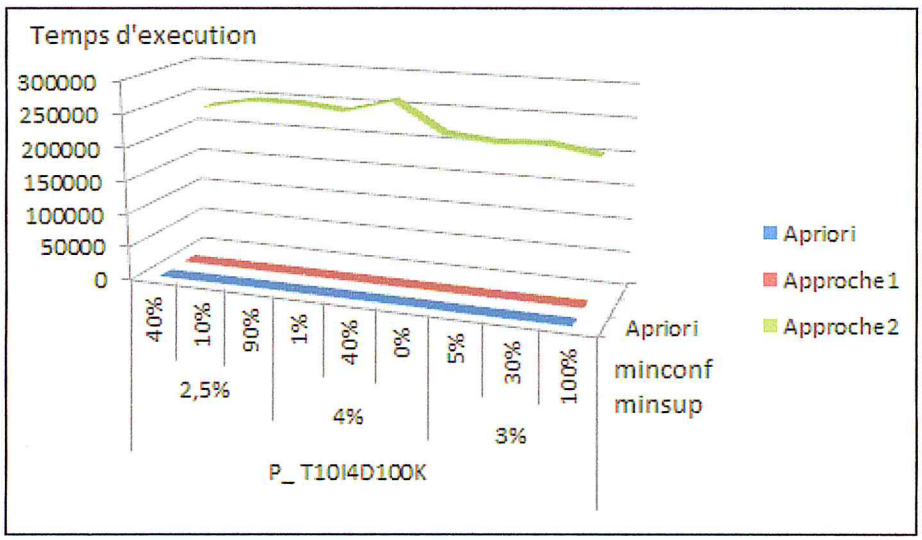


Figure 4.4 Résultat d'execution de P_T10I4D100K.

Par rapport à cette base, l'approche1 a un temps d'exécution presque égal au temps d'exécution de l'Apriori en utilisant un minSup inférieur à 4% mais l'approche2 a un temps considérable par rapport à Apriori et approche1.

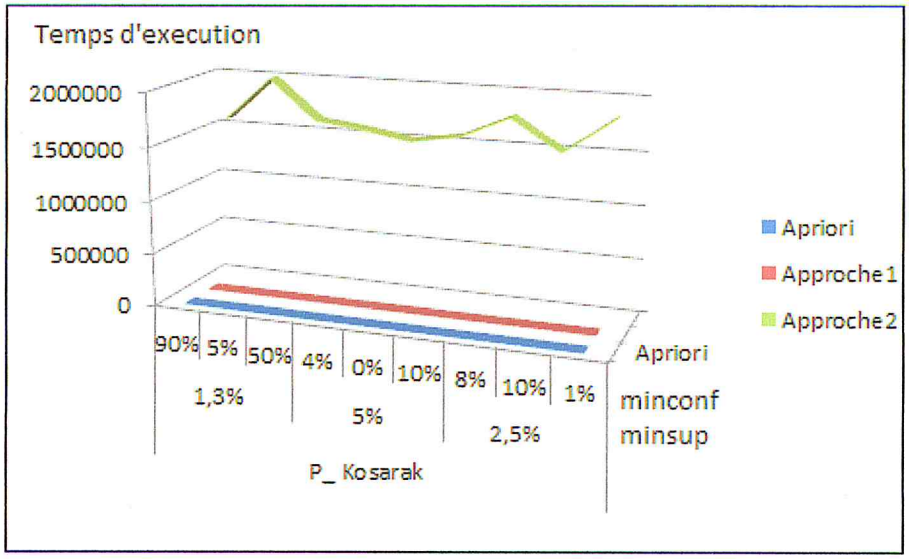


Figure 4.5 Résultat d'execution de P_kosarak.

Nous remarquons que l'approche1 et l'Apriori sont presque identiques en utilisant un support inférieur à 5%, et mieux que l'approche2 en temps d'exécution.

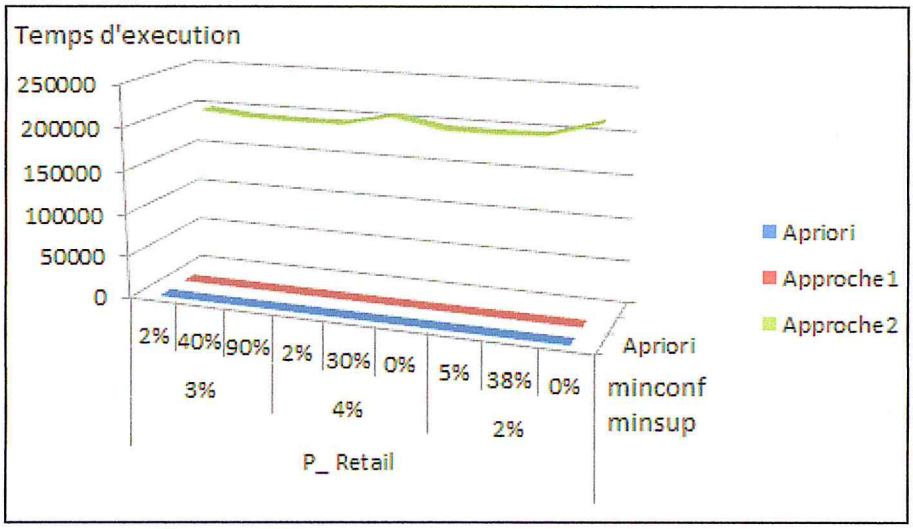


Figure 4.6 Résultat d'execution de P_Retail.

Aussi par rapport à la base Retail, l'approche1 identique avec l'Apriori et l'approche2 a le plus grand temps d'exécution

A partir de la table 4.2 et les graphes précédents nous déduisons que l'approche1 est proportionnellement mieux que l'Apriori mais l'approche2 n'est efficace qu'avec les base de transactions qui ont des transactions de petite taille.

IV. Conclusion :

Dans ce chapitre, nous avons présenté une étude expérimentale sur l'algorithme Apriori et les deux approches proposées, pour l'extraction des règles d'association. Les expérimentations montrent une amélioration remarquable en termes de temps d'exécution par rapport à l'algorithme Apriori. Nous remarquons aussi que la première approche est complète et optimale en trouvant toutes les règles obtenues par l'algorithme Apriori, alors que la deuxième approche ne trouve qu'une partie des règles d'association. Comme futur travail, nous allons essayer d'améliorer la deuxième approche afin de vérifier sa complétude.

Conclusion générale

La fouille de données est un domaine qui est conclu à la suite d'un problème qui se présente aux entreprises. Ce problème consiste dans l'analyse de grosses masses de données stockées dans leurs bases de données. La fouille de données englobe plusieurs techniques, dont l'extraction des règles d'association. L'extraction des règles d'association est un processus qui permet de déduire un ensemble de règles reliant les différentes données d'une base de transactions.

Le problème d'extraction des règles d'association propose plusieurs variantes séquentielles et parallèles pour les algorithmes séquentielles exact il y deux types des algorithmes : les algorithmes d'extraction des règles d'association basés sur générer et tester (apriori, DHP, DIC, Apriori partitionné,….) telle que Apriori est le base de la majorité des algorithmes servant à découvrir des règles d'association, et les algorithmes d'extraction des règles d'association basés sur diviser pour régner (FP-growth, FI-growth).

L'algorithme d'Apriori est très lourd, car il parcourt la base de transactions plusieurs fois c'est pour cette raison que nous avons introduit deux nouvelles approches permettant d'améliorer l'algorithme Apriori. La première approche permette d'organiser l'espace de la base de transaction, alors que dans l'étape de génération des itemsets fréquents elle détermine l'ensemble de k-itemset fréquent de manière récursive à partir de (k-1)-itemset fréquent à la fin on élimine toutes les transactions qui ne contiennent aucun itemset fréquent et tout sous-itemset d'un itemset fréquent doit être fréquent. Pour construire l'ensemble de k-itemset candidat on applique le jointeur sur (k-1)-itemset, ensuite on doit calculer le support de chaque itemset et éliminer les itemsets qui ont un support inferieur au minSup. Et la deuxième approche base sur le parcours de base de transaction une seule fois afin de construire un tableau *n*2* (n présente le nombre les itemsets possibles) appelé la table de hachage, le remplissage de ce table se fait par un parcours de la base de transactions transaction par transaction, pour chaque transaction on extrait tous les itemsets possibles avec une fonction de hachage h(x)=h(x)+1 (On incrémente la fréquence si l'itemset existe déjà sinon on initialise la fréquence par 1). Avec la contribution de ces approches, nous déduisons que théoriquement l'approche1 est plus efficace que l'Apriori lorsque le minimum support est grande et l'approche2 est mieux que l'approche1 lorsqu'on des bases de transactions non condensées (comprend un nombre considérable de transactions de petite taille).

Nous avons fait les tests sur les bases benchmark standard qui sont des bases (éparses ou bien denses) et après les tests, nous trouvons le même résultat trouvés dans la contribution pour l'approche1, telle que l'approche 1 est mieux que l'Apriori quand la valeur de minimum support est grand quelle que soit la valeur du minimum confiance, et l'approche2 reste toujours mieux lorsque on a des base de transaction non condense.

A la fin, ces approches restent ouvertes à l'amélioration comme un futur travail.

Bibliographie

[1] J.-F. Boulicault an d B. Crémilleux. Editorial du volume. In J.-F. Boulicault an d B. Cremilleux, editors, Revue d'Ingénierie des Systêmes d'Information (ISI), Hermês-Lavoisier, volume 9, pages 722, 2004.

[2] R. Agrawal, T. Imielinski, an d A. Swami. Mining association rules between sets of items in large databases. In Proceedings of the ACM-SIGMOD Intl. Conference on Management of Data, Washington D. C., USA, pages 207216, May 1993.

[3] R. Agrawal an d R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, an d C. Zaniolo, editors, Proceedings of the 20th Intl. Conference on Very Large Databases, Santiago, Chile, pages 478-499, June 1994.

[4] M. J. A. Berry an d G. S. Linoff. Data Mining Techniques : For Marketing, Sales, and Customer Relationship Management, Second Edition. Wiley Publishing, 2004

[5] Fareh, M. «Data Mining, Le Data Mining», 2015, P 10.

[6] DJEFFAL,Abdelhamid. «Data Mining avance, Introduction », 2015, P10.

[7] Lenca, P., Meyer, P., Vaillant, B., &Lallich, S. (2008). On selectinginterestingnessmeasures for association rules : User oriented description and multiple criteriadecisionaid. European Journal of OperationalResearch, 184(2), 610-626.

[8] Hegland, M. (2005). The apriorialgorithma tutorial.Mathematics and computation in imaging science and information processing, 11, 209-262.

[9] Fang, Wenbin, et al. "Frequentitemsetmining on graphics processors." Proceedings of the fth international workshop on data management on new hardware.ACM, 2009.

[10] Agrawal, Rakesh, TomaszImielinski, and Arun Swami. "Mining association rulesbetween sets of items in large databases." ACM SIGMOD Record. Vol. 22. No. 2. ACM, 1993.

[11]Amara, Mohammed et Benouaz, Omar Farouk.Techniques Data Mining pour la sélection d'une configuration d'index de jointure binaire, Telemcan, Université Abou BekrBelkaid, 2011, 72.

[12] Park, Jong Soo, Ming-Syan Chen, and Philip S. Yu. An eective hash-basedalgorithm for mining association rules.Vol. 24.No. 2.ACM, 1995.

[13] Djenouri, Youcef. Fouille de règles d'association sur GPU (Thèse de Doctorat soutenu à L'USTHB), USTHB, 2015, 171.

[14] Brin, Sergey, et al. "Dynamicitemsetcounting and implication rules for market basket data." ACM SIGMOD Record.Vol. 26.No. 2.ACM, 1997.

[15] Savasere, A., Omiecinski, E. R., &Navathe, S. B. (1995). An e-cientalgorithm for mining association rules in large databases.

[16] Mueller, A. (1998). Fastsequential and parallelalgorithms for association rulemining : A comparison.

[17] Zaki, Mohammed Javeed, SrinivasanParthasarathy, and Wei Li. "A localizedalgorithm for parallel association mining." Proceedings of the ninthannual ACM symposium on Parallelalgorithms and architectures.ACM, 1997.

[18] Han, Jiawei, Jian Pei, and Yiwen Yin. "Miningfrequent patterns without candidate generation." ACM SIGMOD Record.Vol. 29.No. 2.ACM, 2000.

[19] Amphawan, K., and A. Surarerks. "An Approach of Frequent Item Tree for Association Generation."Articial Intelligence and Soft ComputingNinth IASTED International ConferenceProceedings. 2005.

[20] ORACL.24/05/2016.NetBeans IDE 7.0.1 Release Information, [En ligne].Adresse URL: https ://netbeans.org/community.

[21]FIMI.Reposity, http :// fimi.cs.helsinki.fi/data.

[22]UCI.24/05/2016.Machine learning Repository [En ligne].Adresse

http://www.ics.uci.edu/~mlearn/ML Repository.html.

[23] IBMResearch.24/05/2016.IBM , [En ligne].Adresse URL: http://www.almaden.ibm.com/software/quest/ Resources/datasets/data/.

[24] J. Hipp, U. Giintzer, an d G. Nakhaeizadeh. Algorithms for association rulemining - a generalsurvey an d comparison. In ACM-SIGKDD Explorations, New York, USA, volume 2, pages 5864, July 2000.

[25] T. Brijs, G. Swinnen, K. Vanhoof, an d G. Wets. Using association rules for productassortmentdecisions : A case study. In Proceedings of the sixth ACM-SIGKDD InternationalConference on KnowledgeDiscovery and Data Mining, San Diego, California, USA, pages254260, August 1518 1999.

[26] K. Geurts. Traffic accidents data set. In B. Goethals an d M. J. Zaki, editors, Proceedings of the IEEE ICDM Workshop on Frequent Itemset MiningImplementations (FIMI'03), volume 90 of CEUR Workshop Proceedings, Melbourne, Florida, USA, November 19, 2003.

[27] Benelhadj,Mohamed El Hadi. Entrepôt de Données et Fouille de Données Un Modèle Binaire et Arborescent dans le Processus de Génération des Règles d'Association (thèse de Doctorat soutenu à UNIVERSITE MENTOURI CONSTANTINE) UNIVERSITE MENTOURI CONSTANTINE, 2012, P 96.