Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Système d'Information et de Connaissance (S.I.C)*

Présenté par :

MAMI Mohammed Nassim

Thème

Extraction des connaissances dans un environnement distribué : synthèse

Soutenu oralement le 3 juillet 2013 devant la commission composée de :

Président :	-	*Mr. BENTAALLAH A.*
Encadreurs :	-	*Mr. SMAHI Med Ismail*
	-	*Mr. BOUAFIA Zoheir*
Examinateurs :	-	*Mme ILES N.*
	-	*Mme HALFAOUI A.*

Année universitaire : 2012-2013

Remerciements

J’adresse mes remerciements les plus sincères à toutes les personnes qui m’ont apporté leur aide et qui ont contribué à l’élaboration de ce mémoire.
Mes remerciements s’adressent également à Monsieur SMAHI Med Ismail qui est mon encadreur, pour sa générosité, sa confiance, et la grande patience dont il a su faire preuve malgré ses charges.

Je tiens à remercier sincèrement Monsieur BOUAFIA Zoheir, qui, en tant qu’encadreur de ce mémoire, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de mon travail, ainsi que pour l'aide et le temps qu'il a bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Je remercie aussi les membres du jury Mr A. BENTAALLAH d’avoir accepté de présider ce jury, Mme N. ILES et Mme A. HALFAOUI d’avoir accepté d’évaluer et d’examiner ce travail, et de consacrer une partie de leurs précieux temps afin de le juger au mieux.

Dédicaces

Je dédie ce modeste travail

A mes chers parents
En témoignage de ma profonde gratitude et de mon incontestable reconnaissance, pour tous les sacrifices qu'ils me contentent, toute la confiance qu'ils m'accordent et tout l'amour dont ils m'entourent.

A mes sœurs
Souhila, Fatima, Meriem et Sihem que je ne trouverais jamais assez de mots pour leurs exprimer mon amour, En leurs espérant le plein succès.

Aux amis
Nesrine que je remercie pour son soutien et ses encouragements, à mes meilleurs amis, à tous les collègues du Master SIC, et à tous les amis de l'université de Tlemcen.

A tous ceux qui m'ont aidé à réaliser ce travail.

M. Nassim

Résumé

Ce travail s'inscrit dans le domaine du Data Mining qui est une étape importante du processus d'Extraction des Connaissances à partir de Données (ECD) [Knowledge Discovery in Databases (KDD)].

Dans l'ensemble des travaux existants, l'extraction des connaissances est décomposée en deux sous problèmes : la recherche des itemsets fréquents et la génération des règles d'associations. Nous nous sommes intéressées par le premier sous problème, dont la complexité est exponentielle, et constitue la phase la plus couteuse en termes de temps d'exécution et d'espace mémoire. Toutefois, la plupart des algorithmes séquentiels d'extraction des itemsets fréquents voient leurs performances se dégrader lorsque la taille des données augmente. Pour maintenir les performances de ces algorithmes, l'utilisation de méthodes et outils parallèles et distribués apparait comme une solution naturelle.

Le but de ce projet est d'étudier le problème d'extraction des connaissances, en se basant essentiellement sur l'étape d'extraction des itemsets fréquents dans un environnement distribué. Après une synthèse et une classification des algorithmes les plus cités dans la littérature, nous nous somme intéressés à l'un des algorithmes distribué : ECLAT.

Mots clés : Datamining parallèle et Distribué, Itemsets Fréquents, algorithme ECLAT.

Table des matières

1 Datamining Distribué 9
1.1 Introduction 10
1.2 Définition du datamining 10
1.3 Le processus du datamining 11
1.4 Méthodes de datamining 14
1.4.1 Méthodes descriptives (ou non supervisées) 14
1.4.2 Méthodes prédictives (ou supervisées) 14
1.5 Les tâches effectuées par le datamining 14
1.5.1 La description 14
1.5.2 L'estimation 14
1.5.3 La segmentation 15
1.5.4 La classification 15
1.5.5 La prévision 15
1.5.6 L'association 16
1.6 Les domaines d'applications 16
1.6.1 Exemple 16
1.7 Apport du datamining distribué 17
1.8 Architectures parallèles 18
1.8.1 Parallélisme de type SMP/CCNUMA 18
1.8.2 Parallélisme sur réseaux de stations et grille de calcul 19
1.9 Partitionnement de la base de données 20
1.10 Techniques du datamining distribué 22
1.10.1 Techniques Parallèles 22
1.10.2 Techniques d'agrégation 22
1.11 Conclusion 25
2 Processus d'extraction des connaissances dans un environnement séquentiel 26
2.1 Introduction 27
2.2 Extraction des connaissances 28
2.3 Extraction des itemset fréquent 28
2.3.1 Notions de base 29
2.3.2 Exemple 30

2.4 Génération des règles d'associations 32
2.4.1 Notions de base 32
2.4.2 Obtenir une règle d'association à partir d'un motif 32
2.5 Conclusion 35
3 Algorithmes distribués d'extraction des connaissances 36
3.1 Introduction 37
3.2 Notations 38
3.3 Count Distribution (CD) 38
3.3.1 Les etapes de l'algorithme CD 40
3.3.2 Exemple de déroulement 41
3.3.3 Discussion 44
3.4 Data distribution (DD) 45
3.4.1 Les étapes de l'algorithme DD 47
3.4.2 Exemple de déroulement 49
3.4.3 Discussion 52
3.5 Intelligent data distribution (IDD) 53
3.5.1 Les étapes de l'algorithme IDD 53
3.6 Eclat 54
3.6.1 Partitionnement en classe d'équivalence 54
3.6.2 Les étapes de l'algorithme Eclat 55
3.6.3 Exemple de déroulement 56
3.6.4 Discussion 58
3.7 Classification des algorithmes 58
3.8 Conclusion 60

Liste des figures

FIGURE 1 PROCESSUS KDD [1] 8
FIGURE 2 PROCESSUS DE DATAMINING (CRISP-DM) [3] 11
FIGURE 3 ARCHITECTURE SMP [6] 18
FIGURE 4 ARCHITECTURE DE TYPE RESEAU DE STATIONS ET GRILLE DE CALCUL [6] 19
FIGURE 5 POSSIBILITE DE DISTRIBUTION DE LA BASE DE DONNEES [6] 20
FIGURE 6 ETAPE DU PROCESSUS D'EXTRACTION DES CONNAISSANCES [9] 28
FIGURE 7 TREILLIS D'ITEMSETS [9] 31
FIGURE 8 TREILLIS D'ITEMSETS FREQUENTS 31
FIGURE 9 COMMUNICATION ALL-TO-ALL [10] 45

Introduction générale

Durant ces dernières années, on assiste à une forte augmentation tant dans le nombre que dans le volume des informations mémorisées par des bases de données scientifiques, économiques, financières, administratives, médicales etc.

Le stockage en lui même ne pose pas de réelles difficultés du point de vue informatique, mais le besoin d'interpréter ou de trouver de nouvelles relations entre les éléments stockés dans ces bases a suscité beaucoup d'intérêt. Ainsi, la mise au point de nouvelles techniques informatiques est devenu un thème important pour un bon nombre de chercheurs. Le "**Knowledge Discovery in Databases**" **(KDD)** et le "**Datamining**" représentent un domaine émergeant essayant de répondre à ces objectifs.

Le Knowledge Discovery in Databases (KDD) ou Extraction des Connaissances à partir de Données (ECD) est l'extraction d'information implicite, non connue et potentiellement utile, stockée dans des bases volumineuses. Ses concepts s'appuient sur le constat qu'il existe au sein de chaque entreprise des informations cachées dans des gisements de données appelés entrepôt de donnée (Data Warehouse). Ils permettent, grâce à un certain nombre de techniques spécifiques, de faire apparaître des connaissances. Dans la littérature, on distingue différents objectifs du KDD, à savoir la classification, le regroupement, la régression, la découverte de règles associatives, etc. En effet, l'information découverte peut être exprimée sous forme d'un ensemble de règles associatives, qui permettent de définir des liens entre les données, et par la suite, prédire la conduite d'autres données différentes de celles stockées dans la base.

Le Datamining ou Fouille de données est souvent vu comme un processus équivalent au KDD, bien que la plupart des chercheurs voient en lui une étape essentielle de la découverte de connaissance. C'est en effet une étape non triviale du processus d'extraction des connaissances qui consiste à identifier des motifs (patterns) valides, nouveaux, potentiellement utiles et compréhensibles à partir d'une grande collection de données.

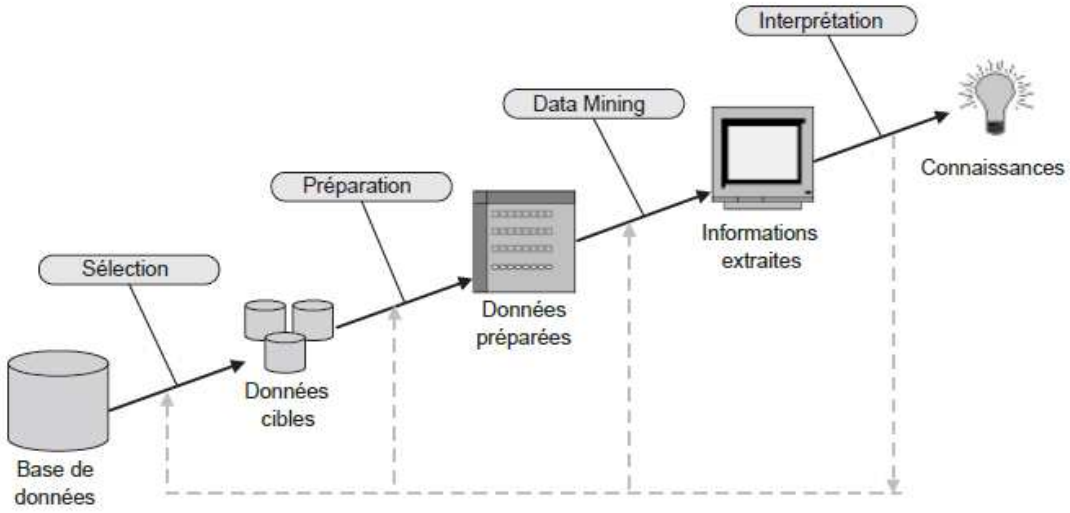


Figure 1 Processus KDD [1]

Le problème traité dans ce mémoire est l'extraction des itemsets fréquents dans un environnement de calcul distribué. Nous présentons un échantillon d'algorithmes distribués cités dans la littérature, en nous focalisant essentiellement sur l'étape de découverte des itemsets fréquents avec des exemples de déroulement. Ensuite nous proposons une classification de ces algorithmes.

Le reste de ce mémoire est organisé comme suit :

Dans le premier chapitre, nous donnons un bref aperçu sur le datamining distribué en présentant ses avantages, et les différentes architectures et techniques existants pour ce type de calcul.

Dans le deuxième chapitre, nous présentons le processus d'extraction des connaissances dans un environnement séquentiel.

Pour le chapitre qui suit, nous présentons un échantillon d'algorithmes distribués cités dans la littérature, en nous focalisant essentiellement sur l'étape de découverte des itemsets fréquents avec un exemple de déroulement. Nous proposons aussi une classification de ces algorithmes sur la base de critères et de dimensions que nous avons définis à cet effet.

Enfin, nous terminons ce mémoire par une conclusion et par la proposition de quelques perspectives de recherche.

1 Datamining Distribué

1.1 Introduction

L'accroissement du volume des données stockées dans les bases a fait qu'il est devenu urgent et vital de recourir à des techniques et méthodes pour extraire de l'information à partir de cette masse volumineuse de données. Le Datamining est alors devenu rapidement un thème de recherche suscitant l'intérêt de la communauté scientifique.

Dans ce chapitre nous allons d'abord faire un tour d'horizon sur le datamining, ensuite nous verrons la nécessité du datamining distribué.

1.2 Définition du datamining

Le datamining signifie littéralement « fouille de données » ou « forage de données». Ce procédé, basé sur une série d'algorithmes ou modèles de datamining permet d'extraire des informations à partir de données, informations qui, grâce à l'analyse, se convertissent en connaissances. [2]

« Le datamining est l'analyse d'un ensemble d'observations qui a pour but de trouver des relations insoupçonnées et résumer les données d'une nouvelle manière, de façon qu'elles soient plus compréhensibles et utiles pour leurs détenteurs » (David Hand, 2001). Autrement dit, il consiste à analyser des informations collectées dans des entrepôts de données afin d'y détecter des relations qu'il serait a priori impossible d'identifier sans cet outil. C'est un élément essentiel dans la relation client et de système d'aide à la décision.

Le datamining est annoncé comme étant «un des développements technologiques les plus révolutionnaires des dix prochaines décennies » selon le magasine en ligne ZDNET News. (Rachel Konrad, février 2001). En effet, cette technologie est perçue comme étant réellement indispensable de nos jours à l'analyse de la quantité toujours plus vaste d'informations produites par tous les systèmes d'information de l'entreprise.

En tant que processus, il est pertinent de souligner ici que le datamining ne se réfère pas seulement à des outils et à une technologie informatique très développée. Effectivement, il faut également relever le rôle fondamental de l'humain dont l'implication se doit d'être totale dans chaque phase du processus. Il est erroné de penser que le datamining est une entité qui fonctionne de manière autonome.

1.3 Le processus du datamining

Il est très important de comprendre que le datamining n'est pas seulement le problème de découverte de modèles dans un ensemble de donnée. Ce n'est qu'une seule étape dans tout un processus suivi par les scientifiques, les ingénieurs ou toute autre personne qui cherche à extraire les connaissances à partir des données. En 1996 un groupe d'analystes définit le datamining comme étant un processus composé de cinq étapes sous le standard CRISP-DM (Cross-Industry Standard Process for Datamining) [3] comme schématisé ci-dessous :

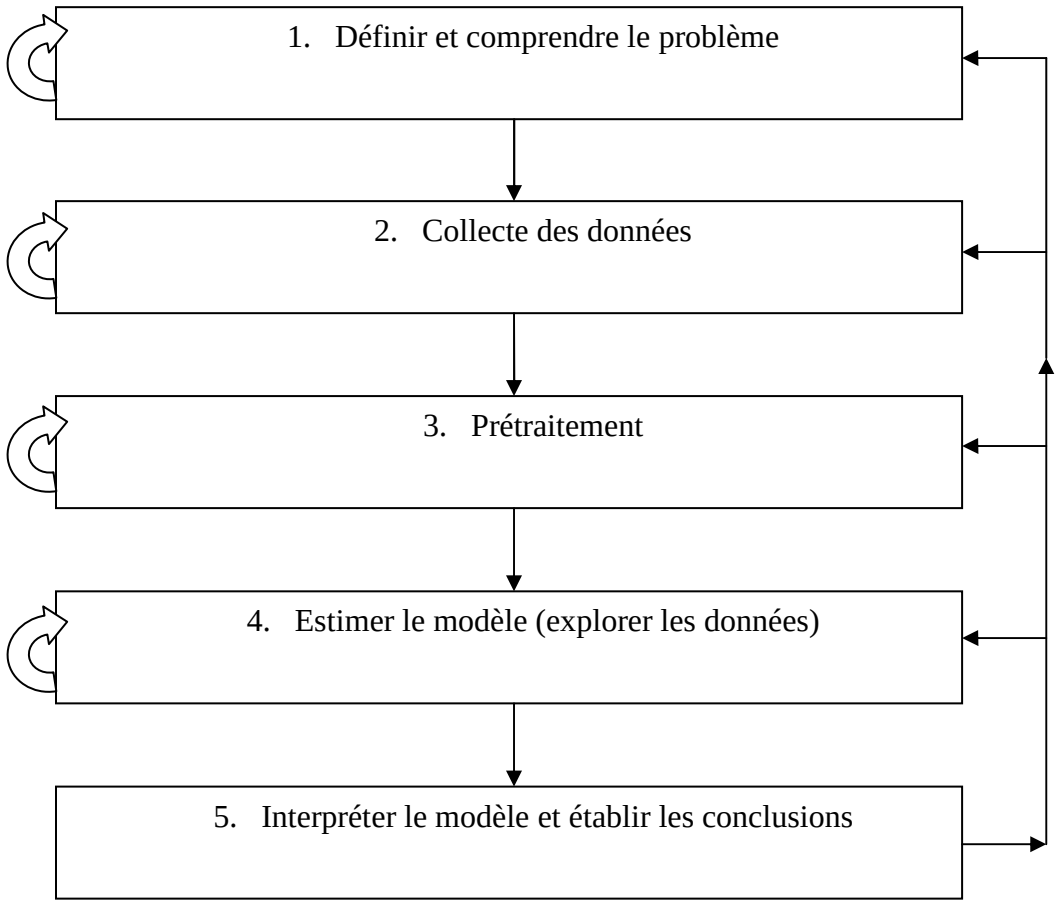


Figure 2 Processus de datamining (CRISP-DM) [3]

Ce processus, composé de cinq étapes, n'est pas linéaire, on peut avoir besoin de revenir à des étapes précédentes pour corriger ou ajouter des données. Par exemple, on peut découvrir à l'étape d'exploration (5) de nouvelles données qui nécessitent d'être ajoutées aux données initiales à l'étape de collection (2). Décrivons maintenant ces étapes :

1. Définition et compréhension du problème : Dans la plus part des cas, il est indispensable de comprendre la signification des données et le domaine à explorer. Sans cette compréhension, aucun algorithme ne va donner un résultat fiable. En effet, Avec la compréhension du problème, on peut préparer les données nécessaires à l'exploration et interpréter correctement les résultats obtenus. Généralement, le datamining est effectué dans un domaine particulier (banques, médecine, biologie, marketing, ...etc) où la connaissance et l'expérience dans ce domaine jouent un rôle très important dans la définition du problème, l'orientation de l'exploration et l'explication des résultats obtenus. Une bonne compréhension du problème comporte une mesure des résultats de l'exploration, et éventuellement une justification de son coût. C'est-à-dire, pouvoir évaluer les résultats obtenus et convaincre l'utilisateur de leur rentabilité.

2. Collecte des données : dans cette étape, on s'intéresse à la manière dont les données sont générées et collectées. D'après la définition du problème et des objectifs du datamining, on peut avoir une idée sur les données qui doivent être utilisées. Ces données n'ont pas toujours le même format et la même structure. On peut avoir des textes, des bases de données, des pages web, ...etc. Parfois, on est amené à prendre une copie d'un système d'information en cours d'exécution, puis ramasser les données de sources éventuellement hétérogènes (fichiers, bases de données relationnelles, temporelles, ...). Quelques traitements ne nécessitent qu'une partie des données, on doit alors sélectionner les données adéquates. Généralement les données sont subdivisées en deux parties : une utilisée pour construire un modèle et l'autre pour le tester. On prend par exemple une partie importante (suffisante pour l'analyse) des données (80 %) à partir de laquelle on construit un modèle qui prédit les données futures.

Pour valider ce modèle, on le teste sur la partie restante (20 %) dont on connaît le comportement.

3. Prétraitement : Les données collectées doivent être "préparées". Avant tout, elles doivent être nettoyées puisqu'elles peuvent contenir plusieurs types d'anomalies : des données peuvent être omises à cause des erreurs de frappe ou à causes des erreurs dues au système lui-même, dans ce cas il faut remplacer ces données ou éliminer complètement leurs enregistrements. Des données peuvent être incohérentes c.-à-d. qui sortent des intervalles permis, on doit les écarter où les normaliser. Parfois on est obligé à faire des transformations sur les données pour unifier leur poids. Un exemple de ces transformations est la normalisation des données qui consiste à la projection des données dans un intervalle bien précis [0,1] ou [0,100] par exemple. Un autre exemple

est le lissage des données qui considère les échantillons très proches comme étant le même échantillon. Le prétraitement comporte aussi la réduction des données qui permet de réduire le nombre d'attributs pour accélérer les calculs et représenter les données sous un format optimal pour l'exploration. Une méthode largement utilisée dans ce contexte, est l'analyse en composantes principales (ACP).

Une autre méthode de réduction est celle de la sélection et suppression des attributs dont l'importance dans la caractérisation des données est faible, en mesurant leurs variances. On peut même réduire le nombre de données utilisées par le datamining en écartant les moins importantes. Dans la majorité des cas, le pré-traitement doit préparer des informations globales sur les données pour les étapes qui suivent tel que la tendance centrale des données (moyenne, médiane, mode), le maximum et le minimum, le rang, les quartiles, la variance, ... etc. Plusieurs techniques de visualisation des données telles que les courbes, les diagrammes, les graphes,... etc, peuvent aider à la sélection et le nettoyage des données. Une fois les données collectées, nettoyées et prétraitées on les appelle entrepôt de données (data warehouse).

4. Estimation du modèle : Dans cette étape, on doit choisir la bonne technique pour extraire les connaissances (exploration) des données. Des techniques telles que les réseaux de neurones, les arbres de décision, les réseaux bayésiens, le clustering, ... sont utilisées. Généralement, l'implémentation se base sur plusieurs de ces techniques, puis on choisit le bon résultat. Dans le reste de ce rapport on va détailler les différentes techniques utilisées dans l'exploration des données et l'estimation du modèle.

5. Interprétation du modèle et établissement des conclusions : généralement, l'objectif du datamining est d'aider à la prise de décision en fournissant des modèles compréhensibles aux utilisateurs. En effet, les utilisateurs ne demandent pas des pages et des pages de chiffres, mais des interprétations des modèles obtenus. Les expériences montrent que les modèles simples sont plus compréhensibles mais moins précis, alors que ceux complexes sont plus précis mais difficiles à interpréter.

1.4 Méthodes de datamining

1.4.1 Méthodes descriptives (ou non supervisées)

Objectif : trouver des « formes » interprétables qui permettent de décrire les données sans référence à une base d'exemples. C'est donc la construction d'un modèle et la découverte de relations dans les données. [4]

- clustering (K-means, CAH), **règles d'associations**, SOM, …

1.4.2 Méthodes prédictives (ou supervisées)

Objectif : à partir d'exemples, inférer sur les données pour réaliser des prédictions. En ce basant sur un ensemble d'exemples, on infère par exemple les classes d'appartenance d'autres individus. Les classes sont donc ici connues. [4]

- classification, régression, k-ppv …

1.5 Les tâches effectuées par le datamining

1.5.1 La description

L'importance de cette tâche est de permettre à l'analyste d'interpréter les résultats d'un modèle de datamining, soit d'un algorithme, de manière la plus transparente et efficace possible.

« Ainsi, les résultats du modèle de datamining doivent décrire des caractéristiques claires qui puissent amener à une interprétation et à une explication intuitive.

Certaines méthodes de datamining sont plus adaptées que d'autres pour une interprétation transparente ». [1]

1.5.2 L'estimation

La variable cible est numérique. L'estimation permet par exemple d'estimer la somme d'argent qu'un couple va dépenser pour des produits solaires cet été ou encore d'estimer le nombre de buts que va marquer l'équipe de Suisse lors de l'Euro.

On pourra également estimer le revenu d'une famille en fonction de divers critères (profession, catégorie de voiture...).

L'intérêt principal de cette tâche est de pouvoir ordonner les résultats afin d'avoir la possibilité de retenir seulement les meilleurs valeurs, technique qui sera surtout utilisée

en marketing dans le but de pouvoir proposer des offres aux meilleurs clients potentiels de l'entreprise.

On peut définir ici la variable cible comme étant celle que l'on cherche, à l'aide des modèles de datamining, à expliquer ou prédire la valeur.

1.5.3 La segmentation

Ici, la variable cible n'est pas numérique, mais catégorielle, comme par exemple le revenu, qui peut être divisé en trois catégories : faible revenu, revenu moyen et revenu élevé.

« La segmentation consiste à répartir les clients en groupes homogènes, qu'il convient ensuite d'aborder par des moyens spécifiques et adaptés aux caractéristiques et attentes de chaque groupe. Les membres d'un même groupe réagissent de la même manière aux stimuli marketing. Ils ont en commun un mode de communication, des comportements d'achat et/ou des besoins spécifiques » [5].

1.5.4 La classification

La classification est différente de la segmentation en ce sens qu'il n'y a pas de variable cible pour segmenter. La classification va s'intéresser au regroupement de données ou d'observations en groupes d'objets similaires. En d'autres termes, elle va segmenter l'ensemble des données afin de former des sous-groupes homogènes.

Ceux-ci s'appellent des clusters, à savoir des classes, qui sont des groupes dans lesquels les données sont similaires entre elles et, par définition différente des autres groupes.

1.5.5 La prévision

Les résultats de la prévision portent, comme son nom l'indique, sur le futur, ce qui la différencie de l'estimation. Pour le reste, elle est similaire à ces deux tâches expliquées précédemment.

La prévision permet par exemple de prévoir quels vont être les gagnants du championnat de basket-ball en tenant compte de la comparaison des résultats de chaque équipe ou encore de prévoir quel va être la taux de décroissance des décès sur la route l'année suivante en prenant compte de l'augmentation des limitations de vitesse et des mesures de réprimande plus strictes adoptées par la police en ce qui concerne l'alcool au volant.

1.5.6 L'association

Cette fonction du datamining permet de découvrir quelles variables vont ensemble, quelles sont les règles qui vont permettre de quantifier les relations entre deux ou plusieurs variables.

Par exemple, si l'on s'intéresse à 500 clients qui viennent faire leurs courses au supermarché le vendredi soir et que l'on constate que sur ces 500 clients, 100 achètent des fruits et que sur ce nombre, 30 achètent du lait, ainsi, la règle d'association est « si l'on achète des fruits, alors on achète du lait », avec une mesure de support de 100/500 = 20% et un seuil de confiance de 30/100 = 33%. [2]

1.6 Les domaines d'applications

Parmi les domaines concernés par le Datamining, c'est probablement celui de la fidélisation de clientèle qui mobilisera le plus d'attention dans les prochaines années ; en effet, il est aujourd'hui surprenant de voir avec quelle aisance, un bon nombre d'entreprises évoquent le montant des ventes pour chacun de leurs produits, alors qu'en même temps, elles sont le plus souvent incapables d'identifier les mêmes informations pour leurs principaux clients. Plus grave encore, un grand nombre d'entreprises sont incapables aujourd'hui de connaître des informations aussi basiques que le nombre de leurs clients.

Dans l'ère industrielle où il est souvent impossible d'associer à chaque client un "chargé de compte", la mémoire humaine de ceux-ci laisse désormais la place à une mémoire d'entreprise, capable de mémoriser les habitudes de consommation de chacun de ses clients, son profil, son potentiel d'achat, ou encore ses demandes fréquentes.

Le Datamining est utilisé dans beaucoup de domaines, principalement dans:

L'analyse de risque, le marketing direct, la grande distribution, la gestion des stocks, la maintenance, le contrôle de qualité, le domaine médical, l'analyse financière, en télécommunication, industrie, administration …etc.

1.6.1 Exemple

Lorsque nous passons à la caisse d'un supermarché, il est difficile de ne pas entendre les « bip, bip » des scanners lisant les codes barres sur les produits. Mais à quoi servent ces données enregistrées ?

Chacune d'entre elles, c'est-à-dire chaque « bip » représente une nouvelle «observation», une nouvelle information concernant nos habitudes d'achat. Toutes ces informations sont collectées et stockées dans des bases de données destinées à l'analyse

et vont permettre ensuite à l'entreprise d'identifier, par exemple, les mauvais clients ou alors, au contraire, de lancer une campagne de fidélisation en offrant des produits à ses meilleurs clients.

A l'image des scanners des caisses qui enregistrent quotidiennement des milliers et des milliers de transactions, l'augmentation de la production de données représente un des facteurs déterminants de la montée en puissance du datamining. D'autres causes alimentent ce phénomène, comme par exemple, « la pression concurrentielle pour accroître la part de marché dans un marché mondialisé, le développement de logiciels de datamining prêts à l'emploi, l'énorme croissance de la puissance informatique et de la capacité de stockage » et, enfin, « le stockage des informations dans les entrepôts de données qui permettent à toute l'entreprise d'accéder à une base de données fiable ».

1.7 Apport du datamining distribué

Malgré le gain apporté par la plupart des algorithmes séquentiels, ces algorithmes voient leurs performances se dégradé lorsque la taille des données augmente, surtout en matière de temps d'exécution et d'espace mémoire. Pour faire face à ces problèmes plusieurs solutions ont été proposées, parmi elles le datamining distribué.

Le datamining distribué c'est le processus du datamining classique qui consiste à extraire une information non triviale « nouvelle connaissance» à partir de sources de données distribuées, avec le minimum d'interaction entre les sites de données. En plus de cet aspect distribué qu'il faut prendre en considération, cette démarche a hérité aussi du datamining classique deux défis majeurs : augmenter le taux de précision et diminuer le temps de calcul.

Ce processus a été utilisé en générale dans deux perspectives différentes. La première perspective est la fouille de données intrinsèquement distribuées où les données doivent être fouillées dans leur site, à cause de plusieurs contraintes comme le coût de stockage, de communication, de calcul et sécurité. La deuxième est le besoin de préserver l'efficacité des algorithmes en présence d'un très grand volume de données, dans ce cas, les données peuvent être partitionnées et distribuées sur différents sites, afin d'effectuer le processus du datamining simultanément, et sur des parties de données moins volumineuse.

1.8 Architectures parallèles

Du fait de la grande quantité de données à traiter, le recours au parallélisme s'avère indispensable. Ainsi des méthodes de calcul haute performance pour le Datamining sont proposées. [6]

On pourra ainsi considérer deux types de traitements parallèles :

• le parallélisme de type SMP/CCNUMA ;

• le parallélisme distribué pour l'utilisation de réseaux de stations, de grappes de PC ou à plus grande échelle de grilles de calcul (Grid Computing).

Ainsi les spécificités de ces types de support doivent être prises en considération.

1.8.1 Parallélisme de type SMP/CCNUMA

Les principes de base des architectures de type SMP ou CCNUMA sont:

• l'utilisation simultanée de plusieurs processeurs ;

• l'existence d'une mémoire commune ou partagée par les processeurs (SMP : Shared Memory Processors, CCNUMA : Cache Coherent Non Uniform Memory Access) ;

• la disponibilité d'un réseau d'interconnexion interne rapide.

• l'homogénéité d'un système d'exploitation unique.

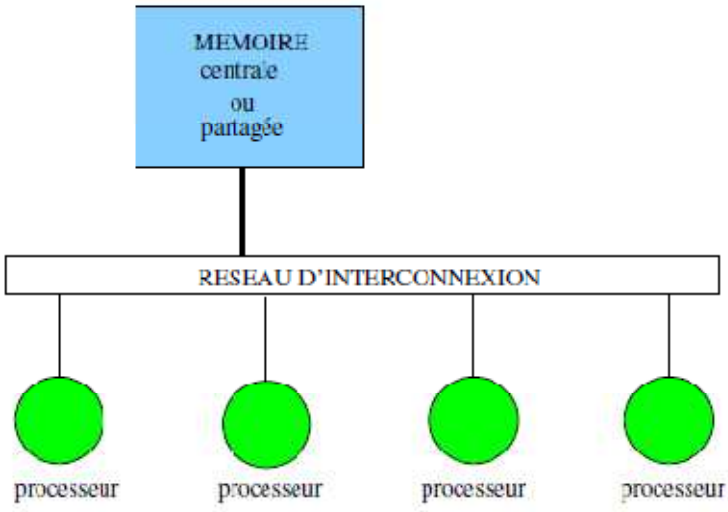


Figure 3 Architecture SMP [6]

Le problème des supercalculateurs de ce type réside dans leur coût, les rendant indisponibles en dehors de sociétés ou de centres spécifiques.

1.8.2 Parallélisme sur réseaux de stations et grille de calcul

Les principes de base de ce type de support sont :

• l'utilisation simultanée de plusieurs stations de travail ;

• l'absence de mémoire commune ;

• la présence d'un réseau d'interconnexion lent par rapport à la puissance des machines (type Internet).

• l'hétérogénéité d'un système composé d'une grappe de stations ou d'une grappe de grappes.

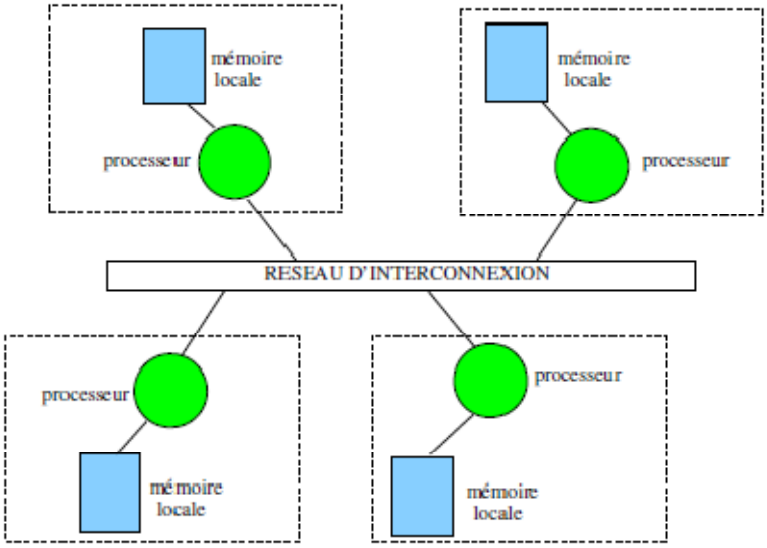


Figure 4 Architecture de type réseau de stations et grille de calcul [6]

Ces plateformes d'exécution proviennent de la mise en commun de ressources de traitement et de stockage disponibles dans le réseau et s'avèrent plus abordables financièrement (disponibilité d'un réseau de stations de travail sous-exploitées au sein d'une entreprise de taille moyenne).

Les utilisateurs se partagent les ressources du réseau (ou de la la grille) et chacun d'entre eux peut accèder à la totalité des ressources. C'est en fait une "globalisation virtuelle" d'infrastructures informatiques qui peut être composée de tout type d'unités de traitement et de stockage.

La distinction entre réseau de stations et grille réside dans la taille de l'infrastructure :

• un réseau de stations consiste à mettre en commun des stations de travail ou PC (exemple : les stations disponibles au sein d'une entreprise, d'un laboratoire d'université, d'une salle de travaux pratiques).

On désigne par ce terme des grappes de stations spécialisées avec un middleware disponible ou un réseau homogène basique.

• à plus grande échelle, une grille de calcul permet de mettre en commun des réseaux de stations et éventuellement des supercalculateurs en une hiérarchie de clusters (grappe de grappes). Chaque sous-réseau de stations peut être appelé sous-grappe.

Le développement de logiciels d'exploitation de plateformes de type réseau de stations de travail ou grille de calcul est un des challenges de la recherche actuelle.

1.9 Partitionnement de la base de données

Les solutions parallèles existantes utilisent l'existence d'une mémoire commune (ou partagée) avec accès rapide aux données. Dès lors ces solutions ne peuvent être utilisées dans un contexte de réseau de stations ou de grille puisqu'elles sont développées pour une exécution sur un système homogène.

Dans un système hétérogène de type grille de calcul, la base de données peut être :

• centralisée ;

• distribuée par attributs (fragmentation verticale) : multi-base

• distribuée par instances (fragmentation horizontale) : base distribuée

Figure 5 Possibilité de distribution de la base de données [6]

Dans ce type d'infrastructure réseau, on ne dispose pas de mémoire centrale.

Pour assurer un traitement optimal, il faudra s'assurer de distribuer les données de manière adéquate pour permettre la distribution du traitement et l'exploitation des ressources de stockage locales.

Une fragmentation de la structure de données à exploiter pré-existe ou s'avère indispensable.

Une fragmentation existante peut nécessiter une adaptation : base distribuée au départ et nécessité d'un contexte multibase, ou multibase au départ et nécessité d'une base distribuée.

Une fois les données distribuées dans les mémoires locales et du fait de la non-existence d'une mémoire commune, le traitement devra s'effectuer sur les vues

partielles disponibles localement sur chaque noeud, nécessitant ainsi des communications dans le schéma de résolution.

Le réseau utilisé dans une plateforme d'exécution distribuée étant relativement lent par rapport à la puissance des machines, l'efficacité du parallélisme distribué pourra être amélioré par :

• un recouvrement des temps de communications par l'exécution des instructions CPU pour pallier à la lenteur de ces communications ;

• l'utilisation de l'asynchronisme de manière à utiliser les temps d'attente inactifs (pour une exploitation maximale de la puissance de calcul) et à éviter les temps d'attente inactive (lors de la synchronisation des tâches) ;

• l'utilisation d'un principe de pipeline (data-flow par disponibilité) pour anticiper des calculs et ainsi exploiter la puissance de calcul.

On prendra ainsi en compte les contraintes du support d'exécution:

• exploiter au maximum la puissance de calcul et les capacités de stockage disponibles;

• utiliser au maximum le parallélisme aux différents niveaux du schéma de résolution;

• effectuer des traitements parallèles les plus indépendants possibles ;

• limiter les communications et interdire les phases de synchronisations (du fait de la lenteur et de la non-fiabilité du réseau) ;

• exploiter au maximum les approches pipeline, le recouvrement des communications par des instructions CPU.

• l'utilisation de Java RMI permettra d'assurer l'interopérabilité entre les nœuds de la grappe ou de la grille.

Les trois éléments majeurs à prendre en compte sont :

1. la distribution des données ;

2. la distribution des traitements ;

3. les collaborations entre les traitements.

Les points 1 et 2 visent à assurer une adéquation entre la distribution des données et des traitements. Le point 3 devra exploiter le recours à l'asynchronisme.

1.10 Techniques du datamining distribué

A partir de la littérature liée au datamining distribué, on peut repérer deux techniques utilisées: la technique du parallélisme [7] qui fait souvent appel à des machines dédiées et des outils de communication entre les processus parallèles, et une autre technique que nous appelons « technique d'agrégation »[8], qui procède avec une vision purement distribuée, soit sur les données, soit sur les prédictions ou bien sur les modèles de base.

1.10.1 Techniques Parallèles

Ces techniques se basent sur l'extraction de l'aspect parallèle de l'algorithme utilisé, permettant ainsi d'exécuter plusieurs parties de l'algorithme en parallèle, sur différents processeurs. Ceci nécessite en générale des architectures dédiées comme les systèmes à mémoire partagée. Les grilles de calcul représentent actuellement une alternative intéressante comme plate forme du calcul parallèle.

Le parallélisme des algorithmes du datamining peut être conçu selon deux approches principales, selon qu'on parallélise les données ou les tâches. Le parallélisme par données correspond à diviser les données sur P processeurs, chaque processeur exécute un même traitement sur sa portion locale des données. Dans le parallélisme des tâches, les processeurs accomplissent des traitements différents indépendamment les uns des autres. En pratique, ces techniques nécessitent une intention particulière sur deux facteurs essentiels : l'équilibrage de charge entre les différents processeurs utilisés dans le calcul parallèle, et la charge de communication sur le réseau. Cette communication est souvent nécessaire pour l'échange de données ou de résultats partiels entre les processeurs.

Les techniques des arbres de décision sont très favorables au parallélisme, à cause de leur nature « diviser et régner». Elles ont fait l'objet de plusieurs travaux, comme SLIQ parallèle, SPRINT Parallèle et RainForest. Ces systèmes tentent d'optimiser le temps de calcul par l'utilisation de structures de données particulières, et le recours au parallélisme.

1.10.2 Techniques d'agrégation

D'autres techniques -que nous appelons « agrégation», sont utilisées dans le but de fouiller des données issues de plusieurs bases distribuées. Elles peuvent être classées

selon que l'algorithme du datamining utilise une partie des données disponibles, dans ce cas on parle d'Agrégation de données-, ou la totalité des données, qui correspond soit à l'agrégation de modèles ou l'agrégation de prédictions :

Agrégation de données : L'échantillonnage consiste en la création d'un échantillon représentatif d'une large base de données sous l'hypothèse qu'un algorithme de datamining entraîné sur cet échantillon n'aura pas de résultats significativement pires qu'un algorithme entraîné sur toute la base de données. Dans le contexte du datamining distribué, l'échantillonnage est appliqué sur chaque base répartie, générant des échantillons distincts dans chaque site. Ces derniers sont regroupés afin d'entraîner un seul algorithme.

L'algorithme d'apprentissage est appliqué sur une partie des données distribuées, qui peut être soit des échantillons issus de chaque base, ou bien quelques bases bien choisies. L'inconvénient de l'agrégation de données est le temps de transfert ainsi que le temps de traitement pour choisir les bons échantillons.

Agrégation de modèles : Dans le cas de classification par exemple, des règles de classification sont construites, ensuite regroupées afin de produire un ensemble de règles unique. Bien qu'elles utilisent la totalité des données, ces techniques peuvent être appliquées sur des échantillons de données bien choisis. On cite comme exemple de systèmes d'agrégation de modèles le travail de Hall et al.; celui ci produit un ensemble de règles de classification en parallèle à partir d'ensembles disjoints de données, ensuite les règles sont unies en un seul système. Ces auteurs proposent un ensemble de techniques de résolution de conflits entre les règles produites. Cette même approche a été utilisée dans, mais avec des algorithmes différents qui sont appliqués localement pour produire les règles.

Agrégation de prédiction : Les techniques d'agrégation de prédictions consistent à construire un ensemble de classificateurs de base –dans le cas de la tâche de classification- dont les prédictions seront combinées afin de classer de nouvelles données dont la classe est inconnue. Ces techniques appelées Méthodes d'apprentissage ensembliste (ensemble learning) ont été conçues au début pour être appliquées sur un seul ensemble de données, dans le but d'améliorer les performances de prédiction.

Malgré que ces techniques ne produisent que des modèles prédictifs qui ne peuvent expliquer le choix de leurs classifications à un analyste, elles présentent une alternative prometteuse pour le datamining distribué. Ceci est dû aux facteurs suivants :

- Leur apport en termes d'augmentation de la précision,
- L'aspect distribué qu'elles offrent,
- L'élimination de la phase de construction d'un classificateur global, comme dans le cas d'agrégation de modèles, pourrait accélérer le processus du datamining.

1.11 Conclusion

Dans ce chapitre nous avons étudié le datamining en général qui est un domaine d'actualité, nous allons le voir en détails dans le chapitre qui suit dans un environnement séquentiel.

2 Processus d'extraction des connaissances dans un environnement séquentiel

Chapitre II : Processus d'extraction des connaissances dans un environnement séquentiel

2.1 Introduction

La découverte des connaissances incluses dans les données, et tout particulièrement les techniques de datamining sont de plus en plus utilisées. Le but de ces techniques d'analyse est d'extraire des informations utiles à partir de grandes bases de données.

Afin d'extraire des connaissances des données en entrée organisées sous forme d'une base de transactions, deux étapes son nécessaires à savoir, l'extraction des ensembles d'itemsets fréquents qui est l'étape la plus coûteuse en traitements et la génération des règles d'association a partir de ces ensembles, par conséquent la plupart des recherches sont consacrés à l'extraction des itemsets fréquents, Cependant, cette approche souffre de deux problèmes majeurs, a savoir le coût de l'extraction des itemsets fréquents a partir desquels seront extraites de grandes quantités de règles d'association. Cette quantité énorme de connaissance rend quasi impossible leur analyse par un expert humain.

2.2 Extraction des connaissances

L'extraction des connaissances est un processus itératif et interactif. Ce processus peut se décomposer en quatre phases qui sont représenté dans la figure 6. Les connaissances de l'utilisateur concernant le domaine d'application sont nécessaires lors des phases de prétraitement, afin d'assister la sélection et la préparation des données, et de post-traitement, pour l'interprétation et l'évaluation des règles extraites. En fonction de l'évaluation des règles extraites, les paramètres utilisés lors des précédentes phases (critère de sélection et préparation des données et seuils minimaux de support et de confiance) peuvent être modifiés avant d'effectuer à nouveau l'extraction des connaissances, ceci afin d'améliorer la qualité du résultat.

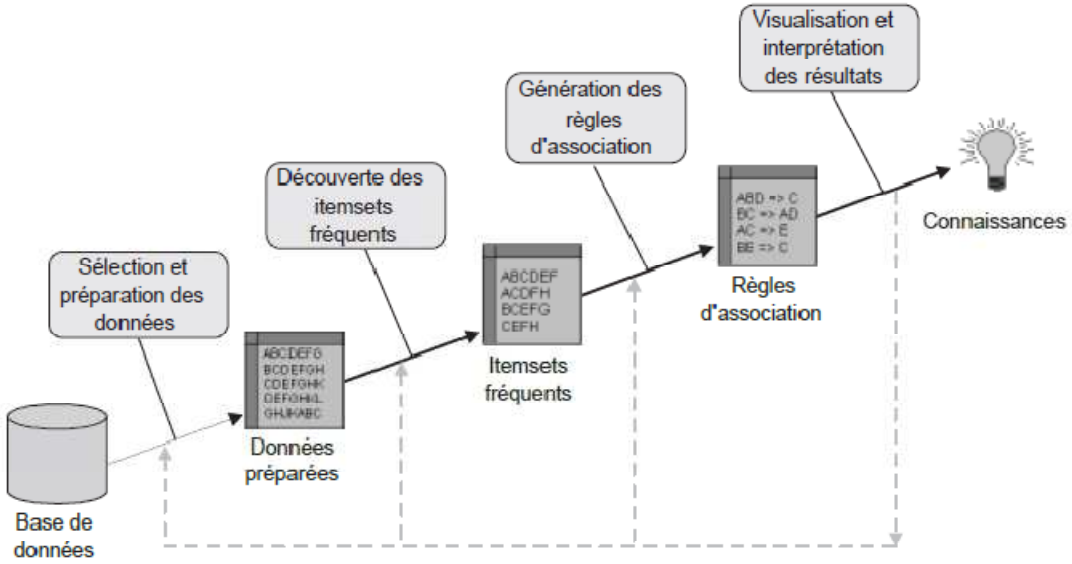


Figure 6 Etape du processus d'extraction des connaissances [9]

2.3 Extraction des itemset fréquent

La recherche des itemsets fréquents est un problème non trivial car le nombre d'itemsets potentiellement fréquents est exponentiel par rapport au nombre d'items considérés dans la base de données.

La phase de recherche de ces itemsets fréquents est la phase la plus coûteuse de l'extraction de règles d'association du fait de la taille exponentielle de l'espace de recherche et du nombre élevé nécessaire de balayages complets du jeu de données.

Ces balayages sont nécessaires afin d'évaluer les supports des itemsets (qui permettent de déterminer lesquels sont fréquents) puis de calculer les confiances des règles d'association, mais constituent des opérations très coûteuses en temps d'exécution.

Soit I, l'ensemble des items, et minsup un seuil minimal de support.

L'ensemble F des itemsets possibles est :

F = {i inclut I | i ≠ Ø ; et support(i) ≥ minsup}

Si Card(I) = M, le nombre d'itemsets possibles est 2^M.

2.3.1 Notions de base

Partant d'une base de transaction, nous souhaitons obtenir des corrélations de présences d'objets différents dans les transactions. Étant donné la taille des bases à analyser, on limite en général la recherche des règles d'associations aux objets les plus courants de la base. On entend par "courant" les objets ou ensembles d'objets qui répondent à un critère minimum de présence choisi par l'utilisateur. Ce critère appelé "support minimum" est le taux de présence minimum exigé pour prendre en compte l'objet ou l'ensemble d'objets dans la recherche de motifs. Une règle associative, écrite sous la forme A => B désigne une relation entre des conjonctions d'attributs A et B, pondérée par une valeur de "confiance". Dans cette expression, A désigne la prémisse de la règle et B sa conclusion.

Dans les traitements classiques de bases de données, on s'intéresse à des bases de données B composées d'un ensemble d'instances (ou enregistrements) et d'attributs en général continus.

Chaque instance d'une base B associe une valeur à chacun des attributs continus (et est donc un ensemble de valeurs continues).

Dans le problème de recherche de règles d'association, on suppose la base de données discrétisée de manière à travailler sur une base D composée d'un ensemble d'instances et d'un ensemble d'attributs discrets (ou nominaux) appelés **items.**

On appelle **itemset** un ensemble d'items; le nombre d'items d'un itemset constitue sa longueur (un itemset contenant k items est appelé un k-itemset).

Chaque instance de la base de données est un itemset.

La recherche de règles s'appuie sur des mesures statistiques qui permettent de générer les règles les plus pertinentes et de limiter l'espace de recherche :

• A chaque itemset, on associe une mesure appelée support : le support d'un itemset est le pourcentage d'instances de la base de données qui contiennent l'itemset.
(le support permet de mesurer l'intérêt de l'itemset, sa fréquence dans le jeu de données).

La règle X => Y a un support « s » dans l'ensemble des transactions D si s% des transactions de D contiennent $\mathbf{X} \cup \mathbf{Y}$. [10]

$$\mathbf{Sup\ (X \Rightarrow Y) = Sup\ (X \cup Y) = P\ (X, Y) = P\ (X \cup Y) / |D|}$$

- **Propriété des itemsets**

Propriété 1 : Tout sous-ensemble d'un ensemble fréquent est fréquent.

Cette propriété permet une limitation du nombre de candidats générés lors de la kème itération par une jointure conditionnelle des itemsets fréquents, de taille k-1, découverts précédemment.

Propriété 2 : Tout sur-ensemble d'un ensemble infréquent est infréquent.

Cette propriété permet la suppression d'un candidat de taille k lorsqu'au moins un de ses sous-ensembles de taille k-1 ne fait pas partie des itemsets fréquents découverts précédemment.

Propriété 3 : Pour qu'un itemset soit globalement fréquent il faut qu'il soit localement fréquent sur au moins un site.

Quel que soit le principe de l'algorithme parallèle choisi, il nécessite de nombreuses communications pour transiter tantôt les itemsets et leurs informations, tantôt les données, la plupart de ces communications devant se faire de manière plus ou moins synchronisée.

2.3.2 Exemple

F = {A, B, C, D, E}

M = 5

Le nombre d'itemsets possibles est 2^5.

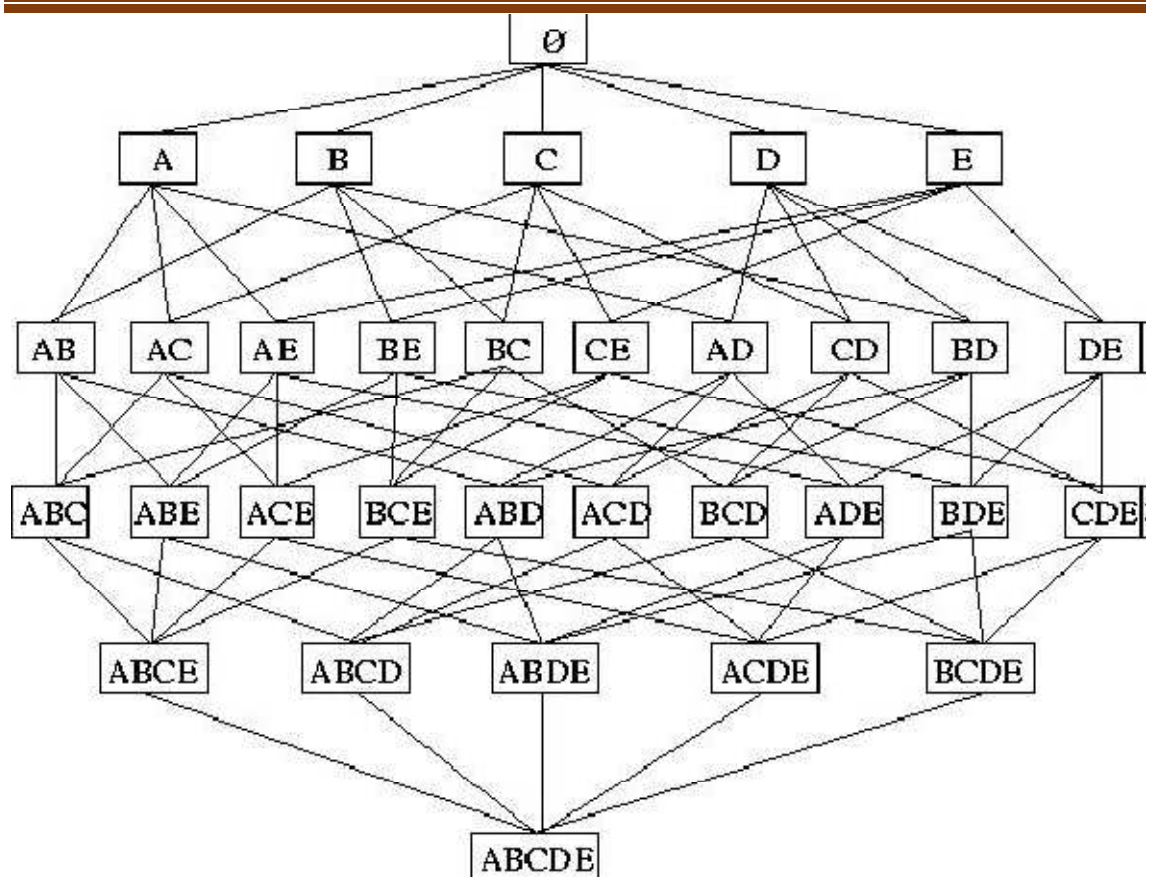


Figure 7 Treillis d'itemsets [9]

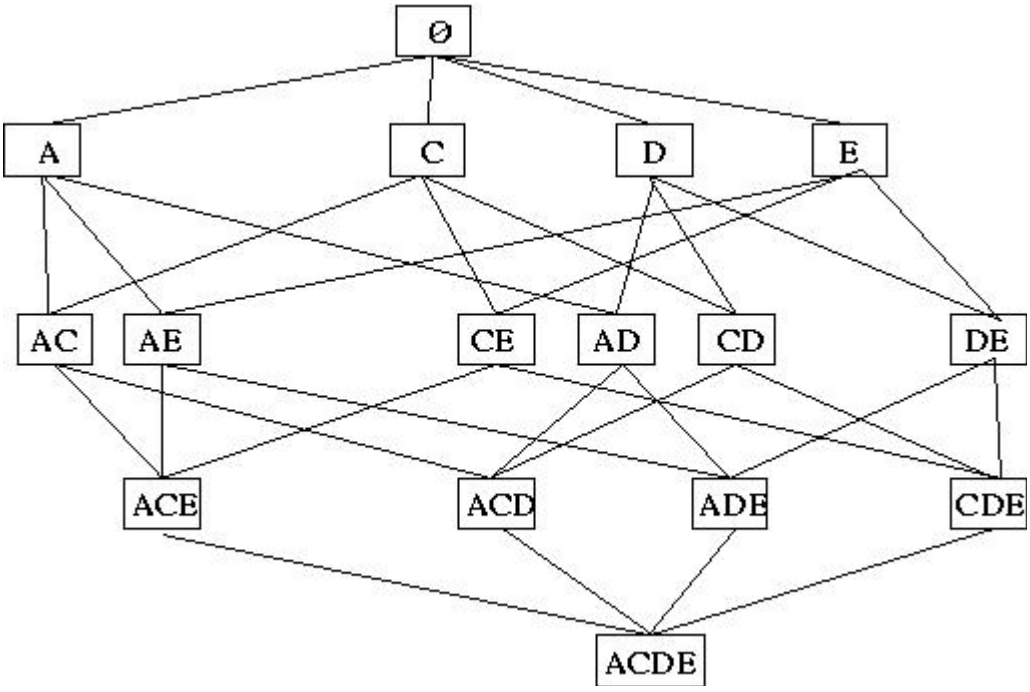


Figure 8 Treillis d'itemsets fréquents

2.4 Génération des règles d'associations

Pour chacun des itemsets fréquents obtenus, on va rechercher les règles d'association qui possèdent une confiance supérieure au seuil minimal de confiance fourni par l'utilisateur.

On regarde les différentes combinaisons condition-conclusion possibles pour chaque itemset.

On garde la combinaison la plus pertinente (celle de confiance la plus élevée), si elle répond au critère de seuil minimal de confiance.

2.4.1 Notions de base

La recherche de règles s'appuie sur des mesures statistiques qui permettent de générer les règles les plus pertinentes et de limiter l'espace de recherche :

• A chaque règle d'association, on associe une mesure appelée confiance :

(la confiance permet de mesurer une réelle causalité entre la condition et la conclusion).

La règle X => Y se reporte à l'ensemble de transactions avec une confiance (confidence) « c » si c% des transactions D qui contiennent l'ensemble des objets X contiennent aussi l'ensemble des objets Y. [10]

$$\mathbf{Conf\ (X => Y) = Sup\ (X \cup Y) / Sup\ (X)}$$

2.4.2 Obtenir une règle d'association à partir d'un motif

Pour trouver les règles d'associations, on commence par chercher des motifs fréquents dans la base de données. On considère un motif comme fréquent à partir d'une présence minimum dans la base (appelée support minimum) donné par l'utilisateur. Ainsi, si le support minimum est fixé à 10 %, les ensembles d'items apparaissant dans moins de 10 % des transactions sont éliminés des motifs fréquents. A partir de ces

motifs et leurs supports, on calcule la valeur de confiance des règles que l'on peut en déduire.

Si par exemple le motif ABC est retenu comme motif fréquent, nous pourrons en tirer les règles :

AB → C

AC → B

BC → A

A → BC

B → AC

C → AB

Pour calculer la valeur de confiance de la première règle, on effectue le calcul support(ABC) / support(AB) ce qui correspond en fait à la probabilité conditionnelle de rencontrer C dans une transaction sachant que celle-ci contient A et B. Le résultat est bien sûr en général différent pour chacune des règles déduites d'un motif. Ainsi, l'utilisateur peut encore filtrer les résultats obtenus sur la valeur de confiance d'une règle afin d'éliminer celles qui ne sont pas réellement significatives.

S'il s'agit de trouver tous les itemsets de la base afin de découvrir toutes les règles d'associations valables, il suffit de choisir pour support minimum 1. Ainsi, chaque itemset apparaissant dans la base est considéré comme fréquent. Le principal problème pour trouver des règles d'associations consiste en la découverte des itemsets fréquents. En effet, ceux-ci sont potentiellement au nombre de 2m pour une base contenant m items. Le choix du support minimum dépend énormément le nombre d'itemsets fréquents, mais pas le nombre de règles d'associations valables. On peut, sur une base de taille importante, ne jamais trouver de corrélation entre la présence d'un item et la présence d'un autre même si les items choisis ont une présence importante dans la base. Alors même que la présence d'un item par ailleurs très peu représenté dans la base peut entrainer la présence d'un autre item. Par exemple :

– De nombreuses personnes achètent du pain et de nombreuses personnes achètent des piles (support important), cependant, on trouve une valeur de confiance peu élevée pour la règle (Pain) => (Piles).

– Peu de personnes achètent un appareil photo numérique, peu de personnes achètent des cartes mémoires (support faible), mais la valeur de confiance de la règle (Appareil photo numérique) => (Carte mémoire) est élevée.

La recherche de règles d'associations valables nécessite donc en général l'emploi d'une valeur de support minimum faible pour ignorer un minimum de règles intéressantes. Or, plus le support minimum choisi est faible, plus les itemsets fréquents sont nombreux et donc plus le travail à fournir est important. C'est pour ces raisons, qu'il est indispensable de bénéficier d'une puissance de calcul et d'espace mémoire important pour réaliser la tâche de rechercher les itemsets fréquents. C'est ce qui a conduit au développement d'algorithmes parallèles. Étant donné la taille des bases à traiter, un calcul sur grille semble fournir une solution adéquate à la puissance de calcul nécessaire à la réalisation de cet objectif. De plus, les bases de données dont on veut tirer les motifs fréquents sont en général de tailles très importantes et nécessitent une capacité de stockage que peut offrir le calcul sur grille.

2.5 Conclusion

Dans ce chapitre nous avons présenté les deux étapes de l'extraction des connaissances et aussi les notions de bases que nous utiliserons au chapitre suivant, avec des exemples de l'algorithme séquentiel APRIORI, qui est l'algorithme de base pour les algorithmes présenté dans ce qui suit.

3 Algorithmes distribués d’extraction des connaissances

3.1 Introduction

La technique d’extraction des connaissances est l’une des solutions les plus sollicitées de datamining. Les premiers algorithmes de recherche d’itemsets fréquents sont pour l’essentiel de nature séquentielle, qui malgré leur efficacité ne garantissent pas la scalabilité en terme de taille de données et de temps d’exécution, d’où l’intérêt d’orienter les recherches vers des environnements d’exécution parallèles et distribués. Plusieurs algorithmes parallèles et distribués ont été développés ces dernières années.

Dans ce chapitre, nous présentons un état de l’art sur un échantillon non exhaustif mais assez représentatif d’algorithmes distribués basés sur l'extraction des itemsets fréquents proposés dans la littérature.

3.2 Notations

$\boldsymbol{C_k}$: k_itemsets_condidats.

$\boldsymbol{L_k}$: k_itemsets_frequents.

$\boldsymbol{C_{k^i}}$: k_itemsets_condidats du processeur $\boldsymbol{P_i}$.

$\boldsymbol{L_{k^i}}$: k_itemsets_frequents du processeur $\boldsymbol{P_i}$.

$\boldsymbol{D_i}$: la portion de la base de transaction qui est stockée dans la mémoire centrale du processeur $\boldsymbol{P_i}$.

N : nombre de processeurs.

3.3 Count Distribution (CD)

Cet algorithme se base sur une répartition "aléatoire" des données, on effectue un découpage horizontal des données (répartition des instances sur les processeurs). Chaque processeur compte les supports des mêmes itemsets candidats.

A chaque itération, les supports locaux de chaque itemset sont calculés, puis une synchronisation permet de sommer ces supports locaux, de manière à prendre les décisions sur des supports globaux. Cette synchronisation de supports a lieu entre chaque itération.

Partitionnement horizontal (n transactions, m items) et répartition sur P processeurs.

Pour k =1, chaque site

Génère l'ensemble des candidats C_1 de taille 1 (le même ensemble dans chaque site).

Calcule le support local de l'ensemble des candidats C_1.

Diffuse les supports locaux aux autres sites.

Détermine l'ensemble des fréquents L_1.

Répéter

k++;

Générer l'ensemble des candidats C_k à partir de L_{k-1}.

Calculer les supports locaux des candidats C_k (par un scan de la partition locale).

Diffuser les supports locaux aux autres sites.

Déterminer l'ensemble des fréquents de taille k, noté L_{k+1}.

Jusqu'à ce que tous les itemset fréquents soient trouvés.

3.3.1 Les etapes de l'algorithme CD

- **Etape préliminaire**

Au cours de cette étape on génère l'ensemble des items fréquents de taille 1 (1 itemset_frequents).

On distingue trois sous étapes exécutées par tous les processeurs :

1. Parcourir la portion locale de la base de données et déterminer les différents items qui y figurent avec leur nombre d'apparition.

2. Transmettre le résultat aux autres processeurs.

3. Faire la mise à jour du résultat localement calculé.

A la fin, chaque processeur possède donc exactement le même ensemble des 1-itemset_frequents.

Cette étape est exécutée une seule fois, c'est une phase d'initialisation de l'algorithme.

- **Première étape**

Chaque processeur génère l'ensemble des k_itemset_condidats $\boldsymbol{C_k}$ à partir de l'ensemble des (k-1)-itemset_frequents $\boldsymbol{L_{k-1}}$ calculé à la dernière étape de la passe précédente. On note que, tous les processeurs génèrent le même ensemble $\boldsymbol{C_k}$ puisque ils possèdent tous le même $\boldsymbol{L_{k-1}}$.

- **Deuxième étape**

Chaque processeur Pi parcoure sa portion locale de base de données et calcule le support des items candidats générés dans la première étape.

- **Troisième étape**

Echange des supports calculés à l'étape précédente entre les différents processeurs.

- **Quatrième étape**

Chaque processeur génère l'ensemble $\boldsymbol{L_k}$ à partir de $\boldsymbol{C_k}$.

- **Cinquième étape**

Chaque processeur, indépendamment des autres, décide de terminer ou continuer vers la prochaine passe. On note que les décisions seront identiques puisqu'ils ont le même $\boldsymbol{L_k}$.

Remarque

La sauvegarde des 1-itemsets_frequents se fait dans un ordre croissant afin d'optimiser le traitement au cours des étapes suivantes.

3.3.2 Exemple de déroulement

Pour un support minimum = 2.

Tid	Items
1	Fdbe
2	Feb
3	Adb
4	Aefc
5	Ade
6	Acfe

Base de données

Tid	Items	Processeurs
1	fdbe	P0
2	feb	
3	adb	P1
4	aefc	
5	ade	P2
6	acfe	

Partitionnement de la base de données

Item	Support local			Support global
	P0	P1	P2	
A	0	2	2	4
b	2	1	0	3
c	0	1	1	2
d	1	1	1	3
e	2	1	2	5
f	2	1	1	4

1-itemsets fréquent

Chaque item a un support supérieur au support minimum.

2-Itemset	Support local			Support global
	P0	P1	P2	
ab	0	1	0	1
ac	0	1	1	2
ad	0	1	1	2
ae	0	1	2	3
af	0	1	1	2
bc	0	0	0	0
bd	1	1	0	2
be	2	0	0	2
bf	2	0	0	2

cd	0	0	0	0
ce	0	1	1	2
cf	0	1	1	2
de	1	0	1	2
df	1	0	0	1
ef	2	1	1	4

2-itemsets fréquent

A l'itération suivante, on ne prend pas en considération les 3-itemsets contenant : ab, bc, cd, df.

Item	**Support local**			**Support global**
	P0	**P1**	**P2**	
ace	0	1	1	2
acf	0	1	1	2
ade	0	0	1	1
aef	0	1	1	2
bde	1	0	0	1
bef	2	0	0	2
cef	0	1	1	2

3-itemsets fréquent

A l'itération suivante, on ne prend pas en considération les 4-itemsets contenant: ade, bde.

Item	Support local			Support global
	P0	P1	P2	
acef	0	1	1	2

4-itemsets fréquent

3.3.3 Discussion

L'algorithme **CD,** proposé par **R.Agrawal** et **J.C.Shafer,** est une exécution parallèle de l'algorithme séquentiel **Apriori**.

Cet algorithme se base sur une répartition aléatoire des données. Les processeurs sont indépendants et il n'y a communication entre eux qu'à la fin du calcul. Ce qui cause un problème de mémoire.

Cet algorithme ne divise pas l'ensemble des candidats ce qui constitue une redondance de calcul.

Dans le cas où le nombre de candidats est très élevé, la structure de données utilisée pour stocker l'ensemble des candidats sera par conséquent assez volumineuse et ne pourra pas être chargé en totalité dans la mémoire centrale, chaque processeur sera donc obligé de la partitionner en plusieurs parties et à chaque fois effectuer une passe correspondant à la portion qui est chargée en mémoire et effectuer par la suite une opération d'entrée/sortie pour charger une autre portion et ainsi de suite. Le temps de calcul sera alors alourdi par un temps d'entrée/sortie assez important ce qui dégrade considérablement les performances de cet algorithme qui donne de très bons résultats sur les petites bases de données où le nombre de candidats n'est pas très élevé d'autant plus qu'il n'y a presque pas de communication entre processeurs.

Le nombre de candidats augmente dans les cas où le nombre d'items distincts dans la base est élevé et/ou que le support minimal est très bas. L'algorithme CD est donc très efficace pour des bases de données dont le nombre d'items distincts est réduit, s'il est exécuté avec un support minimal assez élevé.

3.4 Data distribution (DD)

Tout comme CD, cet algorithme se base sur une répartition "aléatoire" des données, mais ici, chaque processeur génère des candidats différents.

Les supports des itemsets sont toujours calculés de façon locale, à chaque itération. Puis, entre deux itérations, on effectue une diffusion des partitions locales de données (c'est-à-dire des instances). Cette diffusion des $\boldsymbol{D_i}$ locaux s'effectue par une communication de type **all-to-all**.

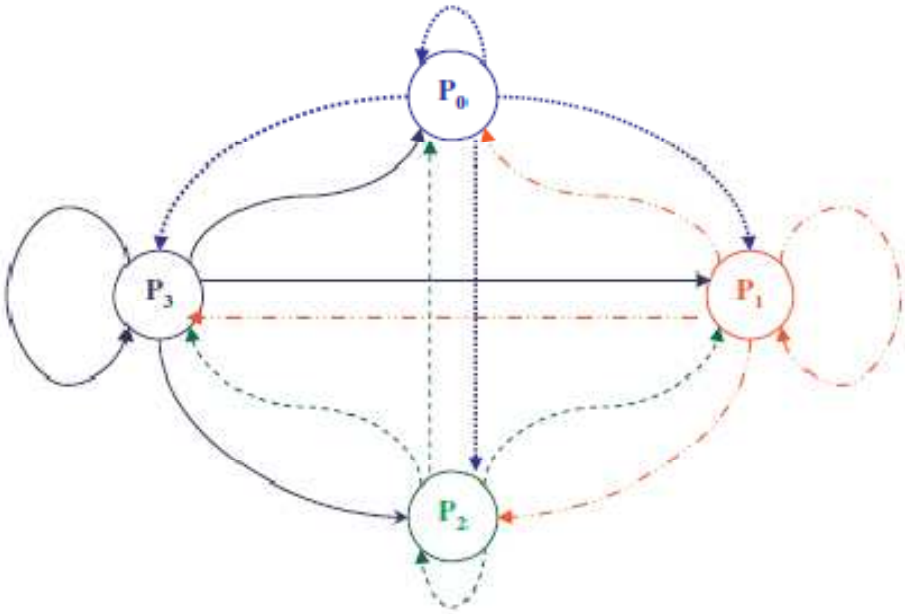


Figure 9 Communication all-to-all [10]

P0 transmet ses données à P1, P2, P3.

P1 transmet ses données à P0, P2, P3.

P2 transmet ses données à P0, P1, P3.

P3 transmet ses données à P0, P1, P2.

Partitionnement horizontal (n transactions, m items) et répartition sur P processeurs.

Pour k =1, chaque site

Détermine les candidats locaux C_1 de taille 1.

Calcule les supports des candidats locaux à partir des données locales et distantes.

Détermine les itemsets fréquents locaux L_1 de taille 1.

Diffuse l'ensemble des itemsets fréquents locaux L_1 aux autres sites.

Détermine l'ensemble global des itemsets fréquents L_{1G} .

Répéter

k++;

Générer l'ensemble des candidats C_k à partir de L_{k-1}.

Calculer les supports des candidats locaux C_k à partir des données locales et distantes.

Détermine l'ensemble local des itemsets fréquents L_k.

Diffuser l'ensemble des itemsets fréquents locaux L_k aux autres sites.

Détermine l'ensemble global des itemsets fréquents L_{kG}.

Jusqu'à ce que tous les itemsets fréquents soient

3.4.1 Les étapes de l'algorithme DD

- **Etape préliminaire**

Au cours de cette étape on génère l'ensemble des items fréquents de taille 1 (1-itemset_frequents).

On distingue trois sous étapes exécutées par tous les processeurs :

1. Parcourir la portion locale de la base de données et déterminer les différents items qui y figurent avec leur nombre d'apparition.

2. Transmettre le résultat aux autres processeurs.

3. Faire la mise à jour du résultat localement calculé.

A la fin, chaque processeur possède donc exactement le même ensemble des 1-itemset_frequents.

Cette étape est exécutée une seule fois, c'est une phase d'initialisation de l'algorithme.

- **Première étape**

Le processeur $\boldsymbol{P_i}$ génère $\boldsymbol{C_k}$ à partir de $\boldsymbol{L_{k-1}}$, ne retient que 1/N de l'ensemble précédemment formé (la partie en question est choisie grâce a l'identificateur de processeur) cette partie est noté $\boldsymbol{C_{k^i}}$ et elle sera unique.

- **Deuxième étape**

Le processeur $\boldsymbol{P_i}$ calcule le support de chaque élément de l'ensemble $\boldsymbol{C_{k^i}}$ à partir de sa base locale et des bases stockées chez les autres processeurs (cette étape est la plus importante et la plus coûteuse, elle sera un peut plus détaillée ci-dessous).

- **Troisième étape**

Le processeur $\boldsymbol{P_i}$ détermine $\boldsymbol{L_{k^i}}$ et cela en éliminant les items dont le support n'a pas atteint le minimum minsup.

- **Quatrième étape**

Le processeur $\boldsymbol{P_i}$ transmet $\boldsymbol{L_{k^i}}$ à tous les autres processeurs. Chaque processeur dispose maintenant de $\boldsymbol{L_k}$ qui servira à construire $\boldsymbol{C_k}$ à l'étape suivante.

On remarque que la deuxième étape de l'algorithme DD est la plus importante, en effet elle comporte les principaux traitements : accès à la base, calcul de support et communication.

Cette étape est aussi la plus coûteuse en temps d'exécution.

- **Détail de la deuxième étape**

Le processeur $\boldsymbol{P_i}$

1. alloue N buffers B (N : nombre de processeur)
2. charge une page de $\boldsymbol{D_i}$ dans le buffer $\boldsymbol{B_i}$
3. effectue un broad cast de $\boldsymbol{B_i}$
4. boucle jusqu'à ce que tous les buffers soient traités :

si buffer non_traité et buffer non_vide

alors traiter le buffer

sinon passer au buffer suivant

fin si

Fin de la boucle

5. Barrière de synchronisation.
6. si la base $\boldsymbol{D_i}$ n'est pas traitée en totalité alors reprendre en 3.

Fin.

Traiter un buffer $\boldsymbol{B_j}$ revient à :

Calculer le support des candidats dans ce buffer

Faire une mise à jour du compte ultérieur

Si i = j alors charger une nouvelle page de $\boldsymbol{D_i}$.

Si i ≠ j alors vider le buffer.

Chaque processeur alloue P buffers (de taille une page, un buffer pour chaque processeur). Pour le processeur Pi le iemme buffer est utilisé pour stocker les transactions de la portion locale de la base de données, les autres sont utilisés pour stocker les transactions provenant des autres processeurs. Chaque processeur Pi contrôle les P buffers pour voir lequel contient de l'information. Soit ce buffer j, le processeur calcule le nombre d'apparition de ses candidats dans cette page et met à jour le compte ultérieur. Maintenant si ce buffer correspond à celui qui stocke la portion locale (c à d que i = j), il est alors envoyé à tout les autres processeurs (via une transmission asynchrone). Si ce buffer correspond à un buffer qui stocke une transaction provenant d'un autre processeur (c à d que i ≠ j) alors son contenu est effacé et il est marqué prêt pour une autre opération de réception asynchrone provenant d'un autre processeur. Ces opérations se déroulent jusqu'à ce que tous les processeurs aient accédé à toutes les transactions de la base. A ce stade tout les processeurs ont calculé le nombre total

d'apparition de leurs candidats locaux respectifs, chacun détermine alors l'ensemble des items fréquents à partir de l'ensemble des candidats. Cet ensemble est alors transmis à tous les autres processeurs par une opération de diffusion générale (all-to-all broadcast).

3.4.2 Exemple de déroulement

Pour un support minimum = 2.

Tid	**Items**
1	fdbe
2	feb
3	adb
4	aefc
5	ade
6	acfe

Base de données

Partitionnement horizontal de la base et répartition sur les processeurs.

P0

Tid	**Items**
1	fdbe
2	feb

C_1	**supp**
a	4
b	3

P1

Tid	**Items**
3	adb
4	aefc

C_1	**supp**
c	2
d	3

P2

Tid	**Items**
5	Ade
6	acfe

C_1	**Supp**
e	5
f	4

L_1	a	b
supp	4	3

L_1	c	d
supp	2	3

L_1	e	F
supp	5	4

L_{1G}	a	b	c	d	e	f
supp	4	3	2	3	5	4

1-itemsets fréquents

Chaque item a un support supérieur au support minimum, on pas a la 2ème itération.

P0

C_2	supp
ab	1
ac	2
ad	2
ae	3
af	2

P1

C_2	supp
bc	0
bd	2
be	2
bf	2
cd	0

P2

C_2	Supp
ce	2
cf	2
de	2
df	1
ef	4

L_2	ab	ac	ad	ae	af
supp	1	2	2	3	2

L_2	bc	bd	be	bf	cd
supp	0	2	2	2	0

L_2	ce	cf	de	df	ef
supp	2	2	2	1	4

L_2	ab	ac	ad	ae	af	bc	bd	be	bf	cd	ce	cf	de	df	ef
supp	1	2	2	3	2	0	2	2	2	0	2	2	2	1	4

2-itemsets fréquents

A l'itération suivante, on ne prend pas en considération les 3-itemsets contenant : ab, bc, cd, df.

P0

C_3	**supp**
Ace	2
acf	2
ade	1

P1

C_3	**supp**
aef	2
bde	1
bef	2

P2

C_3	**Supp**
cef	2

L_3	ace	acf	ade
supp	2	2	1

L_3	aef	bde	bef
supp	2	1	2

L_3	cef
supp	2

L_3	ace	acf	ade	aef	bde	bef	cef
supp	2	2	1	2	1	2	2

3-itemsets fréquents

A l'itération suivante, on ne prend pas en considération les 4-itemsets contenant : ade, bde.

P0

C_4	supp
acef	2

→

L_4	acef
supp	2

↓

L_4	acef
supp	2

4-itemsets fréquents

3.4.3 Discussion

L'algorithme DD, proposé par R.Agrawal et J.C.Shafer, est une exécution parallèle de l'algorithme séquentiel Apriori. Cet algorithme a traité le problème de mémoire de l'algorithme CD en partitionnant l'ensemble des candidats entre les processeurs. Ce partitionnement est aléatoire, en effet l'algorithme DD utilise le mode Round Robin. DD partitionne donc et la base de données et l'ensemble des candidats. Chaque processeur calcule alors les nombres d'apparition des candidats, qui lui sont attribués, dans la totalité de la base pour cela il a besoin d'accéder aux portions de la base qui sont stockées chez les autres processeurs aussi bien que sa portion locale.

Les processeurs sont indépendant les uns des autres et il y a communication entre eux après chaque itération, la communication de type all-to-all broadcast pose le problème de temps de communication très élevé. Cela diminue des performances de cet algorithme.

Cet algorithme divise l'ensemble des candidats ce qui évite la redondance de calcul.

3.5 Intelligent data distribution (IDD)

L'algorithme IDD a était développé pour remédier aux problèmes de temps de communication élevé de l'algorithme DD provoqué par des opérations d'émission et de réception collectives de buffers assez volumineux.

Dans IDD, on ne fait plus de all-to-all broadcast. En effet les processeurs sont vues comme étant un anneau logique, où chacun connaît son prédécesseur et son successeur et possède un buffer d'émission (SBuf) et un buffer de réception (RBuf).

Le processus d'échange de données entre les processeurs se fait comme suit :

Initialement SBuf contient un bloc de la base de données, ensuite chaque processeur entame une opération d'envoie asynchrone de SBuf à son successeur et une opération de réception asynchrone dans RBuf en provenance de son prédécesseur.

Pendant que ces opérations d'émissions et de réceptions se déroulent, chaque processeur parcoure les transactions de SBuf et met à jour le support des items candidats locaux. Après l'opération de mise à jour, le processeur attend la fin des communications asynchrones, celle-ci étant terminé, les contenus de SBuf et RBuf sont permutés. Ainsi l'opération précédente se poursuit (P-1) fois ; avec P : le nombre de processeurs.

3.5.1 Les étapes de l'algorithme IDD

La seule différence entre les étapes de cet algorithme et l'algorithme DD, c'est la deuxième étape :

Notation : BSend : Buffer d'émission, BRecv : Buffer de réception.

Détail de la deuxième étape

Le processeur Pi :

1. alloue deux buffers BSend et BRecv.
2. charge une page de Di dans le buffer BSend.
3. Répéte le traitement suivant N-1 fois :

    Déclencher une opération d'émission asynchrone de BSend.

    Déclencher une opération de réception asynchrone dans BRecv.

    Traiter BSend.

    Attendre la terminaison des deux opérations de transmissions asynchrones.

    Copier le contenu de BRecv dans BSend.

4. si la base Di n'est pas traitée en totalité alors reprendre en 2.

Fin.

Traiter le buffer BSend revient à :

Calculer le support des candidats dans ce buffer

Faire une mise à jour du compte ultérieur

3.6 Eclat

Introduit par M. J. Zaki en 1997, cet algorithme repose sur le découpage de la base en classes d'équivalences et distribution de la charge de travail sur tous les processeurs. On considère que deux itemsets (ensemble d'items) sont dans la même classe d'équivalence s'ils, désignés par les items qu'ils contiennent dans l'ordre lexicographique, possèdent un préfixe commun. Par exemple, les itemsets ABC et ABD sont dans la classe d'équivalence AB. Au lieu de transmettre des supports locaux ou des portions de base de données comme dans les principaux algorithmes dérivés d'Apriori, cet algorithme fonctionne en transmettant les listes de transactions correspondant à chaque classe d'équivalence au processeur qui s'occupe de celle-ci.

3.6.1 Partitionnement en classe d'équivalence

En définissant une application bijective f de l'ensemble des items vers l'ensemble des entiers, on peut définir un ordre total sur l'ensemble des items. En effet si f est bijective, alors chaque item dispose d'un identifiant unique et chaque identifiant est propre à un item. Dans la suite pour clarifier la notation, on désignera un item par une lettre majuscule et l'ordre utilisé sera l'ordre lexicographique.

Un ensemble d'items sera représenté par une suite de lettres majuscules qu'on ordonnera donc suivant l'ordre lexicographique. Disposant d'un ordre sur les items, on peut définir des classes d'équivalences sur les itemsets basées sur les préfixes communs de l'écriture de deux itemsets.

Les items A, B, C et D sont tels que f (A) < f (B) < f (C) < f (D) alors, les itemsets s'écrivent dans l'ordre croissant de la valeur par f de leurs éléments. Ainsi, l'itemset contenant les items B, A et C s'écrit ABC, celui contenant les items D, B et A s'écrit ABD et ces deux itemsets appartiennent à la classe d'équivalence E_{AB}définie par leur préfixe commun de taille | itemset | - 1, AB.

3.6.2 Les étapes de l'algorithme Eclat

- **Phase d'initialisation**

L'algorithme commence par scanner la base afin de construire les itemsets fréquents de taille 2. En effet, il est possible de générer ces ensembles avec peu de coût supplémentaire par rapport à la génération des itemsets fréquents de taille 1, profitant ainsi du gain obtenu en évitant de scanner la base 2 fois. Cependant, si la base de données contient un grand nombre d'items, il est peut-être préférable de scanner la base deux fois afin d'éliminer les items infréquents avant de générer les itemsets de taille 2. A ce stade, on dispose de l'ensemble des itemsets fréquents L_2.

– On scanne la base de données.

– On construit un tableau de deux dimensions indexé par les items sur la hauteur et la largeur.

– Chaque processeur calcule le support local des itemsets puis effectue une réduction de somme des résultats des autres processeurs afin de construire les supports globaux de chaque itemset de L_2.

- **Phase de transformation**

L'algorithme commence par partitionner L_2 en classes d'équivalences qui seront redistribuées sur les processeurs avec une politique d'équilibrage de charge basée sur une heuristique. On effectue alors une transformation de la base afin d'obtenir, non plus une liste d'items par transaction mais une liste de transactions par item (transformation verticale de la base.)

– Partitionnement de L_2 en classes d'équivalences

– Calcul de la charge de travail pour chaque classe d'équivalence

La mesure est effectuée en fonction du nombre d'éléments s de la classe d'équivalence. On considère toutes les paires à traiter par la suite. Ainsi, la charge est calculée par la valeur C_s^2 .

Par exemple, si dans la classe d'équivalence [A] on trouve les itemsets AB, AC et AD, la charge calculée sera $C_3^2 = 3$. Il s'agit alors de repartir les tâches à effectuer sur les processeurs en fonction d'une heuristique sur les charges calculées : On assigne toutes les classes d'équivalences par charge décroissantes sur le processeur qui a la charge la plus petite au moment de l'assignation.

Chaque processeur peut effectuer cette tâche indépendamment des autres puisqu'ils disposent alors tous des supports globaux de L_2 .

– Phase de transformation verticale de la base

Chaque processeur scanne sa portion locale de la base afin de construire les listes de transactions correspondant aux classes d'équivalence de L_2. Il faut alors transmettre respectivement les listes aux processeurs chargés de la classe d'équivalence correspondante.

- **Phase asynchrone**

Algorithme *Construction* (E_{k-1})

pour tous les itemsets I_1 et $I_2 \in E_{k-1}$

si ((I_1.transactions $\cap$ I_2.transactions) $\geq$ support minimum)

Ajouter (I1 U I2) à L_k

Partitionner L_k en classes d'équivalences.

pour chaque classe d'équivalence $E_k \in L_k$

Construction (E_k) [11]

Les processeurs effectuent concurremment la construction des itemsets de tailles croissantes par intersection des listes de transactions des éléments de chaque classe d'équivalence entre eux. Eliminant les itemsets de support insuffisant, on réduit rapidement le travail à effectuer en même temps qu'on augmente la taille des itemsets construits. C'est du moins ce qu'il se passe sur des données réelles.

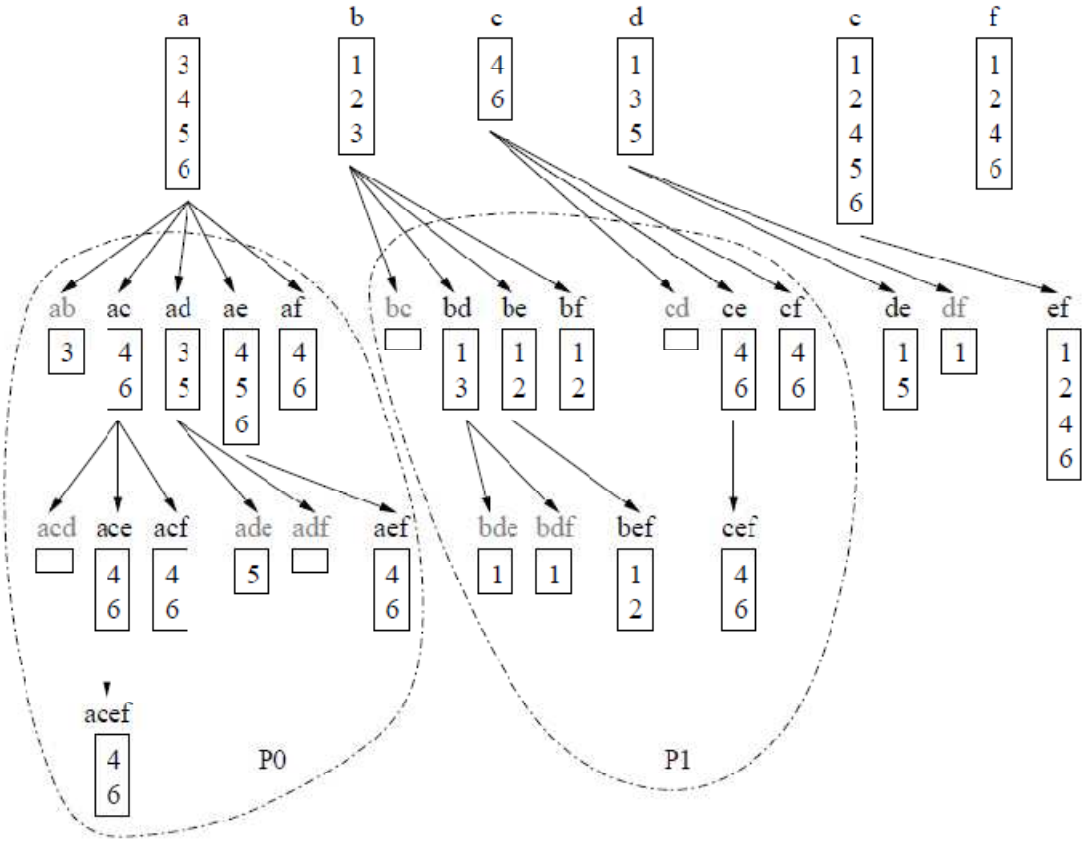
- **Phase de réduction finale**

La dernière tâche de l'algorithme consiste en l'accumulation et la réunion des résultats de chaque processeur.

3.6.3 Exemple de déroulement

Pour un support minimum = 2.

Tid	Items
1	fdbe
2	feb
3	adb
4	aefc
5	ade
6	acfe



3.6.4 Discussion

L'algorithme *Eclat* est le premier algorithme proposant une approche en profondeur. Cette méthode consiste à énumérer les motifs fréquents dans un ordre prédéfini. Par exemple, les motifs ADEB et ABDE sont les mêmes motifs et il existe plusieurs façons de générer ce motif.

En choisissant d'ordonner les motifs par ordre lexicographique, le motif ABDE ne sera généré qu'une seule fois, par son parent ABD, et il sera généré seulement si ABD est fréquent. Ce principe simple améliore grandement les performances de l'algorithme.

Cet algorithme a plusieurs avantages, il utilise des classes d'équivalences qui sont indépendantes, il peut donc être parallélisé et après le partitionnement aucune communication n'est nécessaire entre les processus. Les classes d'équivalences sont basées sur la notion de préfixe commun, par exemple, les motifs ABC, ABD et ABE font partie de la même classe d'équivalence, car ils ont le motif AB pour préfixe commun. Les 4-motifs qui peuvent être générés par cette classe d'équivalence sont ABCD, ABCE et ABDE. Notons que ces motifs ne peuvent pas être générés par une autre classe d'équivalence.

3.7 Classification des algorithmes

On présente dans ce qui suit une liste des caractéristiques permettant de montrer les différences majeures qui pourraient exister entre les algorithmes parallèles et distribués passés en revue.

Architecture : il existe deux types d'architecture, à savoir séquentiel et distribué, dans notre tableau il n y a que les algorithmes distribué.

Stratégie de partitionnement : Deux stratégies de partitionnements ont été avancées : un partitionnement des données et un partitionnement de l'espace de recherche.

Algorithme de base : Se sont des algorithmes séquentiels de base utilisés pour extraire les itemsets fréquents.

Technique de parcours : Il existe deux techniques d'exploration de l'espace de recherche, à savoir « tester et générer » et « diviser pour régner ».

Type de représentation : c'est la forme de représentation des données sur la mémoire. Il existe plusieurs types de représentation, parmi elles : les tableaux, les arbres ou les pointeurs.

Communication : Quelque soit le principe de l'algorithme distribué, il nécessite de nombreuse communications entre les divers sites

	CD	**DD**	**IDD**	**Eclat**
Architecture	distribué	distribué	distribué	distribué
Stratégie de partitionnement	Horizontale	Horizontale	Horizontale	Verticale
Algorithme de base	Apriori	Apriori	Apriori	Apriori
Technique de parcours	Tester et générer	Tester et générer	Tester et générer	Tester et générer
Type de Représentation	tableau	tableau	tableau	Arbre
Communication	Candidats + supports	itemsets fréquents locaux + supports	itemsets fréquents locaux + supports	listes de transactions

Le Tableau résume la catégorisation des algorithmes par rapports aux dimensions données précédemment. On remarque que la majorité des algorithmes basés sur le partitionnement des données se focalisent sur le partitionnement horizontal.

Ainsi, dans la plupart des algorithmes cités précédemment, on peut aisément remarquer que l'algorithme de base est l'algorithme Apriori qui utilise la technique tester et générer.

La phase de communication majeure vient de l'étape de transfert des candidats ou les itemsets fréquents locaux.

Enfin, il est important de noter qu'il reste beaucoup à faire pour améliorer certaines performances, surtout en matière de communications.

3.8 Conclusion

Dans ce dernier chapitre, nous avons étudié en détail les algorithmes distribués d’extraction des connaissances avec une méthode de décomposition en plusieurs étapes, avec une classification qui suit des critères que nous avons définis.

Conclusion générale et perspectives

Dans ce mémoire, nous nous somme intéressés au datamining parallèle et distribué pour résoudre le problème des meilleurs algorithmes séquentiels d'extraction des itemsets fréquents, qui malgré leurs performances relatives, ils risquent une saturation de l'espace mémoire et un coût de calcul élevé surtout pour les grands volumes de données.

Pour résoudre ces problèmes, plusieurs algorithmes parallèles et distribués ont été développés, comme CD, DD, IDD, ECLAT présentés sur ce mémoire.

Dans ce travail, nous avons présenté une synthèse des algorithmes distribués trouvés dans la littérature, basés sur la technique de découverte des itemsets fréquents, ensuite nous avons proposé une classification et une comparaison entre les algorithmes distribués en se basant sur des critères que nous avons définis.

Ces algorithmes s'intéressent au partitionnement des données pour un travail parallèle sur plusieurs processeurs. En effet, La majorité de ces algorithmes sont des extensions de l'algorithme APRIORI sur l'environnement distribué.

Dans nos futurs travaux, nous envisageons d'expérimenter l'un des meilleurs algorithmes distribués ECLAT sur d'autres types de plateformes, notamment des multiprocesseurs et des machines massivement parallèles comme les grilles.

Références

[1] : Daniel T.Larose, Des données à la connaissance, une introduction au datamining, Editions Vuibert Informatique, Paris, 2005.

[2] : Professeur Dr. Andreas Meier et Assistant Darius Zumstein. Le CRM analytique : Les outils d'analyse OLAP et le Data Mining, Dans le cadre du séminaire « Customer Relationship Management », Fribourg, le 26 avril 2008.

[3] : Han, Jiawei. Data mining : concepts and techniques / Jiawei Han, Micheline Kamber, Jian Pei. – 3rd ed. ISBN 978-0-12-381479-1, 2009.

[4] : Bertrand Jouve, « fouille de données » éléments de cours master 2 université de Lyon, France. 2011 – 2012.

[5] : Ed Peelen, Frédéric Jallat, Éric Stevens et Pierre Volle : Gestion de la relation client, 2ème édition, Pearson Education Benelux, Gap, 2006.

[6] : V. Fiolet, algorithmes distribués d'extraction des connaissances, Université des sciences et technologie de Lille, France, 2006.

[7] : A. Lazarevic and Z. Obradovic: Boosting Algorithms for Parallel and Distributed Learning. Distributed and Parallel Databases: An International Journal, Special Issue on Parallel and Distributed Data Mining, 2002.

[8] : B. Park and H. Kargupta: Distributed data mining: algorithms, systems, and applications. In: Nong Ye editor Data Mining Handbook, 2002.

[9] : N. Pasquier, datamining : algorithmes d'extraction et de réduction des règles d'associations dans les bases de données, Université clermant ferrand 2 ecole doctorale, France, 2000.

[10] : Rakesh Agrawal and Jhon Shafer. Parallel mining of association rules: Design, implementation and experience. Research Report RJ 10004, IBM Almaden Research Center, San Jose, California, February 1996.

[11] : Mohammed Javeed Zaki, Srinivasan Parthasarathy, and Wei Li. A localized algorithm for parallel association mining. In ACM Symposium on Parallel Algorithms and Architectures, 1997.