الجمـــهوريـــة الجـــزائريـــة الديـمقراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT
كـلـيـــــة الـعـلـــــوم
FACULTE DES SCIENCES
قسم الرياضيات و الاعلام الالي
DEPARTEMENT DE MATHEMATIQUE ET INFORMATIQUE

Mémoire De Master

Domaine : Mathématiques et Informatique(**MI**)

Filière : Informatique

Option : Système d'information et aide à la décision(SID)

Présentée et soutenue publiquement par

SEGHIER Kaouther

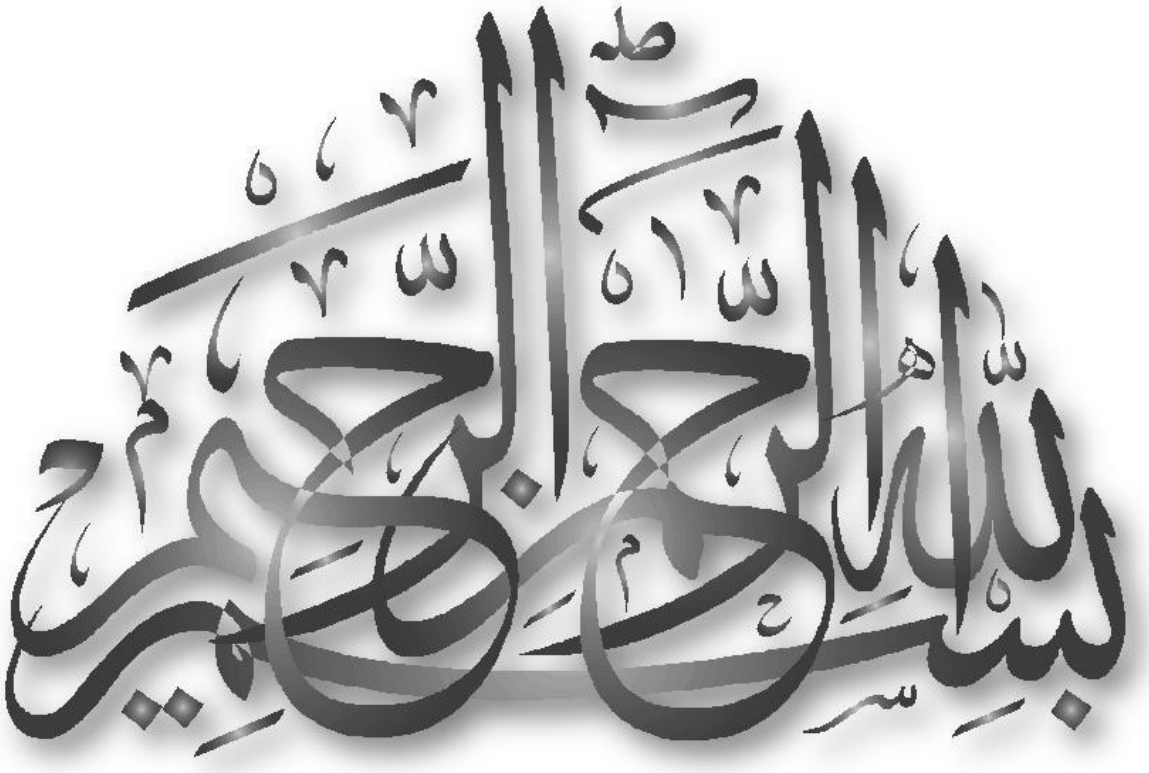
Thème

Recherche des motifs fréquents fermés : l'implémentation de l'algorithme close

Soutenu devant le jury composé de :

Mr .Laaradj Chellama	**(Examinateur/Président)**	**Université De Laghouat**
Mr .Bouakkaz Mustapha	**(Examinateur)**	**Université De Laghouat**
Mr .Benameur Ziani	**(Encadreur)**	**Université De Laghouat**

2014/2015


بسم الله الرحمن الرحيم

Dédicace

Je dédie ce travail :

A mes très chers parents qui ont toujours été là pour moi, «Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous m'avez donné un magnifique modèle de labeur et de persévérance. Je suis redevable d'une éducation dont je suis fier».

A mon cher marie qui m'aide à sauter beaucoup de difficulté et problème de santé et réussir de finir ce travaille.

A mon cher enfant, mes frères, mes sœurs et mes amis.

Remerciements

Je tiens à exprimer toute ma reconnaissance à mon directeur de mémoire monsieur

BENAMEUR ZIANI. Je le remercie de m'avoir encadré, orienté, aidé et conseillé.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté à me rencontrer et répondre à mes questions durant mes recherches.

Résumé :

Pour faire face aux nouveaux enjeux, les entreprises doivent collecter, traiter, et analyser les informations de leurs environnement afin de résoudre et anticiper les problèmes, les données collectées doivent être valorisés afin d'extraire des connaissances de la fouille de donnée.

La recherche des motifs fréquents est une tâche importante de la fouille de données. Cependant le nombre de motif générées est exponentiel, pour n items, 2^n motifs sont potentiellement testés et générés. Les représentations condensées des motifs fréquents visent à remédier à ce problème.

Dans ce travaille on s'intéresse à une des représentations condensés les plus connu a savoir les motifs fréquents fermés. Ce travail réalisé consiste a analysés et implémentés l'algorithme close pour la recherche des motifs fréquents fermés.

Mots clé : Fouille de données, motifs fréquents fermés, Règles d'association, Algorithme close.

Abstract:

To cope with the new challenges, companies must collect process and analyze information from their environment to solve and anticipate problems; the data collected must be recycled in order to extract knowledge using data mining techniques.

Discovering frequent patterns is an important task of data mining. However, the number of pattern generated is exponential, for n items, 2^n patterns are tested and potentially generated. The condensed representations of frequent patterns are designed to remedy this problem.

In this work we are interested of one of a condensed representation we analyze and implement the close algorithm for extracting closed frequent itemsets.

Keywords: data mining, Close algorithm, association rules, frequent closed items.

تلخيص

لمواجهة التحديات الجديدة، يتعين على الشركات جمع ومعالجة وتحليل المعلومات من بيئتهم لحل و استباق المشاكل, هذه البيانات التي تم جمعها يجب اعادة تقييمها من أجل انتزاع المعلومات من قاعدة بيانات البحث.

البحث عن أنماط متكررة مهمة هامة للتعدين البيانات. ومع ذلك، فإن عددالانماط هوأسي، من اجل ن بند، يتم اختبارو توليد 2^{n} نمط . اذا التمثيل المكثف للانماط المتكررة قد صممت لعلاج هذه المشكلة.

في هذا العمل نحن مهتمون بواحدة من التمثيل المكثف و الاكثر شيوعا و هي البحث عن الانماط المتكررة المغلقة. و هذا التطبيق يعمل على تحليل و تنفيذ كلوز للبحث عن الانماط المتكررة المغلقة.

كلمات البحث, استخراج البيانات, معادلة العلاقات, خوارزمية كلوز, انماط متكررة مغلقة

Table des matières :

Table des figures :

Liste des tableaux :

Table des codes sources :

Introduction Générale

Introduction

La collection de données pour faire l'extraction de connaissances qu'aide à la décision produit une masse de données. Cette dernière passe par une série d'analyses et traitements pour l'utilisation de différentes techniques de la fouille de données, telle que la classification, le clustering, l'extraction des règles d'association et la recherche de motif fréquent.

L'application de technique de recherche de motif fréquent produit un grand nombre de motifs ou patterns dont ils contiennent des motifs non importante et des fois on utilise pas durant l'analyse, ce problème introduit à réfléchi à une autre recherche de motif fréquent avec le concept d'avoir un nombre minimum et important des motifs pour l'analyse donc ils ont inventé le domaine de représentation condensé qui assure ces conditions avec beaucoup d'avantages qui n'existe pas dans la recherche de motif fréquent comme le temps d'exécution la complexité…etc.

Dans notre projet on intéresse à la recherche de motif fréquent fermé du domaine de représentation condensé afin d'extraire des motifs fréquents minimum et importante,

Le présent travail se focalise principalement sur un algorithme d'extraction des motifs fréquents fermés : *close,* étant donnés une base de transaction et un support minimum connu.

L'objectif principal de ce travail est de réalisé une implémentation de l'algorithme *close* en langage java.

Organisation de mémoire:

Après une introduction générale, ce mémoire est divisé en quatre chapitres.

- le premier chapitre présent des généralités sur la fouille de données, qui explique les différentes techniques, et les concepts principales de la fouille de donnée (un motif, un motif fréquent, une transaction, une base de transaction, un support…etc.) ;
- le deuxième chapitre défini la technique de recherche de motif fréquent.
- le troisième chapitre décrit l'algorithme Close et les définitions (les motifs fréquents fermés, la fermeture…etc.) ;
- le quatrième chapitre est consacré à une étude expérimentale de l'implémentation réalisée ;

- Une conclusion et des perspectives sont présentes en fin du mémoire ;

Chapitre I

Généralité sur La fouille de données

1. Le processus d'extraction des connaissances ECD :

L'ECD est un procédé non trivial d'extraction des connaissances à partir des données, l'ECD est un processus itératif qui met en œuvre un ensemble de techniques provenant des bases de données, de la statistique, de l'intelligence artificielle, de l'analyse des données, des interfaces de communication homme-machine. L'ECD vise à transformer des données (volumineuses, multiformes, stockées sous différents formats sur des supports pouvant être distribués) en connaissances. Ces connaissances peuvent s'exprimer sous forme d'un concept général qui enrichit le champ sémantique de l'usager par rapport à une question qui le préoccupe. Elles peuvent prendre la forme d'un rapport ou d'un graphique. Elles peuvent s'exprimer comme un modèle mathématique ou logique pour la prise de décision. Les modèles explicites, quelle que soit leur forme, peuvent alimenter un système à base de connaissances ou un système expert.

Ce processus est complexe et se déroule suivant une série d'opérations. Des étapes de pré-traitement ou préparation sont nécessaires avant la fouille de donnée en tant que telle. Le prétraitement concerne la mise en forme des données entrées selon leur type (numériques, symboliques, images, textes, sons), ainsi que le nettoyage des données, le traitement des données manquantes, la sélection d'attributs ou la sélection d'instances. Cette première phase est cruciale car du choix des descripteurs et de la connaissance précise de la population va dépendre la mise au point des modèles de prédiction. L'information nécessaire à la construction d'un bon modèle de prévision peut être disponible dans les données. [1]

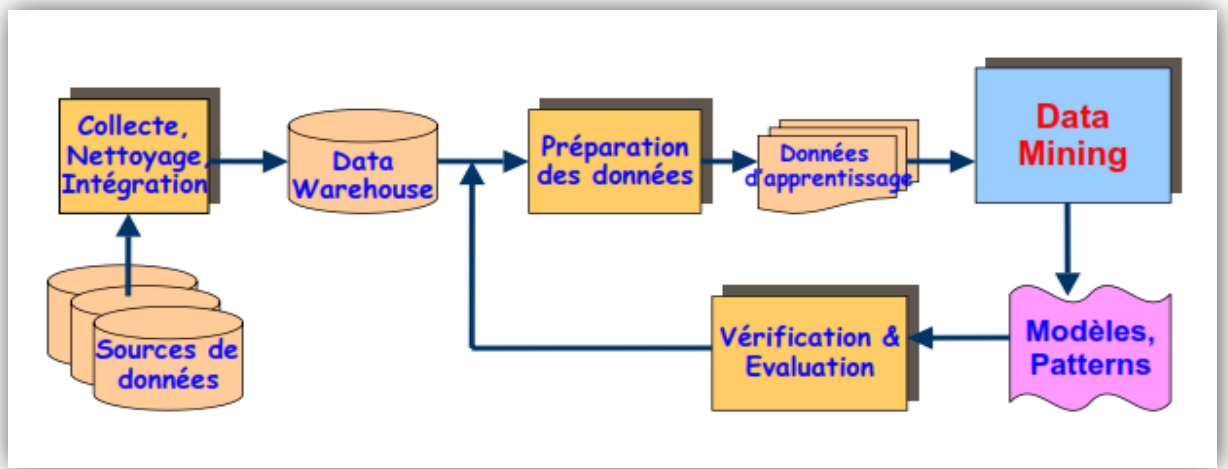


Figure 1.1 : schéma de processus d'extraction des connaissances [1]

1.1. Des données aux connaissances :

- **Donnée :**

Une donnée est une valeur des champs des enregistrements des Tables de base de données qui concerne un domaine et comme est une collection d'objets et leurs attributs ou caractéristiques et n'aide pas à la décision.

Exemple d'une donnée : réseau, transport, santé…etc**.**

- **Information :**

Une information est une collection de donnée qui peut être composé de deux ou trois mots au minimum et lorsqu'on entendre une information on peut compris quelque idées mais on ne peut pas décider.

Exemple d'une information : client 1 visite le site S, l'homme M traverse la route R.

- **Connaissance :**

Une connaissance est une collection de valeurs plus significative que donnée et information et est comme un résultat d'étude faite sur un groupe de nature et qui donne un sens pour l'aide à la décision.

*des conditions sur les connaissances pour être accepter dans le domaine de fouille de donnée est :

Nouvelles, valides, potentiellement utiles et compréhensibles.

Nouvelles : non prévisibles.

Valides : valables dans le futur.

Utiles : permettent à l'utilisateur de prendre des décisions.

Compréhensibles : présentation simple.

Les connaissances permettent de prendre des décisions **:**

Une décision est généralement des consignes sont prendre afin d'améliorer les situations de sujet étudier.

- Exemple d'une décision : pour augmenter le taux des visiteurs ou utilisateurs de site d'université de laghouat on va ajouter au site les cours et les séries de TD en ligne.

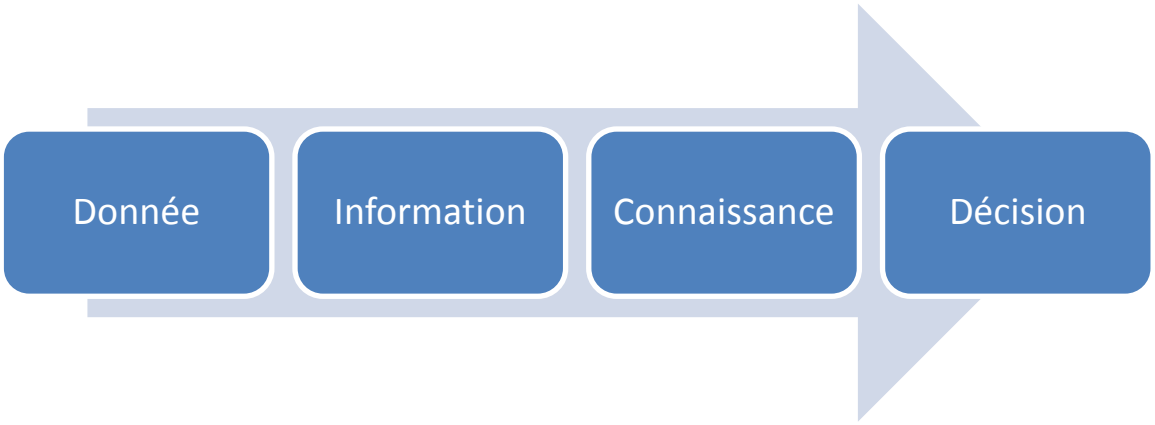


Figure 1.2 : le processus ECD de donnée à connaissance

2. La fouille de donnée (data mining) :

2.1. Définition :

Est le cœur de processus d'extraction des connaissances et est un processus qui fait intervenir des méthodes et des outils issus de différents domaines de l'informatique, de la statistique ou de
L'intelligence artificielle en vue de découvrir des connaissances utiles.

La fouille de données est un procédé d'exploration et d'analyse de grands volumes de données en vue d'une part de les rendre plus compréhensibles et d'autre part de découvrir des corrélations significatives, comme des règles de classement et de prédiction dont la finalité ultime la plus courante est l'aide à la décision.[2]

2.2. Démarche et processus de la fouille de données :

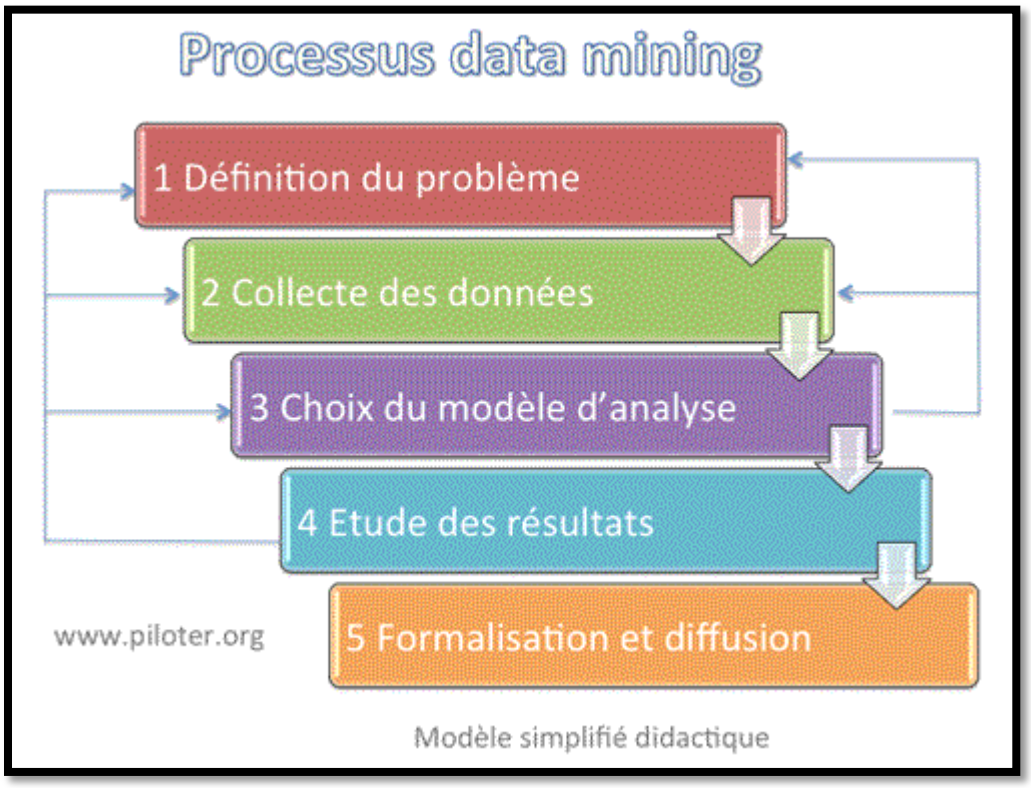


Figure 1.3 : processus de la fouille de données [4]

- **Définition du problème**

Quel est le but de l'analyse, Quels sont les objectifs ? Comment traduire le problème en une question pouvant servir de sujet d'enquête pour cet outil d'analyse bien spécifique ? A ce sujet, se souvenir que l'on travaille à partir des données existantes, la question doit être ciblée selon les données disponibles.

- **Collecte des données**

Une phase absolument essentielle. On n'analyse que des données utilisables, propres et consolidées. On n'hésitera pas à extraire de l'analyse les données de qualité douteuse. Bien souvent, les données méritent d'être retravaillées. S'assurer au final que la quantité de données soit suffisante pour éviter de fausser les résultats. Cette phase de collecte nécessite le plus grand soin.

- **Construire le modèle d'analyse**

Ne pas hésiter à valider notre choix d'analyse sur plusieurs jeux d'essais en variant les échantillons. Une première évaluation peut nous conduire à reprendre les points 1 ou 2.

- **Etude des résultats**

Il est temps d'exploiter les résultats. Pour affiner l'analyse on n'hésitera pas à reprendre les points 1, 2 ou 3 si les résultats s'avéraient insatisfaisants.

- **Formalisation et diffusion**

  Les résultats sont formalisés pour être diffusé. Ils ne seront utiles qu'une fois devenus une connaissance partagée. C'est bien là l'aboutissement de la démarche. C'est aussi là que réside la difficulté d'interprétation et de généralisation. [4]

Ces dernières années ont vu le développement des techniques de fouille de données dans de nombreux domaines d'applications dans le but d'analyser des données temporelles, volumineuses et complexes parmi les domaines d'application :

- Domaine bancaire.
- Domaine de la grande distribution.
- Domaine des télécommunications.
- Domaines de l'assurance et de la pharmacie.
- Domaine de la santé. [2]

2.3. Les techniques de la fouille de données :

2.3.1. Le problème de recherche des motifs fréquents :

Sera détaillé dans le chapitre suivant.

2.3.2. L'extraction des règles d'association :

- **Principe** :

Le principe d'extraction des règles d'association est d'après un ensemble de transactions donné, trouver des règles qui permettront de prédire un item sur la base des occurrences des autres items dans la transaction : il s'agit de mettre en relation des données, dont la confiance et le support sont supérieurs à des seuils donnés.

Mais on ne chercher les règles d'association que dans les ensembles fréquents, c'est-à-dire dont le support est supérieur au seuil fixé [2].

2.3.3. Classification :

- **Définition : ' classes connues'**

Étant donné un ensemble d'individus (appelé ensemble d'apprentissage) répartis en k groupes (appelés classes), il s'agit de définir un modèle (fonction des valeurs d'attributs des individus de **l'ensemble d'apprentissage**) qui permet d'attribuer chaque nouvel individu à l'une des k classes.

Un ensemble d'individus dont la répartition dans les classes est connue sert à tester le modèle. Il est appelé **ensemble test**.

Avec L'œil et le cerveau humain son extrêmement efficace dans les taches de classification [2].

- **Démarche du processus de classification :**

Etape 1 :

-

Construction du modèle à partir de l'ensemble d'apprentissage.

Etape 2 :

Utilisation du modèle: tester la précision du modèle et l'utiliser dans la classification de nouvelles données [2].

- **Les méthodes de classification :**

  Les méthodes utilisées dans la classification sont :

  - Méthode K-NN (plus proche voisin).
  - Arbres de décision.
  - Réseaux de neurones.
  - Classification bayésienne [3].

2.3.4 Clustering :

- **Définition :**

Le problème de clustering ou classification non supervisée (ou les classes sont non connu) est un problème plus difficile que la classification supervisée.

Soient N instances de données à K attributs, alors le partitionnement en C clusters ou groupes ayant un sens de similitude, dont l'objectif est de Minimiser les distances intra-cluster et Maximiser les distances inter-clusters [2].

- **Caractéristiques :**

- Extensibilité.
- Abilité à traiter différents types de données.
- Découverte de clusters de différentes formes.
- Connaissances requises (paramètres de l'algorithme).
- Abilité à traiter les données bruitées et isolées. [2]

Chapitre II

Problème de recherche des motifs fréquents

1. Introduction :

La recherche de motif fréquent est un processus itératif et interactif constitué de plusieurs phases allant de la sélection et la préparation des données jusqu'à l'interprétation des résultats, en passant par la phase de recherche des connaissances [2], La plupart des approches proposées pour l'extraction des motifs fréquents reposent sur les phases suivantes [5]:

Préparation des données : Cette phase consiste à sélectionner les données (attributs et objets) de la base de données utiles à l'extraction des motifs fréquents et transformer ces données en un contexte d'extraction. Ce contexte, ou jeu de données, est un triplet B= (O, A, R) dans lequel O est un ensemble d'objets, A est un ensemble d'attributs, également appelés items, et R est une relation binaire entre O et A. Un contexte d'extraction des motifs fréquents D constitué de six objets, chacun identifié par son OID, et cinq items est représenté dans la Table 2.1. Cette phase est nécessaire afin qu'il soit possible d'appliquer les algorithmes de recherche des motifs fréquents sur des données de natures différentes provenant de sources différentes, de concentrer la recherche sur les données utiles pour l'application et de minimiser les temps d'extraction [5].

OID	ITEMS
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E
6	B C E

Tableau 2.1: Contexte d'extraction des motifs fréquents D.

Extraction des ensembles fréquents d'attributs : Cette phase consiste à extraire du contexte tous les ensembles d'attributs binaires $l \subseteq A$, appelés itemsets, qui sont fréquents dans le contexte D. Un itemset l est fréquent si son support, qui correspond au nombre d'objets du contexte qui «contiennent » l, est supérieur ou égal au seuil minimal de support minsupport défini par l'utilisateur. L'ensemble des itemsets fréquents dans le contexte D [5].

Le problème de l'extraction des itemsets fréquents est de complexité exponentielle dans la taille m de l'ensemble d'items puisque le nombre d'itemsets fréquents potentiels est 2^m. Ces itemsets forment un treillis dont la représentation sous forme de diagramme de Hasse pour le contexte D est présentée dans la Figure 2.1. De plus, des balayages du contexte doivent être réalisés lors de cette phase, et il est donc nécessaire de développer des méthodes efficaces d'exploration de cet espace de recherche exponentiel [5].

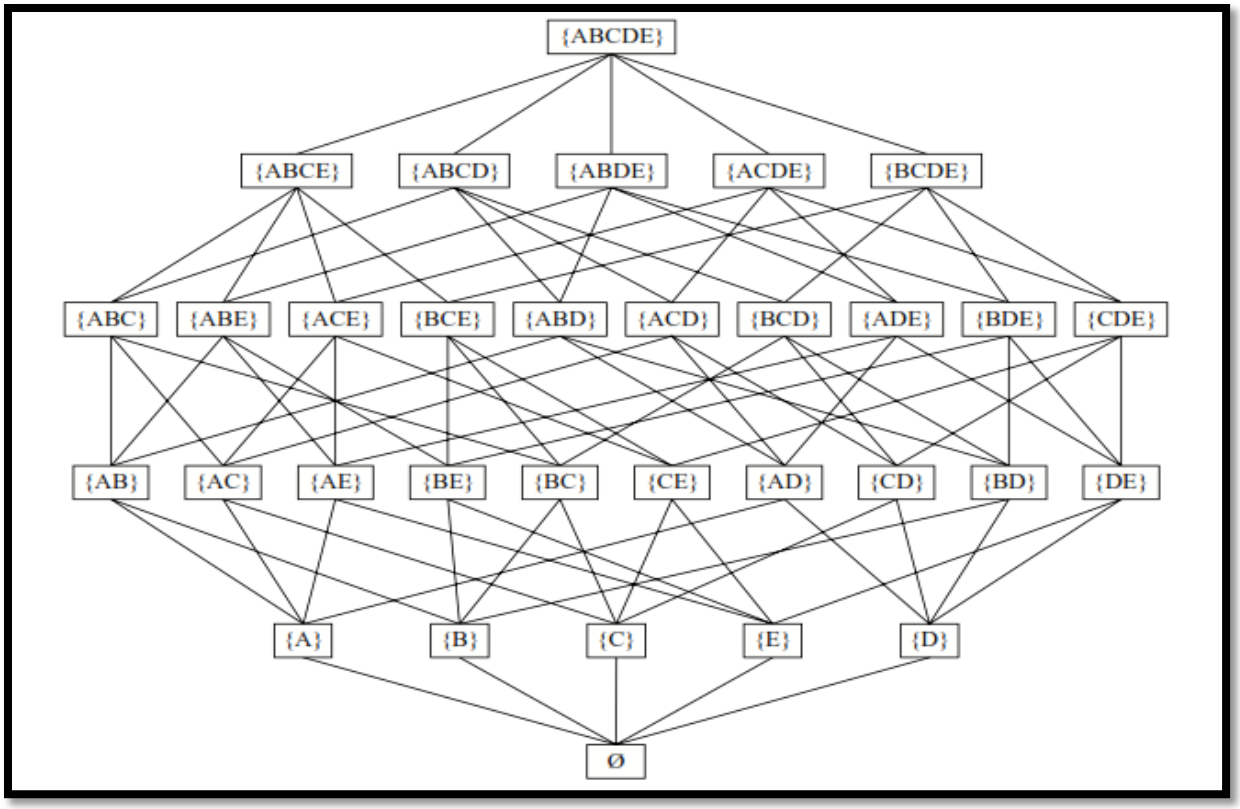


Figure2.1 : Diagramme de Hasse représentant le treillis des itemsets [5].

2. Définition du problème de recherche des motifs fréquents:

Le problème de la recherche des motifs fréquents fut introduit par Agrawal et al. (Agrawal et al, 1993b), Le problème de la recherche des motifs fréquents consiste à fixer un seuil de support minimal *minsup* et à rechercher, dans la base des transactions, tous les motifs qui ont un support supérieur ou égal à *minsup*. [5].

Le problème de recherche des motifs fréquents associé aux données (O, A, D, f min) consiste à déterminer le sous-ensemble M ⊆ M des motifs fréquents ainsi que la fréquence de chaque motif fréquent. Ou D : la base de données, O : ensemble d'objets, A : ensemble d'attributs [5].

Les algorithmes de recherche des motifs fréquents doivent parcourir la totalité de la base de données chaque fois qu'ils doivent déterminer la fréquence d'un ou de plusieurs motifs. La base de données D peut être grande au point de ne pouvoir être stockée en mémoire vive [2].

La base réside alors sur un disque dur et sa lecture se traduit par des accès au disque beaucoup plus lents que des accès à la mémoire. La fonction fréquence (le pseudo-code) permet d'accéder efficacement au disque pour déterminer toutes les fréquences d'un ensemble de motifs C ⊆ M : une seule passe est en effet nécessaire, selon une lecture séquentielle beaucoup plus rapide qu'une lecture aléatoire. La fonction place les fréquences calculées dans une table d'association F : M→ N stockant certaines associations entre un motif M et sa fréquence F [M] [5];

Pour être efficaces, les algorithmes de recherche de motifs fréquents doivent alors concilier deux objectifs antagonistes : d'une part le nombre d'accès à la base doit être réduit autant que possible compte tenu des accès particulièrement lents au disque. D'autre part, seuls les motifs susceptibles d'être fréquents doivent être générés et évalués car le nombre de motifs possibles croit de manière exponentielle avec le nombre n d'attributs ($|M| = 2^n$). La solution consistant à calculer la fréquence de tous les motifs possibles en un seul accès à la base n'est donc pas envisageable [5].

Des définitions :

- ✓ ***Une Transaction*** **:**

Représente une collection de donnée de taille un au minimum et de taille n au maximum, et que n : nombre d'item, article ou attribut. Cela s'est produit après le calcul de fréquence pour concevoir une base de transaction, pour former une transaction il est important d'utilisé le codage des items exemple : A …Abricot, B …Banane, pour minimiser la taille de la base de transaction et facilite le traitement sur ces données [3].

- ✓ ***Une Base de transaction*** **:**

Soit O un ensemble fini d'objets, P un ensemble fini d'éléments ou items, et R une relation binaire entre ces deux ensembles. On appelle base de transaction ou contexte formel, le triplet D = (O, P, R). On supposera que l'ensemble P est (totalement) ordonné, par exemple par l'ordre lexicographique [3]**.**

TID	Items.
1	A, C
2	B, C, D
3	A, C, D
4	C
5	A, D

Tableau 2.2 : Exemple d'une base de transaction.

Dans l'exemple si dessus cette base de transaction contenant une partie (TID) transaction identification c'est le numéro de la transaction, et la deuxième partie c'est des transactions de taille différentes, ici il y a cinq transactions formé d'items = {A, B, C, D}.

- ✓ ***Un Motif* :**

Un motif est un sous-ensemble de P. On dit qu'un motif P est inclus dans l'objet o (ou que o contient P) si P et o sont en relation : $\forall p \in P, (o, p) \in R$. Un motif de taille k est noté k-motif. Les motifs sont aussi appelés ensembles d'items (« itemsets » dans la littérature anglo-saxonne). Par abus de langage, nous parlerons indifféremment de « motif de P » et de « motif de D » [2].

- ✓ ***Un support :***

Le support d'un motif est la proportion des objets de la base de données qui le contiennent :

$$sup(P) = card(f(P)) / card(O).$$

Cette définition est relative à la taille de la base de données ; le support est parfois défini de manière absolue. Puisque f est antitone[*2], cette fonction est décroissante par rapport à la taille du motif : le support d'un ensemble est toujours inférieur ou égal au support de ses sous-ensembles [2].

- ✓ ***Un motif fréquent :***

Soit un seuil $minsup \in [0, 1]$, appelé le support minimum. Un motif est dit fréquent :

- si $sup(P) \geq minsup$.

Le motif vide est évidemment fréquent quel que soit le support minimal, et fort peu intéressant [2].

1. Algorithme *fréquence*

1. Algorithme :(D, O, C, F) → F :

---

1 : pour (chaque M ∈ C) faire

2 : F [M] ← 0 ;

3 : fin pour

4: pour (chaque o ∈ O) faire

5: Lecture de d(o) dans la base D ;

6: pour (chaque M ∈ C) faire

7: si (M ⊆ d(o)) alors

8: F [M] ← F [M] + 1 ;

9: fin si

10: fin pour

11: fin pour

12: retourner F ;

- **Propriétés** :

✓ Si un ensemble n'est pas fréquent alors aucun de ses sur-ensemble ne peut être fréquent.

✓ Si un ensemble est fréquent alors tous ses sous-ensemble le sont.

Exemple :

Voici une base de transaction :

O I D	Transaction
1	A, B, D
2	A, D, C
3	A, C
4	C
5	D, C

Tableau 2.3 : base de transaction (motif fréquent).

Avec un support minimum = 2/5.

Pour A : sup(A) = 3/5 → A est un motif fréquent.

Pour B : sup(B) = 1/5 → B n'est pas un motif fréquent.

Pour C : sup(C) = 4/5 → C est un motif fréquent.

Pour D : sup(D) = 3/5 → D est un motif fréquent.

F = {A, C, D}.des motifs fréquents.

Un algorithme très connu et importante dans la recherche des motifs fréquents est l'algorithme Apriori.

3. Présentation de l'algorithme Apriori :

Tous les motifs fréquents fermés qu'on a entrains d'étudier sont des motifs fréquents alors on doit présenter un algorithme connu de recherche des motifs fréquents et extraction des règles d'association est l'algorithme *Apriori*.

L'algorithme *APriori* est un algorithme d'exploration de données conçu en 1993, par *Rakesh Agrawal* et *Ramakrishnan Sikrant*, dans le domaine de l'apprentissage des règles d'association [5]. Il sert à reconnaitre des propriétés qui reviennent fréquemment dans un ensemble de données

L'algorithme est établi en une très importante étape :

- ✓ Recherche des k-itemsets fréquents (support ≥ MINSUP) (Pain, Fromage, Lait) = 3-itemset.

Principe : Les sous-itemsets d'un k-itemset fréquent sont obligatoirement fréquents.

Propriétés:

- ✓ Si A et B sont deux attributs alors le motif {A, B} est noté par AB.
- ✓ si M1et M2 sont deux motifs, nous noterons M1M2 le motif résultant de l'union de ces deux motifs : M1M2= M1∪ M2.
- ✓ Soit une base de transactions T définie par un ensemble d'attributs I. soit M1et M2 deux motifs données et σ un seuil de support fixé au préalable. Alors :

  Support(M1) < σ ⇒ Support(M1M2) < σ.

L'algorithme Apriori repose sur les propriétés suivantes :

- Les itemsets construits à partir d'itemsets.
- Un sous-itemset d'un itemset.
- Un k-itemset Z candidat est si les itemsets sont triés et si Z est un k-itemset candidat, alors il existe [5].

2. Algorithme Apriori :

Données : Base de transaction, seuil σ

Sorties : L'ensemble des motifs fréquents F

1 : **F ← Θ ;**

2 : **Générer tous les motifs de taille 1 ;**

3 : **Mettre les motifs fréquents dans C_1 ;**

4 : **Garder les motifs fréquents dans F ;**

5 : **tant que $C_i \neq \Theta$ faire**

6 : **C_{i+1} ← Générer le niveau i + 1 à partir de C_i ;**

7 : **C_{i+1} ← Vérifier l'hérédité entre C_{i+1} et C_i ;**

8 : **C_{i+1} ←Vérifier les valeurs de support dans C_{i+1} ;**

9 : **Garder les motifs fréquents de C_{i+1} dans F ;**

10 : **fin ;**

11 : **retourner F ;**

4.Exemple de déroulement de l'algorithme Apriori:

Soit la table de transaction suivante .Fixons le support = 20%.un motif fréquent doit apparaitre dans au moins deux transactions.

Transaction	Items
100	{1, 2}
200	{1, 2, 3}
300	{1, 3, 4}
400	{2, 4}
500	{2, 4, 5}
600	{2, 5}
700	{3, 4}
800	{3, 4, 5}
900	{3, 4, 5, 7}
1000	{3, 5, 6}

Tableau 2.4 : Base transactionnelle de déroulement de l'algorithme Apriori.

Its	Fréquence
1	3
2	5
3	6
4	6
5	5
6	**1**
7	**1**

Ensemble C1

Tableau 2.5 : Tableau de l'ensemble C1 de l'algorithme Apriori.

It1	It2	Fréquence
1	2	2
1	3	2
1	**4**	**1**
1	**5**	**0**
2	**3**	**1**
2	4	2
2	5	2
3	4	4

3	5	3
4	5	3

Ensemble C2

Tableau 2.6 : Tableau de l'ensemble C2 de l'algorithme Apriori.

It1	It2	It3	Fréquence
1	**2**	**3**	**1**
2	**4**	**5**	**1**
3	4	5	2

Ensemble C3

Tableau 2.7 : Tableau de l'ensemble C3 de l'algorithme Apriori.

Motifs	Support
{1}	3/10
{2}	5/10
{3}	6/10
{4}	6/10
{5}	5/10
{1, 2}	2/10
{1, 3}	2/10
{2, 4}	2/10
{2, 5}	2/10
{3, 4}	4/10
{3, 5}	3/10
{4, 5}	2/10
{3, 4, 5}	2/10
Les motifs fréquents	

Tableau 2.8 : Tableau final d'extraction des motifs fréquents avec Apriori

Chapitre III

Extraction des motifs fréquents fermés : L'algorithme *Close*

1. Des définitions :

1.1. Définition d'un motif fréquent fermé :

Un motif fermé est un ensemble maximal de motifs communs à un ensemble d'objets. Un motif i tel que support(i) >= minsup est appelé motif fréquent, mais un motif fréquent fermé :est un motif fréquent avec condition que: support de ce motif fréquent fermé ≠ support de ses sur ensemble,

Par exemple, dans le contexte d'extraction du Tableau 3.1, le motif {B C E}, est un motif fermé car il est l'ensemble maximal d'items communs aux objets {2; 3; 5; 6} avec un support =4/6. Le motif {B C} n'est pas un motif fermé car tous les objets contenant B et C (objets 2, 3, 5 et 6) contiennent également l'item E avec un support = 4/6 alors on prend le maximal.

1.2. Définition de la fermeture :

La fermeture d'un ensemble d'attributs A est un ensemble d'attributs B tel que B apparaît dans les mêmes transactions que A.

Pour la calculer on utilise deux fonctions :

- **f** : associe à un ensemble d'attributs les transactions ou il apparaît
- **g** : associe à un ensemble de transactions les attributs qu'ils ont en commun

Soit A un ensemble d'attributs :

Fermeture(A) = $g_0 f(A)$.

2. L'approche générale de l'extraction des motifs fréquents fermés :

Les itemsets fermés fréquents sont définis en utilisant la fermeture, ces itemsets sont les itemsets fréquents qui sont fermés selon l'opérateur de fermeture γ qui est la composition de l'application ϕ, qui associe à un ensemble $O \subseteq O$ les items communs à tous les objets $o \in O$, et de l'application ψ , qui associe à un itemset $l \subseteq I$ les objets en relation avec tous les items $i \in l$. L'opérateur $\gamma = \phi \circ \psi$ associe à un itemset l l'ensemble maximal d'items communs à tous les objets contenant l, c'est-à-dire l'intersection de ces objets [5].

Les itemsets fermés fréquents, selon cet opérateur de fermeture, constituent un ensemble générateur non redondant minimal pour tous les itemsets fréquents et leurs supports, peuvent donc être déduits efficacement, sans accéder au jeu de données, à partir des itemsets fermés fréquents et leurs supports. Cette propriété découle du fait que le support d'un itemset fréquent est égal au support de sa fermeture et que les itemsets fréquents maximaux sont des itemsets fermés fréquents maximaux, Les itemsets fermés fréquents forment un treillis dont la taille est bornée par la taille du treillis des itemsets fréquents $2^{|A|}$.

Toutefois, en pratique, la taille de ce treillis est en moyenne bien inférieure à la taille du treillis des itemsets, Le diagramme de Hasse représentant le treillis des itemsets fermés associé au contexte D est représenté dans la Figure 3.1.

Les générateurs des itemsets fermés sont définis : les générateurs d'un itemset fermé f sont les itemsets minimaux g dont la fermeture est égale à f : γ (g) = f. Les générateurs de l'itemset fermé {BCE} sont les itemsets {BC} et {CE} car les itemsets {BE} et {C} sont fermés et la fermeture des itemsets {B} et {E} est {BE}. L'algorithme Close un algorithme d'extraction des itemsets fermés fréquents par niveaux : ils considèrent un ensemble de générateurs candidats d'une taille donnée, et déterminent leurs supports et leurs fermetures en réalisant un balayage du contexte lors de chaque itération. Les fermetures (fréquentes) des générateurs fréquents sont les itemsets fermés fréquents extraits lors de l'itération. Les générateurs candidats sont construits en combinant les générateurs fréquents extraits durant l'itération précédente. L'algorithme Close permet d'identifier les itemsets fermés fréquents et leurs générateurs parmi les itemsets fréquents, sans accéder au jeu de données [5].

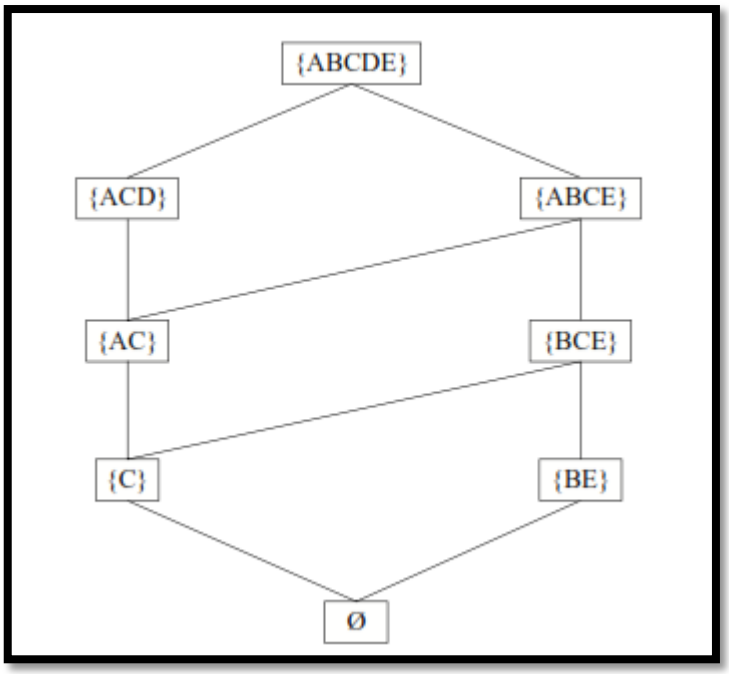


Figure 3.1 : Diagramme de Hasse représentant le treillis des itemsets fermés [5].

3. Présentation de l'algorithme Close :

L'algorithme Close d'extraction des itemsets fermés fréquents , proposé en 1998, est un algorithme itératif d'extraction des itemsets fermés fréquents qui parcourt l'ensemble des générateurs des itemsets fermés fréquents par niveaux [5]. Durant chaque itération k de l'algorithme, un ensemble FFC_k de k-générateurs candidats est considéré. Chaque élément de cet ensemble est constitué de trois éléments : le k-générateur k candidat, sa fermeture, qui est un itemset fermé candidat, et leur support. À la fin de l'itération k, l'algorithme stocke un ensemble FF_k contenant les k-générateurs fréquents, leurs fermetures, qui sont des itemsets fermés fréquents, et leurs supports.

L'algorithme commence par initialiser l'ensemble FFC_1 des 1-générateurs avec la liste des 1itemsets du contexte et exécute ensuite un ensemble d'itérations.

- ✓ Durant chaque itération k:

• La fermeture de tous les k-générateurs ainsi que leur support sont calculés. La détermination des fermetures des générateurs est basée sur la propriété que la fermeture d'un itemset l est égale à l'intersection de tous les objets du contexte contenant l dont le décompte fournit le support du générateur qui est identique au support de sa fermeture. Un seul balayage du contexte est donc nécessaire pour déterminer les fermetures et les supports de tous les k-générateurs.

• Tous les k-générateurs fréquents, dont le support est supérieur ou égal au seuil minimal de support minsupport, ainsi que leur fermeture et leur support sont insérés dans l'ensemble FF_k des itemsets fermés fréquents identifiés durant l'itération k [5].

• L'ensemble des (k+1)-générateurs candidats (utilisés durant l'itération suivante) est construit, en joignant les k-générateurs fréquents de l'ensemble FF_k comme suit (procédure GenGenerator):

1. Les (k+1)-générateurs candidats sont créés en joignant les k-générateurs de FF_k qui possèdent les mêmes k-1 premiers items. Les 3-générateurs {ABC} et {ABD} par exemple seront joints afin de créer le 4-générateur candidat {ABCD}.

2. Les (k+1)-générateurs candidats dont on sait qu'ils sont soit infréquents, soit non minimaux sont ensuite supprimés. Ces générateurs sont identifiés par l'absence d'un de leurs sous-ensembles de taille k parmi les k-générateurs fréquents de FF_k.

3. La troisième phase permet de supprimer parmi ces générateurs ceux dont la fermeture a déjà été calculée. Un tel générateur est identifié car il est inclus dans la fermeture d'un k-générateur fréquent de FF_k dont il est un sur-ensemble.

Les itérations cessent lorsqu'aucun nouveau générateur candidat ne peut être créé et l'algorithme s'arrête alors. L'algorithme Close, développé afin d'améliorer l'efficacité de l'extraction dans le cas de données faiblement corrélées, ne calcule pas les fermetures des générateurs candidats durant les itérations, mais lors d'un ultime balayage réalisé après la fin de ces itérations [5].

3.Algorithme close :

FFC_K Ensemble des K-itemsets fréquents candidats.

FC_K Ensemble des K-itemsets fermés fréquents.

Chaque élément de ces ensembles posséde trois champs :

i) Gen : le générateur ; ii) supp : le support
ii) Ferm : la fermeture.

Entrée : K : Contexte d'Extraction, minsup

Sortie : $FC=U_K\ FC_K$: Ensemble des itemsets fermés fréquents.

{/ Initialisation*/}*

1 : **FFC_1 = {1-itemsets}**

2 : **pour (K=1 ; FFC_k-gen ≠ Θ ; k++)**

3 : **FFC_k . ferm = Θ ;**

4 : **FFC_k. supp = Θ ;**

5 : **FFC_k = GEN – FERMETURE (FFC_k);**

{/ Etape de Construction*/}*

6 : **pour tous c appartient à FFC_k**

7 : **si c.supp ≥ minsup alors**

8 : **FC_K = FC_K U c ;**

9 : **finsi**

10 : **FFC_{k+1} = GEN – GENERATEUR (FC_k);**

11 : **finpour ;** 12 : **finpour**

12 : **retourner $FC=U_K\ FC_K$;**

4. Exemple de déroulement de l'algorithme Close :

OID	ITEMS
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E
6	B C E

Tableau 3.1: Contexte d'extraction des motifs fréquents fermés D.

Les tableaux suivants représentent l'exécution de l'algorithme Close au contexte d'extraction D pour un seuil minimal de support de 2/6.

1. Balayage de contexte D :

	FFC 1	
Générateur	**Fermé**	**Support**
{A}	{AC}	3/6
{B}	{BE}	5/6
{C}	{C}	5/6
{D}	{ACD}	1/6
{E}	{BE}	5/6

Tableau 3.2 : tableau de balayage 1 de l'algorithme Close.

2. Suppression des items sets infréquentés :

	FF 1	
Générateur	**Fermé**	**Support**
{A}	{AC}	3/6
{B}	{BE}	5/6
{C}	{C}	5/6
{E}	{BE}	5/6

Tableau 3.3 : tableau de Suppression des items sets infréquentés 1 de l'algorithme Close.

3. Balayage de contexte D :

	FFC 2	
Générateur	**Fermé**	**Support**
{AB}	{ABCE}	2/6
{AE}	{ABCE}	2/6
{BC}	{BCE}	4/6
{CE}	{BCE}	4/6

Tableau 3.4 : tableau de balayage 2 de l'algorithme Close.

4. Suppression des itemsets infréquents :

	FF 2	
Générateur	**Fermé**	**Support**
{AB}	{ABCE}	2/6
{AE}	{ABCE}	2/6
{BC}	{BCE}	4/6
{CE}	{BCE}	4/6

Tableau 3.5 : tableau de Suppression des items sets infréquentés 2 de l'algorithme Close.

L'ensemble FFC_1est initialisé avec la liste des 1-itemsets du contexte D. La procédure Gen-Fermeture génère les fermetures des 1-générateurs, qui sont les itemsets fermés fréquents potentiels, et leurs supports dans FFC_1 . Les groupes candidats de FFC qui sont fréquents sont insérés dans l'ensemble FF_1. La première phase de la procédure Gen-Generator appliquée à l'ensemble FF_1 génère six nouveaux 2-générateurs candidats : {AB}, {AC}, {AE}, {BC}, {BE} et {CE} dans FFC_2.

Les 2-générateurs {AC} et {BE} sont supprimés de FFC_2 par la troisième phase de la procédure Gen-Generator car nous avons {AC} $\subseteq$ γ ({A}) et {BE} $\subseteq$ γ ({B}). La procédure Gen-Fermeture calcule ensuite les fermetures et les supports du 2-générateurs restant dans FFC_2 et les ensembles FF_2 et FFC_2 sont identiques car tous les itemsets fermés de FFC_2 sont fréquents.

L'application de la procédure Gen-Generator à l'ensemble FF_2

Génère le 3-générateur {ABE} qui est supprimé car le 2-générateur {BE} n'appartient pas à FF_2 et l'algorithme s'arrête.

5. Apriori VS Close dans un exemple :

Soit la base de transaction suivante :

Transaction	Items
1	{A.B.D}
2	{B}
3	{C}
4	{B.C}
5	{B.D}

Tableau 3.6 : tableau d'exemple Apriori VS Close.

Avec un support minimum : 2/5.

1. L'application de l'algorithme Apriori :

Etape 1 :

{A} : 1/5 ;

{B} : 4/5 ;

{C} : 2/5 ;

{D} : 2/5 ;

on garde : {B ; C ; D}.

Etape 2 :

{B C} : 1/5 ;

{B D} : 2/5 ;

{C D} : 0 ;

on garde : {B D}.

- ✓ L'ensemble des motifs fréquent : {B ; C ; D ; BD}.

2. L'application de l'algorithme Close :

Etape 1 :

{A} : 1/5 ; Fermeture : ABD ;

{B} : 4/5 ; Fermeture : B ;

{C} : 2/5 ; Fermeture : C ;

{D} : 2/5 ; Fermeture : BD ;

- On ajoute {B}, {C} et {B D} à l'ensemble des itemsets fermés fréquents.
- On conserve {B}, {C} et {D} pour calculer les générateurs de niveau supérieur.

Etape 2 :

- À partir de {B}, {C} et {D}, on génère les ensembles

{B,C}; {B,D} ; {C,D}

- {B,D}⊆ f({D})

{BC} : 1/5 ; Fermeture :{BC}.

{CD} : 0.

Alors l'ensemble des motifs fréquents fermés :

{B ; C ; BD}.

- Donc on remarque que le nombre des motifs fréquents fermés est inférieur au nombre des motifs fréquents, intuitivement le gain en nombre des motifs générés est proportionnel à la taille de la base des transactions.

Chapitre IV

Implémentation de l'algorithme Close

1. Environnement de développement :

1.1 Le matériel utilisé :

Cet application est réalisée et exécutée sur une machine de système fabriquant LENOVO et un système d'exploitation Windows 7 Edition starter 32 bits (6.1, version 7600). Avec un processeur Intel(R) Core (TM) i3 CPU, plus une mémoire installé RAM : 4,00 Go.

1.2 pourquoi le langage java :

La programmation des algorithmes de la fouille de données est très compliqué à cause de nombre d'opérations et le calcul soit pour l'accès disque ou le temps d'exécution elle dépens de meilleurs langage surtout pour les applications comme la comparaison entre les algorithmes ou l'étude de l'un de ces algorithmes, Les caractéristiques de langage JAVA qu'elles m'attirent et que je vois mon application a besoin de ceux de ses caractéristiques sont :

JAVA est un langage développé par SUN et qui selon ses concepteurs est [7] :

Caractéristiques	Descriptions
Simple	Java est plus simple à utiliser, ✓ pas de pointeurs explicites. ✓ La gestion de la mémoire est transparente pour le programmeur.
Orienté objets	Un objet informatique est une entité munie de propriétés et de méthodes capables d'agir sur ses propriétés ou de réaliser certaines actions. Un objet contient des données et du code permettant de les manipuler. Dans JAVA, cette notion est poussée à l'extrême puisque dans ce langage tout est objet y compris le programme.
Robuste et sur	Le typage des données est extrêmement strict. Aucune conversion de type implicite pouvant provoquer une perte de précision n'est possible. Comme les pointeurs ne sont pas accessibles les erreurs de gestion mémoire sont impossibles. Pour les applets, il est en principe impossible d'accéder aux ressources de la machine hôte. Enfin lors de l'exécution, on vérifie que le code généré par le compilateur n'a pas été altéré.
Indépendant des architectures matérielles	Le compilateur génère un code universel le « byte-code ». Un interpréteur spécifique à l'ordinateur hôte appelé « machine-virtuelle »

	permet l'exécution des programmes. La représentation des données étant indépendante de la machine qui exécute le code, les résultats des calculs sont indépendants de cette machine.
Multitâches	JAVA permet l'exécution en apparence simultanée de plusieurs processus. En réalité on accorde de façon séquentielle un peu du temps processeur à chaque processus. On dit aussi multithread.

Tableau 4.1 : tableau des caractéristiques du langage java [6].

1.3 les données utilisées :

- Un fichier personnel de type text Doc1.txt, qui contient 23 transactions, et 7 items c'est un fichier d'essayé de petite taille leurs items sont de type entier.
- un fichier des bases de benchmark groupe à savoir Mushroom.dat décrivant les caractéristiques de champignons il est composé de 8124 transactions, 119 items et une taille moyenne des transactions 23. (http://fimi.ua.ac.be/data/mushroom.dat), c'est un fichier très connu existe sur le net, c'est une grande base de transaction que j'exécute par des différents valeurs de minsup.

2. Partie déclaration :

2.1. Les classes import utilisées :

Dans le package *proto* j'ai importé des classes prédéfini dans java comme se suit :

```
package proto;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

Figure 4.1. Les classes import utilisés

2.2. Déclaration des variables :

J'ai utilisé dans la partie de déclaration :

Des variables pour les statiques et calcul :

- protected int totalCandidateCount = 0;

  Le nombre des candidats générés durant la dernière exécution.

- protected long startTimestamp;

Le démarrage du temps avec la dernière exécution.

- protected long endTimestamp;

  Le temps d'arrêt de la dernière exécution.

- protected long timeExecution;
- Le temps d'exécution.

- private int itemsetCount;
  Le nombre des itemsets durant la dernière exécution.

Des variables de définition :

- private int databaseSize;
  La taille de la base de transaction.

- private int minsupRelative;
  La valeur de support minimum fixé par l'utilisateur.

- private List<int[]> database = null;
  Représentation de la base de transaction par une liste des transactions (integer) initialisé par null.

- protected proto.Itemsets motifs = null;
  Les motifs extraits.

- BufferedWriter writer = null;
  (Object) Le fichier output si l'utilisateur veut que son résultat affiché dans un fichier txt.

```
29
30      protected int k;
31
32      // variables for statistics
33      protected int totalCandidateCount = 0; // number of candidate generated during last execution
34      protected long startTimestamp; // start time of last execution
35      protected long endTimestamp; // end time of last execution
36      protected long timeExecution; // time of execution
37      private int itemsetCount;  // itemset found during last execution
38      private int databaseSize;
39
40      // the minimum support set by the user
41      private int minsupRelative;
42
43      // A memory representation of the database.
44      // Each position in the list represents a transaction
45      private List<int[]> database = null;
46
47
48      // The  patterns that are found
49      // (if the user want to keep them into memory)
50      protected  proto.Itemsets  motifs = null;
51
52      // object to write the output file (if the user wants to write to a file)
53      BufferedWriter writer = null;
54
55
56       ArrayList<Integer> Itemset;
57       ArrayList<Integer> Itemsets;
58
```

Figure 4.2 : Déclaration des variables

2.3. Définition des classes:

- les classes **Itemset, Itemsets** : sont deux classes qui contient des définitions pour des variables et méthodes son déclaré public pour pouvoir utilisé dans la classe principale.
- La classe **AlgoAprioriClose** : c'est la classe principale du notre application contient tous les méthodes intéressantes dans le programme.
- La classe **Main** : est juste une classe de correction et exécution et d'Apelle de la fonction principale.
- En fin la classe **closeSwing** : est la classe run ou l'exécutable de notre application, c'est une interface d'exécution.

3. L'exécution :

Après l'exécution du programme on aura cette interface qui contient quatre parties :

1. La première partie en haut présente l'identification du projet qui concerne l'Algorithme Close, et le nom de développeur.
2. La deuxième partie à droite présente le paramétrage ou les input comme :
   Un minsup qui est une valeur en pourcentage.
   La base de transaction qui par défaut un fichier .txt contient notre base de transaction j'ai déclaré un fichier de taille minimum juste d'essayé **Doc1.txt** mais on peut import autre fichier.

Un bouton à cocher si l'utilisateur veut avoir son résultat dans un fichier.txt comme le fichier **Doc2.txt**.
Un bouton à cliquer pour lancer l'exécution.

3. La troisième partie à gauche présente le output ou le résultat affiché.
4. La quatrième partie est en bas présente des informations d'exécution comme :
    - Le temps d'exécution en millisecondes.
    - Le nombre de transactions.
    - Et le nombre de motif fréquent fermé.

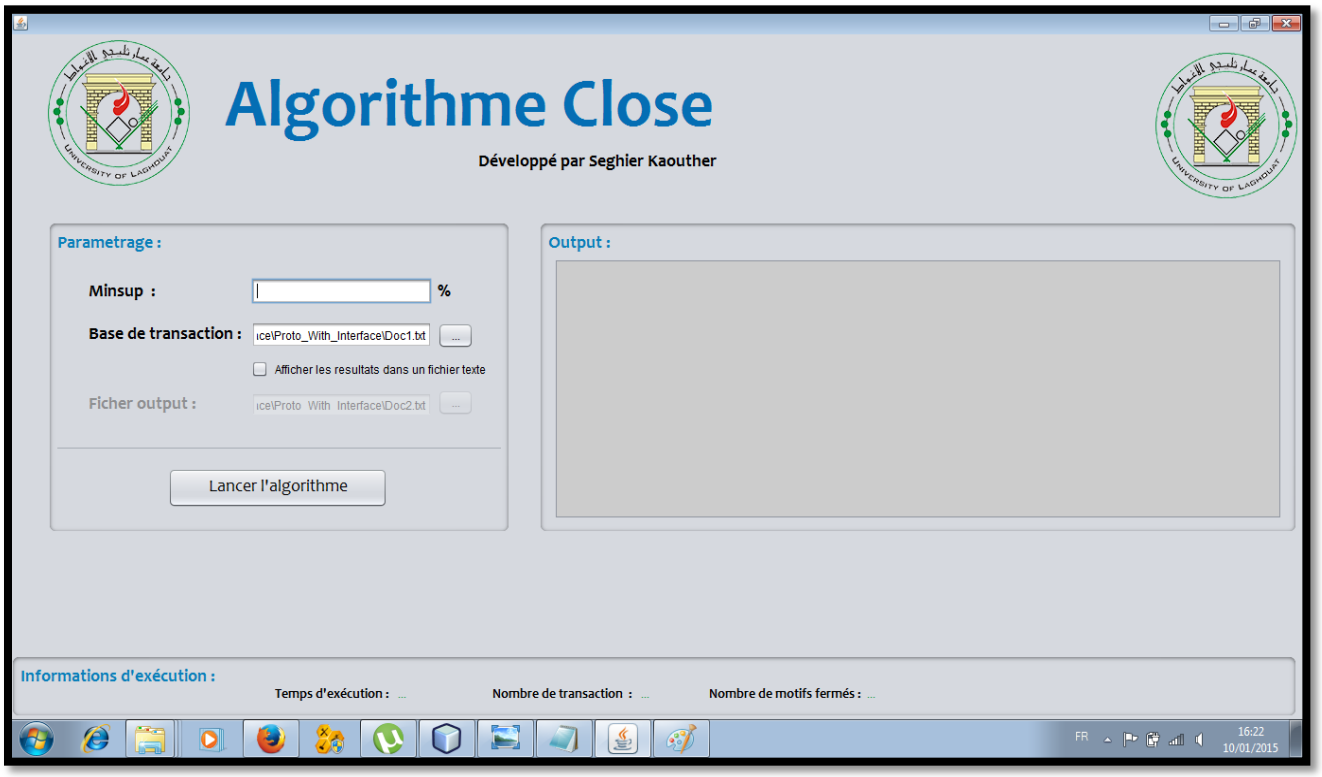


Figure 4.3 : L'interface d'exécution

4.Résultats & discussion :

Pour tester notre implémentation nous avons exécuté l'algorithme implémenté avec deux bases de transactions :

1. Une base crée manuellement forme de 23 transactions et 7 items.
2. Une bas transactionnelle de référence mushroom.dat, cette dernière est traditionnellement utilisée pour tester différents implémentations des algorithmes d'extraction des motifs fréquents [7],

Le fichier d'essayé Doc1.txt le input qui est de petite taille par apport aux les grandes base de transaction.

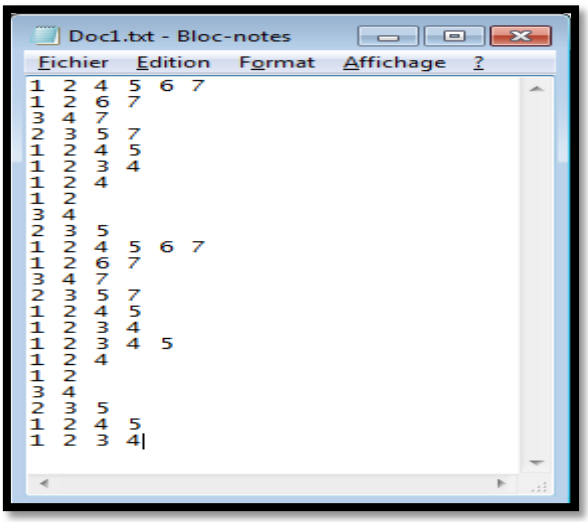


Figure 4.4 : Exemple de fichier en entrée

Après une première exécution sur le fichier, et avec un minsup de 50% on aura le résultat affiché sur la Figure 4.5 : le temps d'exécution : 2 ms, le nombre de transactions : 23, le nombre de motif fréquent fermé : 3. Qui est affiché sur l'output avec leurs support : {2} avec un support 19 ; {3} avec un support 12 ; {4} avec un support 15,

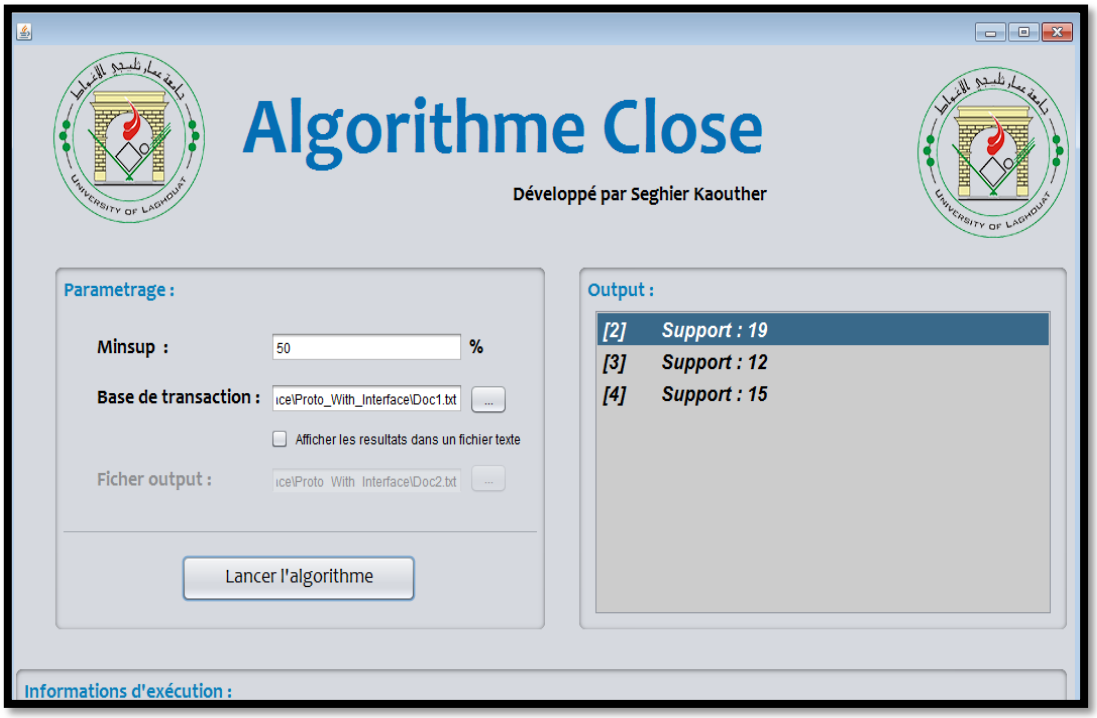


Figure 4.5 : le résultat de première exécution

Il est possible de sauvegarder les résultats d'exécution sur un fichier texte externe l'avantage de ce fichier est lorsque les résultats sont grandes donc ne peuvent pas être représentés clairement sur le output alors il suffit de paramétrer notre application comme indiqué dans la Figure 4.6.

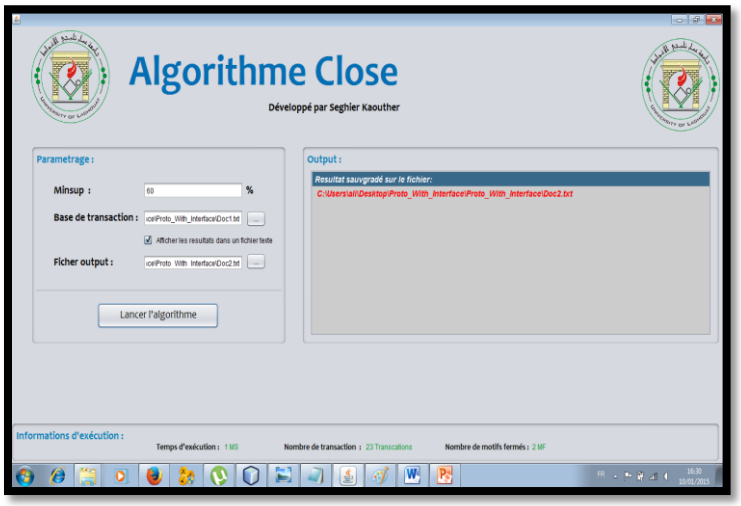


Figure 4.6 : résultat d'exécution avec un fichier output

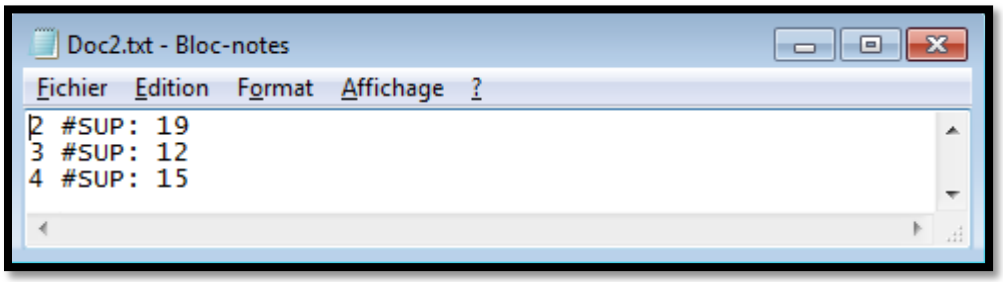


Figure 4.7 : le fichier output

Dans cette expérimentation nous avons exécuté ce fichier avec différents valeurs de minsup le tableau illustre les résultats obtenu.

Support minimum (%)	Temps d'exécution (ms)	Nombre MFF
100	0	0
90	0	0
80	1	1
70	1	1
60	1	2
50	2	3
40	2	3
30	3	4
20	4	5
10	4	7

Tableau 4.2 : Tableau des résultats d'exécution d'une base de transaction en termes de différents valeurs minsup.

Dans cette expérimentation nous avons utilisés un fichier des bases de benchmark groupe à savoir mushroom.dat il est composé de 8124 transactions, 119 items et une taille moyenne des transactions 23.

Les figures indiquent quelque résultat obtenu par plus grande valeur de minsup utilisé dans la Figure 4.8, et plus grande valeur de minsup utilisé dans la Figure 4.9.

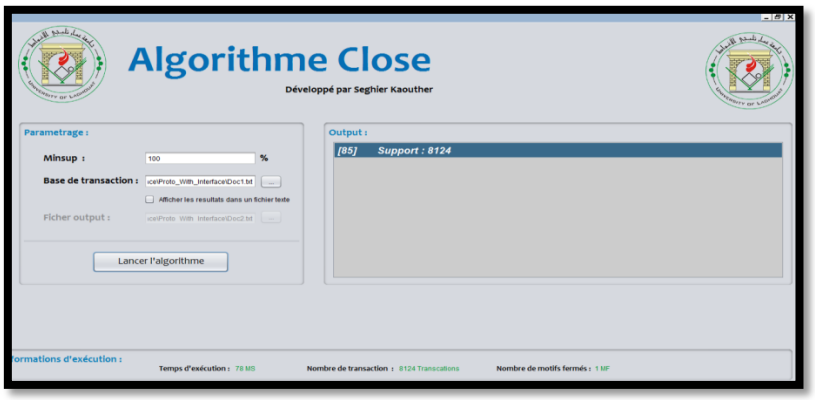


Figure 4.8 : le résultat de la deuxième exécution.

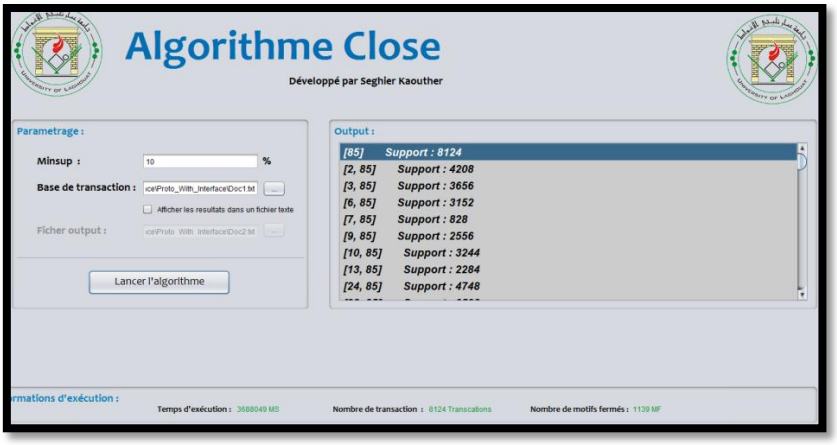


Figure 4.9 : résultat de la troisième exécution

On note que le temps d'exécution est inversement proportionnel à la valeur de support.

Nous avons exécuté notre implémentation pour différents valeurs de support les résultats obtenus sont résumés dans le Tableau 4.3.

SUPPORT (%)	NOMBRE MFF	TEMPS D'EXECUTION(MS)
100	1	78
90	3	110
80	4	120
70	4	135
60	7	187
50	15	265
40	39	499
30	104	1732
20	290	48454
10	1139	3688049

Tableau 4.3 : Tableau des résultats d'exécution de la base transactionnelle Mushroom.dat en termes de différents valeurs minsup.

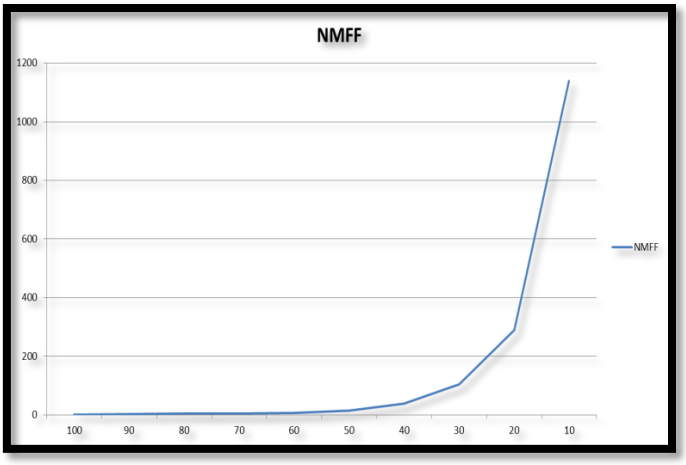


Figure 4.10 : graphe de résultat le nombre des motifs fréquents fermés en terme des différents valeurs de minsup.

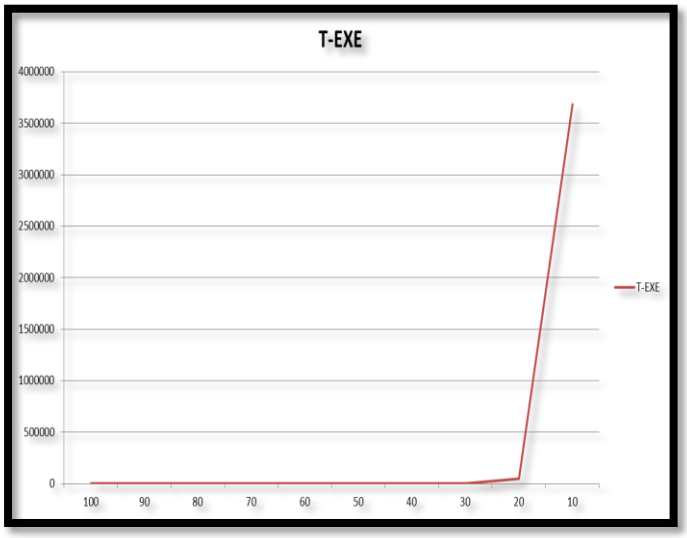


Figure 4.11: graphe de résultat le temps d'exécution en terme des différents valeurs de minsup

Conclusion Générale

Le présent travail nous a permis de découvrir le domaine de la fouille de données et les différentes techniques de traitement des bases de transaction comme classification, clustering, l'extraction des règles d'association et la recherche des motifs fréquents.

En particulier nous nous sommes focalisés sur le problème de la recherche des motifs fréquent. L'inconvénient de ce majeur problème est la volumétrie et complexité de l'extraction des motifs fréquents. En effet, pour n attributs, 2^n motifs sont générés et testés.

Une des solutions proposées pour remédier à ce problème est la représentation condensée des motifs fréquents.

Nous sommes intéressées aux l'extraction des motifs fréquents fermés et en particulier à l'implémentation de l'algorithme Close.

Notre implémentation est réalisé en langage Java et testé sur deux bases transactionnelles, pour différents valeurs de support minimum on a calculé le temps d'exécution et le nombre des motifs fréquents fermés extraits, les résultats obtenus montrent que le temps d'exécution est inversement proportionnel à la valeur de support, ceci peut être expliqué par le fait que plus la valeur du support augmente, moins est le nombre des motifs fréquents fermés extraits.

La partie expérimentale nous a montré la difficulté et la complexité de la tache de recherche des motifs fréquents d'une manière générale et les motifs fréquents fermés en particulier.

Une suite logique du travail consiste à l'étude d'autres représentations condensées des motifs fréquents et particuliers les motifs fréquents maximaux.

Bibliographie :

[2]: Han J., Kamber M., “Data Mining Concepts and Techniques”, Morgan Kaufmann, 2000.

[3] : René L., Gilles V. “Data mining”, édition Eyrolles, 2001.

[4]: Rakesh Agrawal and Ramakrishnan Srikant. “Fast algorithms for mining association rules”, Proceedings of the 20th International Conference on Very Large Databases, June 1994.

[5]: Nicolas Pasquier. «Data mining : algorithme d’extraction et de réduction des régles d’associations dans des bases de donnée », Thése de Doctoract Université Blaise Pascal Clermont-Ferrand, 2000.

Webographie :

[1] : Site : www.techniques-ingenieur.fr/base-documentaire/technologies-de-l-information-th9/bases-de-donnees-42309210/extraction-de-connaissances-a-partir-de-donnees-ecd-h3744/.
Consulté le : 15/05/2014.

[6]: Site : www.latrach.net/les-caracteristiques-de-java

Consulté le : 20/01/2015.

[7] : Site :www.fimi.ua.ac.be/data/mushroom.dat,

Consulté le : 22/01/2015.