N° d'ordre :
N° de série :

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ CHAHID HAMMA LAKHDAR D'EL-OUED
Faculté des Sciences Exactes
Département d'Informatique

THÈSE

Pour l'obtention du grade de

DOCTEUR 3ème CYCLE LMD EN INFORMATIQUE

Option : système d'informations interopérables

Présenté par : Mme MAAMRA Oum Elhana

Titre

Structures de données complexes et distribuées dédiées à la fouille de données

Soutenu le : 02/02/2023

Devant le jury composé de:

Président :	Pr. Lejdel Brahim	Professeur	Université d'El Oued
Rapporteur :	Pr. Kholladi Mohamed-Khireddine	Professeur	Université d'El Oued
Examinateur :	Pr. Laouar Mohamed Rédha	Professeur	Université de Tébessa
Examinateur :	Dr. Bouzenada Mourad	MCA	Université de Constantine 2
Examinateur :	Pr. Amroune Mohamed	Professeur	Université de Tébessa
Examinateur :	Dr. Benali Abdelkamel	MCA	Université d'El Oued

Année universitaire 2022 – 2023

Dédicaces

Je dédie ce modeste travail

A celui qui a fait des grands efforts pour mon bonheur

A celui qui a rêvé de voir cette journée

*A celui qui m'a orienté et m'a donné les secrets de la vie : «**ma chère Mère**»*

A celle qui m'a ouvert les portails et m'a donné la tendresse et le courage

*me rendre heureuse «**mon Père**»*

*A celle qui attend chaleureusement ce jour «**mon cher mari**» pour son aide*

inconditionnel, son encouragement et

sa présence à mon côté dans les moments difficiles.

*A mes filles «**Merieme, Asma, Zaineb, Romaissa**» qui ont été patientes avec moi*

jusqu'à ce que nous voyions ce jour ensemble

A mes chères sœurs et mes chers frères pour leurs conseils et leur

encouragement.

A tous ceux que j'aime,

A tous mes amis.

Remerciements

Avant tous, je remercie dieu le tout puissant de m'avoir donné le courage et la patience pour réaliser ce travail malgré toutes les difficultés rencontrées.

Je remercie infiniment tous qui nous a aidé de prés et de loin d'avoir compléter ce travail et dépasser tous les obstacles surtout notre enseignant " ***Pr. Mohamed-Khireddine kholladi*** *" qui n'a pas cessé de nous donner les conseils et les bonnes orientations et nous prive pas de son temps et aussi on remercie* ***Pr. Saad Harous***

qui nous écoute avec grande patience pendant toute la période de préparation malgré ses contraintes de travail,

Aussi, on remercie ***Dr. Ismail Kertiou*** *qui nous écoute avec grande patience pendant toute la période de révision de mémoire malgré ses contraintes de travail,*

Mes remerciements vont également aux membres de jury qui m'ont fait l'honneur d'accepter de lire et d'évaluer ma thèse et de fournir des critiques pertinentes.

Je suis aussi reconnaissant à tous ceux qui ont collaboré à notre formation en particulier les enseignants du département d'Informatique de la faculté des Sciences exactes de l'université Chahid Hamma Lakhdar d'El-Oued.

Je remercie également tous ceux qui m'ont soutenu, encouragé et aidé de près et de loin pour la réalisation de notre étude.

RESUME

Le traitement de grands jeux de données complexe et distribué à besoin de minimiser le gaspillage de ressources des réseaux et en maximisant l'interaction avec un utilisateur. Où trouver l'utilisation de l'application visée dans le domaine de l'aide à la décision ou d'analyse de données. D'autre part, Le requête plus complexes multicritère Skyline utilisé dans de nombreuses applications nécessitant des décisions multicritères pour identifier les meilleurs résultats en fonction des préférences ou des conditions des utilisateurs. Dans ce contexte, cette thèse présente la proposition de deux approches distinctes se situent dans le cadre d'optimisation le temps d'extraire une information et augmenter la précision de sélection cette information dans une base de données volumineux, hétérogènes et distribués.

Dans la première approche, on a proposé un système multi agent qui compose cinq agents, tel que chaque agent gère un composant du système ce qui assure la division des tâches du système entre les agents. Dans le cadre d'analyse des données nous avons proposé une base de résultats permettons de minimiser la quantité des données dans le moteur de recherche des règles d'associations à la méthode de Pasquier qui divise les générateurs des règles aux deux bases : base générique pour les règles d'association exactes et base informative pour les règles d'association approximatives. La deuxième proposition présente une nouvelle approche de sélection des meilleurs éléments appliquée selon les préférences d'utilisateurs dans le cadre du traitement du problème multi critère. Cette approche s'appuie sur la proposition d'un algorithme de requête Skyline (Dynamics join Skyline) dans un environnement complexe par l'utilisation des options de l'IOT et été appliqué dans le cadre de réservation des services des voyages. Une série d'expérimentations ont été effectuée et elle donne des résultats bien meilleurs.

Mots-clés: Fouille de données, systèmes distribués, IOT, Algorithmique de Skyline, Système complexe, itemset fermé fréquent, Base de données hétérogène, Décision multicritère.

ABSTRACT

The processing of large, complex and distributed datasets needs to minimize the waste of network resources and maximize interaction with a user. Where to find the use of the targeted application in the field of decision support or data analysis. On the other hand, the more complex multi-criteria query Skyline used in many applications requiring multi-criteria decisions to identify the best results based on user preferences or conditions. In this context, this thesis presents the proposal of two distinct approaches located within the framework of optimizing the time to extract information and increased the accuracy of selecting this information in a large, heterogeneous and distributed database.

In the first approach, a multi-agent system has been proposed which consists of five agents, such that each agent manages a system component which ensures the division of system tasks between the agents. As part of data analysis, we have proposed a database of results that makes it possible to minimize the amount of data in the search engine for association rules using the Pasquier method, which divides the rule generators into two bases: generic base for exact association rules and informative base for approximate association rules. The second proposal presents a new approach for selecting the best elements applied according to user preferences as part of the treatment of the multi criterion problem. This approach is based on the proposal of a Skyline query algorithm (Dynamic join Skyline) in a complex environment through the use of IOT options and has been applied in the context of booking travel services. A series of experiments have been carried out and it gives much better results.

Keywords: Data mining, distributed systems, IOT, Skyline algorithms, Complex system, Frequent closed itemset, Data base heterogenic, Multicriteria decision.

ملـخـــص

تحتاج معالجة مجموعات البيانات الكبيرة والمعقدة والموزعة إلى تقليل هدر موارد الشبكة وتعظيم التفاعل مع المستخدم. أين يمكن العثور على استخدام التطبيق المستهدف في مجال دعم القرار أو تحليل البيانات. من ناحية أخرى ، فإن أفق الاستعلام متعدد المعايير الأكثر تعقيدا المستخدم في العديد من التطبيقات التي تتطلب قرارات متعددة المعايير لتحديد أفضل النتائج بناء على تفضيلات المستخدم أو شروطه. في هذا السياق ، تقدم هذه الأطروحة اقتراح نهجين متميزين يقعان في إطار الاستفادة المثلى من الوقت لاستخراج المعلومات وزيادة دقة اختيار هذه المعلومات في قاعدة بيانات كبيرة وغير متجانسة وموزعة.

في النهج الأول ، تم اقتراح نظام متعدد الوكلاء يتكون من خمسة وكلاء ، بحيث يدير كل وكيل مكون نظام يضمن تقسيم مهام النظام بين الوكلاء. كجزء من تحليل البيانات, لقد اقترحنا قاعدة بيانات للنتائج تجعل من الممكن تقليل كمية البيانات في محرك البحث لقواعد الارتباط باستخدام طريقة باسكوير, الذي يقسم مولدات القواعد إلى قاعدتين: قاعدة عامة لقواعد الارتباط الدقيقة وقاعدة إعلامية لقواعد الارتباط التقريبية. ويقدم الاقتراح الثاني نهجا جديدا لاختيار أفضل العناصر المطبقة وفقا لتفضيلات المستخدم كجزء من معالجة مشكلة المعايير المتعددة. ويستند هذا النهج على اقتراح خوارزمية الاستعلام أفق (ديناميكية الانضمام أفق) في بيئة معقدة من خلال استخدام خيارات تقنيات عمليات وتم تطبيقها في سياق حجز خدمات السفر. تم إجراء سلسلة من التجارب وتعطي نتائج أفضل بكثير.

الكلمـــــــات المفتاحـــــية: استخراج البيانات ، الأنظمة الموزعة ، أنترنت الاشياء، خوارزمية الأفق ، النظام المعقد ، مجموعة العناصر المغلقة المتكررة، قواعد البيانات المختلفة، القرار متعدد المعايير.

Table des matières

Chapitre II : Extraction des Motifs Fréquents

Liste des figures

Liste de table

INTRODUCTION GENERALE

Dans un environnement distribué, les grands jeux de données ont besoin d'être manipulé et analysé rapidement pour minimiser la perte des ressources réseaux et maximiser l'interaction avec les utilisateurs. Le contexte d'utilisation des applications de l'aide à la décision ou d'analyse de données dans des domaines très différents : analyse de facteurs de risque pour la santé, découverte de tendances pour le marketing, la recherche d'informations pour le grand public (des compromis multi critères tels que « quel est le meilleur restaurant, proche du centre-ville, de prix abordable, ... ? »)

D'autre part, le traitement d'une requête Skyline est devenu un sujet crucial pour la recherche dans les bases de données, car il permet d'extraire des objets intéressants à partir d'un ensemble de données multidimensionnelles. Les requêtes Skyline peuvent être utilisées dans de nombreuses applications nécessitant des décisions multicritères. Sans utiliser de fonctionnalité cumulative, Skyline permet d'identifier les meilleurs résultats en fonction des préférences des utilisateurs. La principale caractéristique qui distingue une requête Skyline d'autres requêtes de préférence, telle que les requêtes Top K, est qu'elle est indépendante de la définition d'une fonction de score lors de la classification des tuples. Cependant, le seul inconvénient est la relation partielle entre les valeurs de chaque dimension.

1. Contexte du travail

Un système informatique distribué est défini comme étant un système possédant plusieurs éléments de traitement autonomes coopérants pour atteindre un but commun. Aujourd'hui, presque tous les systèmes informatiques sont distribués, pour les raisons suivantes [Site_1, 20] :

- Géographie : les grandes organisations et entreprises sont réparties géographiquement. Les systèmes informatiques doivent manipuler ce problème.
- Parallélisme : pour accélérer le calcul, on utilise des processeurs multicœurs ou des grappes de calcul.
- Fiabilité : les données sont répliquées sur différentes machines pour éviter toute perte de données.
- Disponibilité : les données sont répliquées sur différentes machines pour permettre un accès à tout moment, sans goulots d'étranglement, en minimisant la latence.

Les systèmes distribués à grande échelle (Datacenters, Grille, Clouds, Réseaux) sont des acteurs incontournables dans notre communauté de communication et des échanges électroniques. Ces infrastructures doivent faire face à de nombreux challenges qui limitent leur déploiement : sécurité, qualité de service, extensibilité, consommation électrique, etc.

Cette thèse retrace les travaux menés sur les domaines de la flexibilité des grands systèmes en se focalisant sur la mise en œuvre de solutions dynamiques de déploiement de services. Le but est d'augmenter la valeur ajoutée des infrastructures existantes et proposer de nouveaux services. La flexibilité des systèmes distribués est due à plusieurs facteurs (les Datacenters, les grilles, le Cloud et les réseaux). Malgré cette flexibilité, on a plusieurs défis à résoudre tel que la sécurité, la qualité de service et la consommation d'énergie [Lefevre, 13]. Ce type de problème nécessite de fournir un cadre théorique pour modéliser le processus de sélection d'information (SI), et discuter la complication introduite par la distribution.

La prise de décision multicritères est l'un des facteurs les plus critiques de la veille économique d'entreprise moderne. Il est bien connu que les décisions commerciales complexes sont souvent prises sur la base de critères multidimensionnels. La compétitivité d'une prise de décision commerciale optimale repose généralement sur la recherche d'un bon compromis entre

de nombreux critères différents et éventuellement contradictoires, par exemple, le profit maximum, le prix minimum et la consommation minimale de ressources.

Actuellement, le principal challenge scientifique est d'offrir un système capable d'explorer les sources de données pour sélectionner les meilleurs résultats. Cependant, il faut résoudre les problèmes d'hétérogénéité qui induit à une dégradation des performances du système et une désorientation d'utilisateurs. Les travaux existant sur la sélection de données visent à appliquer des requêtes plus complexes comme le Skyline dans des environnements souvent homogènes.

2. Problématiques

La prise de décision est un champ de recherche majeur en Sciences de gestion, plus particulièrement dans le domaine de la gestion des systèmes d'Information décisionnels. Elle s'intéresse à l'entreposage de données et l'analyse en ligne. L'informatique décisionnelle est aujourd'hui confrontée à de nouveaux défis scientifiques. De nouvelles sources d'information hétérogènes, fortement évolutives, changeantes, dépendantes ou autonomes et distribuées apparaissent. L'extraction de la connaissance depuis les entrepôts de données volumineuses (ECD) nécessite des solutions robustes et scalables pour surpasser les contraintes temporelles et spatiales des algorithmes classiques. L'opérationnalisation du processus ECD est réalisée par la fouille de données qui fait appel à des multiples méthodes de la statistique, de l'apprentissage automatique, de la reconnaissance de formes ou de la visualisation. Pour cela, il est nécessaire d'optimiser toutes les étapes de la chaîne conduisant à l'application finale, grâce à des modèles formels. Dans ce contexte, nous travaillerons pour optimiser le temps d'extraction et améliorer la précision de la sélection des informations dans les bases de données volumineuses, hétérogène et distribuée.

La matérialisation (ou résumés/pré calcul de résultats) est une des techniques permettant d'optimiser les requêtes qui peut trouver « rapidement » les résumés qui sont aidés l'utilisateur à préciser sa requête.

Le problème qui se pose est décrit comme suit : étant donné un ensemble de requêtes cibles (workload), quelles sont les meilleures parties de la base de données qu'on doit matérialiser afin d'optimiser le workload ?

Le travail demandé pour cette thèse consiste dans un premier temps à chercher les parties de la base à précalculer. Par la suite, les solutions sont étendues pour répondre aux requêtes multicritères plus complexes. Autour de ces principales problématiques, plusieurs questions qui se posent quand cherchent les parties de la base à précalculer:

- par qui les calculer dans le cas d'une machine multi-cœur ou un réseau pair à pair,
- où les stocker,
- et comment les exploiter lors des requêtes à l'aide d'interaction visuelles efficaces ?

À travers ce travail, nous développons une nouvelle approche de prise de décision multicritère dédiée aux données distribuées et hétérogènes. Dans le but est de faciliter l'exploration et l'utilisation de ce type de données. Dans la section suivante, on va présenter les différentes contributions proposées dans le cadre de cette thèse.

3. Contributions

Le travail proposé dans cette thèse vise à optimiser les performances de calcul et de stockage distribués de structures de données complexes et dynamiques. Ces données sont souvent stockées dans des nombreuses sources hétérogènes, conçues indépendamment les unes des autres. Dans le cadre de ce travail, nous avons proposé deux approches :

- La première approche permet d'analyser le système de réservation [Maamra, 18]. Cette approche étend l'extraction des connaissances à partir des données distribuées. Ainsi, nous avons basé sur l'extraction des items sets fermés fréquents par une base des résultats proposée qui ont été identifiés à travers notre architecture du système proposé (système de réservation) [Maamra, 18] et [Maamra, 20]. Dans cette proposition, nous avons développé une architecture du système distribuée et hétérogène basée sur le système multi agents.
- La deuxième est une approche de décision multicritère basée sur la requête Skyline. Cette approche permet de traiter le problème de sélection des meilleurs éléments à partir des sources de données complexe selon les préférences des utilisateurs [Maamra, 22]. Nous avons proposé un algorithme pour la requête Skyline (Dynamics join Skyline) dans un environnement complexe par l'utilisation des options de l'IOT et été appliqué dans le cadre de réservation des services des voyages. Le processus de recherche et de sélection proposé est divisé en deux étapes : la première

permet de sélectionner les meilleurs éléments au niveau des serveurs locaux et la deuxième permet de sélectionner le meilleur élément au niveau du serveur du système.

Enfin, nous avons terminé notre travail par une expérimentation qui permet d'évaluer les approches proposées. Les résultats de cette expérimentation montrent l'efficacité et l'adaptabilité de nos contributions pour le traitement du problème de calcul et de stockage distribué de structures de données complexes et dynamiques. Ces données sont souvent stockées dans des nombreuses sources hétérogènes, conçut indépendamment les unes des autres. De plus, la comparaison avec les travaux similaires montre l'originalité de nos contributions.

4. Organisation de la thèse

Notre thèse se compose de cinq chapitres répartis en deux grandes parties. La première partie est divisée en trois chapitres représentant l'état de l'art des thèmes abordés par notre travail. La seconde partie est composée de deux chapitres décrivant nos contributions.

Partie 01. Etat de l'art

Cette partie comporte les chapitres suivants :

Chapitre 01. Bases de données distribuées

Dans ce chapitre, nous présentons une vue globale sur les bases de données distribuées (BDD). Au début, nous définissons les concepts de base de BDD et l'architecture d'un SGBD distribué. Par la suite, nous présentons la structure de base de données distribuée et parallèle. Après, nous expliquons la méthodologie d'intégration de base de données. Enfin, nous discutons la manière de traitement des requêtes distribuées.

Chapitre 02. Extraction des Motifs Fréquents

Dans ce chapitre, nous présentons le processus d'extraction des connaissances. Au début, nous définissons la notion de data mining. Par la suite, nous citons des généralités sur le datamining distribué. Enfin, nous expliquons et discutons les algorithmes d'extraction des itemsets fréquents distribués.

Chapitre 03. Les requêtes Skyline : outils d'optimisation des problèmes de décisions multicritères

Dans ce chapitre, nous présentons les requêtes Skyline. Au début, nous définissons les bases des requêtes Skyline. Puis, nous citons les différentes familles des algorithmes des requêtes Skyline. Enfin, nous expliquons les notions de Skyline subspatiaux et cubes de Skyline.

Partie 02. Contributions

Cette partie présente notre contribution. Elle est composée de deux chapitres :

Chapitre 04. Traitement d'hétérogénéité de données dans un environnement distribué

Dans ce chapitre, nous discutons la première contribution. Nous présentons notre approche d'analyse de très grands jeux de données, dans un environnement distribué. De plus, nous exposons l'architecture du système proposé, les sources utilisées et les algorithmes appliqués.

Chapitre 05. Approche de décision multicritère basée sur la requête Skyline

Dans ce chapitre, nous discutons la deuxième contribution. Au début, nous présentons notre approche de sélection des meilleurs éléments par des requêtes plus complexes multicritères pour satisfaire les désirs d'utilisateurs. Par la suite, nous détaillons les différents modèles utilisés pour la modélisation. Puis, nous présentons le processus de recherche et de sélection proposé. Enfin, nous développons une série d'expérimentations avec l'analyse des résultats obtenus.

Nous terminons ce travail de thèse par une conclusion générale mettant en évidence une synthèse globale des résultats acquis et les perspectives de recherches.

Partie 01

Etat de l’art

Chapitre I

Bases de données distribuées

1. Introduction
2. Bases de données distribuées
    - 1.1. Définitions
    - 1.2. Objectifs
    - 1.3. Complications introduites par la distribution
    - 1.4. Architecture des SGBD distribués
3. Structure d'une base de données distribuée
    - 3.1. Fragmentation des données
    - 3.2. Allocation
    - 3.3. Répertoire de données
4. Intégration des bases de données
    - 4.1. Méthodologie de la conception Bottom-Up
    - 4.2. Correspondance de schémas
    - 4.3. Intégration de schémas
    - 4.4. Planification de schémas
5. Traitement de requête distribuée
    - 5.1. Décomposition de la requête
    - 5.2. Localisation de données
    - 5.3. Optimisation globale
    - 5.4. Optimisation locale
6. Conclusion

1. Introduction

Aujourd'hui, les systèmes distribués (SD) sont devenus le cœur de tous les domaines d'utilisation du Web, ce qui rend les informations et les services accessibles au plus grand nombre possible d'utilisateurs. La flexibilité des systèmes distribués est due à plusieurs facteurs (les datacenters, les grilles, le cloud et les réseaux). Malgré cette flexibilité, les défis de ces systèmes sont la sécurité, la qualité de service et la consommation d'énergie.

Ce chapitre est organisé en trois grandes parties : la première présente les concepts de base de BDD et l'architecture d'un SGBD distribué qui ont été proposés pour fournir un cadre théorique pour la modélisation du processus de sélection d'informations (SI), et discuter la complication introduite par la distribution. La deuxième partie décrit la structure de base de données distribuées et l'intégration des bases de données. La troisième partie sera consacrée au traitement de requêtes puis décomposition des requêtes et localisation de données. Enfin, nous terminons le chapitre par une synthèse des problèmes d'optimisation des requêtes distribuées.

2. Bases de données distribuées

2.1. Définitions

Une base de données distribuée est une collection d'éléments nommés. Chaque élément a un nom et entre 1 et n valeurs qui lui sont associées, où n'est pas le nombre de nœuds dans le système. Chaque valeur d'un élément donné est stockée dans un nœud différent du système. De plus, chaque élément i est associé à un ensemble S (i). L'ensemble S (i) est l'ensemble des nœuds qui ont une valeur pour l'élément qui y est stocké [GARCIA-MOLINA, 82].

On peut définir une base de données distribuée (BDD) comme un ensemble de bases de données multiples, logiquement liées, réparties sur un réseau informatique. Un système de gestion de base de données distribuée (SGBD distribué) est alors défini comme le système logiciel qui permet la gestion de la base de données distribuée et rend la distribution transparente pour les utilisateurs [ÖZSU, 96].

Les bases de données distribuées sont classées comme des bases de données situées sur des différentes machines où des emplacements identiques qui apparaissent comme une base de données centralisée pour les utilisateurs finals [Grech, 09]. Ainsi, toute la charge est partagée par un ensemble de machines/ ordinateurs, plutôt que de fournir une base de données centralisées. En réalité, il s'agit d'un certain nombre de machines serveurs fonctionnant de manière synchronisée pour répondre aux exigences de plusieurs utilisateurs. Ces machines sont reliées entre elles dans un système distribué via une connexion sans fil ou via divers supports de communication qui transmettent des données à haut débit [Orlunwo placida, 21].

Toutes ces définitions partagent l'idée que BDD est une base de données logiquement distribué sur plusieurs sites et visibles à l'utilisateur final comme une seule base de données.

2.2. Objectifs

Les avantages des BDD peuvent être résumés en cinq principes fondamentaux [ÖZSU, 99] :

- Une gestion transparente des données distribuées et répliquées ;
- Des accès fiables aux données via des transactions distribuées ;
- Des performances améliorées ;
- Une extension plus facile du système ;
- Et un traitement des requêtes distribuées.

2.2.1. Gestion transparente des données distribuées et répliquées

La transparence fait la séparation entre le niveau supérieur d'un système et le niveau inférieur. En d'autres termes, un système transparent ″cache″ les détails de mise en œuvre aux utilisateurs. L'avantage d'un SGBD transparent est de fournir un support de haut niveau pour le développement d'applications complexes. Pour ce but, la transparence est généralement divisée en des types, qui sont : indépendance des données, transparence du réseau et transparence de fragmentations [Kahn, 15].

2.2.1.1. Indépendance des données

L'indépendance des données est une forme fondamentale de transparence au sein d'un SGBD. C'est aussi le seul type important dans le cadre d'un SGBD centralisé. Il fait référence à l'immunité des applications des utilisateurs aux changements dans la définition et l'organisation des données, et vice versa [Bouberka, 22].

La définition des données se fait à deux niveaux (structure logique et structure physique). On peut définir deux types d'indépendance des données : l'indépendance logique des données et l'indépendance physique des données. L'indépendance logique des données signifie que les applications des utilisateurs ne sont pas autorisées à modifier la structure logique des données (c'est-à-dire le schéma) de la base de données. L'indépendance physique des données signifie que les détails de la structure de stockage des données seront cachés aux applications des utilisateurs [Gutiérrez, 21].

2.2.1.2. Transparence du réseau

Ce type de transparence fait référence à ce que l'existence du réseau est donc cachée à l'utilisateur. L'accès aux données de la base doit se faire de la même manière soit en mode centralisé ou en mode distribué [ÖZSU, 99].

2.2.1.3. Transparence de réplication

Pour des raisons de performance et de disponibilité des données, il est préférable de faire des copies de quelques données de la base sur plusieurs sites. Par exemple, les données qui se trouvent sur un autre site et qui sont réclamées souvent par un utilisateur, il serait préférable de faire une copie sur la base des données locales de cet utilisateur. En plus, si une machine s'arrête, on peut retrouver les données sur une autre machine.

La décision de réplication des données est de combien de copies peut-on faire ? En fait, cela dépend des applications des utilisateurs. L'utilisateur n'aura pas besoin de savoir si les données sont répliquées ou non. Il envoie ses requêtes comme s'il existe une seule copie des données [ÖZSU, 11].

2.2.1.4. Transparence de fragmentation

La transparence de fragmentation se fait pour des raisons de performances, de disponibilité et de fiabilité. De plus, la fragmentation peut réduire les inconvénients de la réplication. Cette opération de fragmentation doit être faite d'une manière transparente à l'utilisateur. Il utilise la table originale comme si elle n'est pas fragmentée. Il existe deux types de fragmentations [ÖZSU, 11] :

- Fragmentation horizontale : La table est partitionnée en plusieurs sous tables contenants chacune un sous-ensemble de lignes de la table originale.
- Fragmentation verticale : La table est partitionnée en plusieurs sous tables qui sont définies sur un sous-ensemble d'attributs (de colonnes) de la table originale.

2.3. Complications introduites par la distribution

Les problèmes rencontrés dans les systèmes de base de données prennent une complexité supplémentaire dans un environnement distribué. De plus, cette complexité supplémentaire donne lieu à de nouveaux problèmes influencés principalement par trois facteurs :

- Premièrement, dans un environnement distribué, une base de données où des parties de celle-ci peuvent être répliquées dans différents sites sur un réseau informatique. D'autre part, Il n'est pas essentiel que chaque site du réseau contienne la base de données. Mais essentiel qu'il y ait plus d'un site où réside la base de données. La duplication possible des éléments de données est principalement due à des considérations de fiabilité et d'efficacité. Par conséquent, le système de base de données distribuée est responsable (1) de choisir l'une des copies stockées des données demandées pour l'accès en cas de récupération, et (2) de s'assurer que l'effet d'une mise à jour se reflète sur chaque copie de celle-ci.
- Deuxièmement, si certains sites échouent (par exemple, en raison d'un dysfonctionnement matériel ou logiciel), ou si certaines liaisons de communication échouent (rendant certains sites inaccessibles) pendant qu'une mise à jour est en cours d'exécution, le système doit s'assurer que les effets seront reflétés sur les données résidant sur les sites défaillants ou inaccessibles dès que le système peut être récupéré de la panne.

- Et troisièmement, est que chaque site ne pouvant pas disposer d'informations instantanées sur les actions en cours sur les autres sites, la synchronisation des transactions sur plusieurs sites est considérablement plus difficile que pour un système centralisé.

Ces difficultés indiquent un certain nombre de problèmes potentiels avec les SGBD distribués : la complexité de création des applications distribuées, du coût accru de la réplication des ressources et, plus important encore, de la gestion de la distribution, de la dévolution du contrôle à de nombreux centres et de la difficulté à conclure des accords, et des problèmes de sécurité exacerbés (le problème du canal de communication sécurisé) [ÖZSU, 99].

2.4. Architecture des SGBD distribués

Le SGBD distribué est un système gérant une collection de BD reliée logiquement, distribué sur différents sites de façon la plus transparente possible pour l'utilisateur. On distingue trois dimensions d'architecture selon lesquelles les SGBD distribués peuvent être structurés : l'autonomie des systèmes locaux, (2) leur distribution et (3) leur hétérogénéité.

L'architecture d'un système définit sa structure. Cela signifie que les composants du système sont identifiés, la fonction de chaque composant est spécifiée et les interrelations et les interactions entre ces composants sont définies. La spécification de l'architecture d'un système nécessite l'identification des différents modules, avec leurs interfaces et leurs interrelations, en matière de flux de données et de contrôle à travers le système. Il existe quatre types d'architectures pour un SGBD : client / serveur, pair à pair, multibases de données et Cloud.

2.4.1. Autonomie

L'autonomie est due à toutes les fonctions individuelles des SGBD. Les opérations locales des différents SGBD ne sont pas affectées par leur participation au système distribué, échangent des informations et peuvent exécuter des transactions de manière indépendante. Les dimensions de l'autonomie peuvent être précisées comme suit [DU, 89] :

- Autonomie de conception : les SGBD individuels sont libres d'utiliser les modèles de données, langage d'interrogation, sémantique des données et les techniques de gestion des transactions qu'ils préfèrent.

- Autonomie de communication : chacun des SGBD individuels est libre de prendre sa propre décision quant au type d'information qu'il souhaite fournir aux autres SGBD ou au logiciel qui contrôle leur exécution globale.
- Autonomie d'exécution : chaque SGBD décide de l'ordre d'exécution des transactions locales ou des opérations externes.

2.4.2. Hétérogénéité

L'hétérogénéité peut se produire sous diverses formes dans les systèmes distribués, allant de l'hétérogénéité matérielle et des différences dans les protocoles de mise en réseau aux variations dans les gestionnaires de données. La représentation des données avec différents outils de modélisation crée une hétérogénéité en raison des pouvoirs expressifs inhérents et des limites des modèles de données individuels. L'hétérogénéité dans les langages de requête implique non seulement l'utilisation de paradigmes d'accès aux données complètement différents dans différents modèles de données, mais aussi elle couvre les différences de langues même lorsque les systèmes individuels utilisent le même modèle de données.

2.4.3. Distribution

La distribution consiste en des données, qui sont stockées sur des supports répartis géographiquement, et pour l'utilisateur, il s'agit comme une BD unique. Les SGBD ont été distribués en deux classes: client / serveur et pair à pair (ou distribution complète). Avec l'option non distribuée, donc en peut identifier trois architectures alternatives.

2.4.4. Architecture Client-Serveur

L'allocation des fonctionnalités entre les clients et les services diffère selon les types de SGBD distribués (par exemple, relationnel : SGBDR ou orienté objet : SGBDOO). Dans les systèmes relationnels, le serveur effectue la majeure partie du travail de gestion des données. Cela signifie que tout le traitement et l'optimisation des requêtes, la gestion des transactions et la gestion du stockage sont effectués au niveau du serveur. Le client, en plus de l'application et de l'interface utilisateur, dispose d'un module client SGBD chargé de gérer les données qui est mis en cache sur le client et (parfois) gérer les verrous de transaction qui peuvent également avoir été mis en cache. En d'autres termes, le client transmet les requêtes SQL au serveur sans essayer de

les comprendre ou de les optimiser. Le serveur effectue la majeure partie du travail et renvoie la relation de résultat au client [ÖZSU, 99].

Il existe deux types d'architecture client - serveur sont :

- serveur - multiple clients ;
- et multiples serveurs - multiples clients.

2.4.5. Architecture pair à pair

Dans ces systèmes, chaque pair agit à la fois comme un client et comme un serveur pour transmettre des services de base de données. Les pairs partagent leurs ressources avec d'autres pairs et coordonnent leurs activités. Cette architecture comporte généralement quatre niveaux de schémas [Orlunwo Placida, 21] :

- Schéma conceptuel global : décrit la vue logique globale des données ;
- schéma conceptuel local : décrit l'organisation logique des données sur chaque site ;
- schéma interne local : décrit l'organisation physique des données sur chaque site ;
- et schéma externe : décrit la vue utilisateur des données.

2.4.6. Architecture de système multi base de données

Un système multibases de données (MDBS) est un système distribué conçu pour intégrer des données et fournir un accès à une collection de bases de données locales préexistantes distribuées, hétérogènes et autonomes gérées par des systèmes de gestion de bases de données hétérogènes (SGBD) [Orlunwo Placida, 21].

Les différences de niveau d'autonomie entre les MDBS et les SGBD distribués se reflètent également dans leurs modèles architecturaux. La différence fondamentale porte sur la définition du schéma conceptuel global. Dans le cas des SGBD distribués logiquement intégrés, le schéma conceptuel global définit la vue conceptuelle de l'ensemble de la base de données, tandis que dans le cas des MDBS, il ne représente que la collection de certaines des bases de données locales que chaque SGBD local souhaite partager [ÖZSU, 11].

L'architecture la plus populaire de MDBS est l'approche médiateur/wrapper. Un médiateur est un module logiciel qui exploite des connaissances codées sur certains ensembles ou sous-ensembles de données pour créer des informations pour une couche supérieure d'applications

[Wiederhold 92]. Ainsi, chaque médiateur remplit une fonction particulière avec des interfaces clairement définies. Étant donné que les médiateurs peuvent être construits au-dessus d'autres médiateurs, il est possible de construire une implémentation en couches.

Les wrappers sont implémentés dont la tâche est de fournir une correspondance entre une vue de SGBD source et la vue des médiateurs. Par exemple, si le SGBD source est un SGBD relationnel, mais que les implémentations du médiateur sont orientés objets, les mappages (mapping) requis sont établis par les wrappers.

3. Structure de base de données distribuée et parallèle

La conception d'un système informatique distribué implique de prendre des décisions sur le placement des données et des programmes sur les sites d'un réseau informatique, ainsi que sur la conception éventuelle du réseau lui-même. Dans le cas des SGBD distribués, la distribution des applications implique deux choses : la distribution du logiciel SGBD distribué et la distribution des programmes d'applications qui s'exécutent dessus. Pour cela, on peut dire que l'organisation des systèmes distribués tourne autour de trois axes [Özsu, 11] (voir la figure I.1) :

- Niveau de partage :
  - pas de partage ;
  - partage de données ;
  - et partagent des données et des applications.
- Type d'accès :
  - accès statique (localisation des données connue dès le début) ;
  - et accès dynamique (localisation des données inconnues).
- Niveau de connaissances sur le comportement type d'accès :
  - pas de connaissance préalable sur le type d'accès ;
  - informations partielles sur le type d'accès ;
  - et informations complètes sur le type d'accès (le cas la plus pratique).

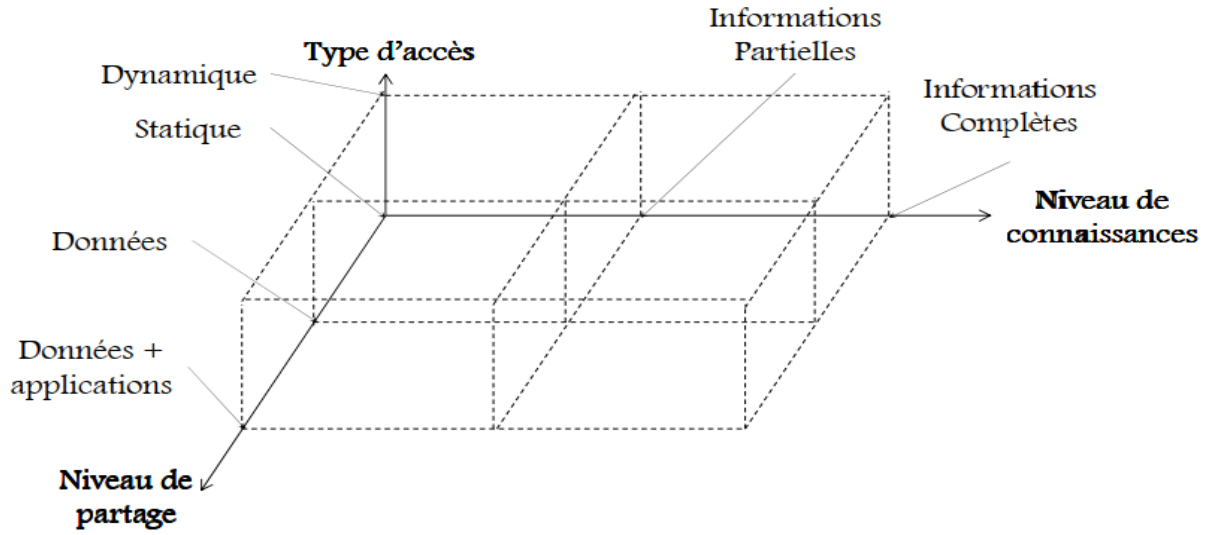


Figure I.1. Les trois axes des systèmes distribués [Özsu, 11].

3.1. Fragmentation des données

La fragmentation est une technique de conception permettant de diviser une seule relation ou classe d'une base de données en deux partitions ou plus de sorte que la combinaison des partitions fournisse la base de données d'origine sans aucune perte d'informations [ÖZSU, 99]. Cela réduit la quantité de données non pertinentes auxquelles les applications de la base de données accèdent, réduisant ainsi le nombre d'accès au disque. La fragmentation peut être horizontale, verticale ou mixte / hybride. La fragmentation verticale permet de partitionner une relation ou une classe en ensembles disjoints de colonnes ou d'attributs à l'exception de la clé primaire. Des combinaisons de fragmentations horizontales et verticales à des fragmentations mixtes ou hybrides sont également proposées.

3.1.1. Fragmentation horizontale

La fragmentation horizontale divise une relation unique R en sous-ensembles de tuples à l'aide de prédicats de requête. Ainsi, chaque fragment a un sous-ensemble des tuples de la relation mère. Il existe deux versions de fragmentation horizontale : primaire et dérivée [Kaundal, 14] (voir la figure I.1).

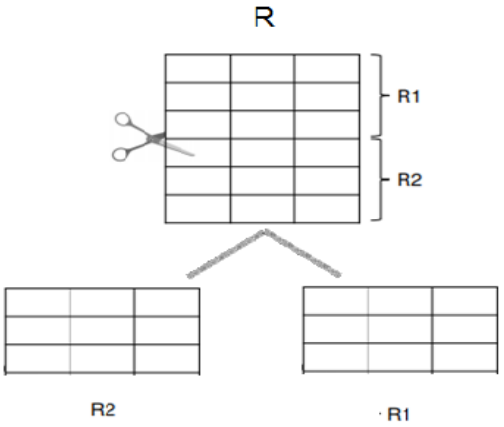


Figure I.2. Fragmentation horizontale [Kaundal, 14].

- La fragmentation horizontale primaire d'une relation est effectuée à l'aide de prédicats définis sur cette relation.
- La fragmentation horizontale dérivée est le partitionnement d'une relation qui résulte de la définition de prédicats sur une autre relation.
- La reconstitution de la relation initiale se fait grâce à l'union des sous relations.

3.1.2. Fragmentation verticale

La fragmentation verticale divise une seule relation R en sous relations qui sont des projections de la relation R, chacune contenant un sous-ensemble d'attributs R ainsi que la clé primaire, l'objectif est de partitionner une relation en un ensemble de relations plus petites pour de nombreuses applications utilisateur ne s'exécutent que sur un seul fragment. La clé primaire est incluse dans chaque fragment pour permettre la reconstruction. Ceci est également bénéfique pour l'application de l'intégrité car la clé primaire détermine fonctionnellement tous les attributs de relation, l'avoir dans chaque fragment permet d'éliminer le calcul distribué pour appliquer la contrainte de clé primaire comme sur la figure I.3.

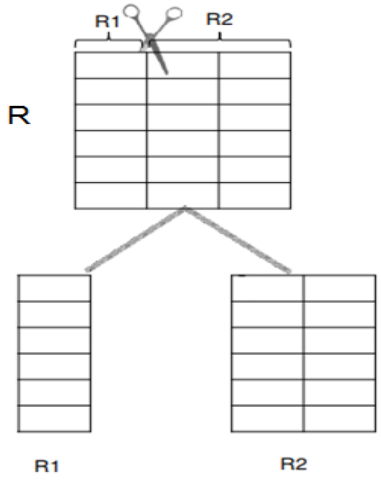


Figure I.3. Fragmentation verticale [Kaundal, 14].

Il existe deux types d'approches heuristiques pour la fragmentation verticale des relations globales [Özsu, 11] :

- Regroupement : commence par affecter chaque attribut à un fragment, et à chaque étape et joint certains des fragments jusqu'à ce que certains critères soient satisfaits.
- Fractionnement : commence par une relation et décide des partitionnements bénéfiques en fonction du comportement d'accès des applications aux attributs.

3.1.3. Fragmentation hybride

Dans certains cas, une simple fragmentation horizontale ou verticale d'un schéma de base de données peut ne pas suffire à satisfaire les exigences des applications utilisateur. Dans ce cas, une fragmentation verticale peut être suivie d'une fragmentation horizontale, ou inversement, produisant un partitionnement structuré en arborescence (voir la figure I.4). Comme les deux types de stratégies de partitionnement sont appliqués l'un après l'autre, cette alternative est appelée fragmentation hybride. Elle a également été nommée fragmentation mixte ou fragmentation imbriquée et [Gawande, 12].

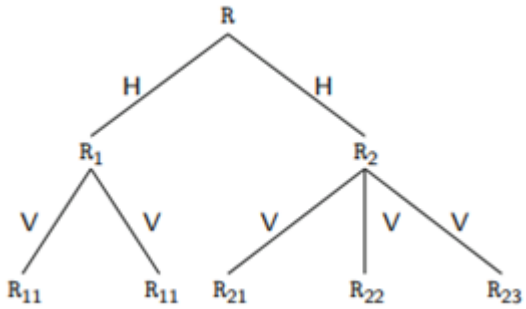


Figure I.4. Fragmentation hybride.

3.2. Allocation

Supposons qu'il existe un ensemble de fragments F= {F_1, F_2, …, F_n} distribué sur un ensemble de sites S= {S_1, S_2, …, S_m} et un ensemble de requêtes Q= {q_1, q_2, …, q_p} utilisant les fragments F situés sur les sites S. Le problème d'allocation consiste à trouver la distribution optimale de F sur les S. L'optimalité peut-être définie par rapport à deux mesures [Dowdy et Foster, 82] :

- Performance : la stratégie d'allocation est conçue pour maintenir une mesure de performance. Deux objectifs bien connus sont de minimiser le temps de réponse et de maximiser le débit du système sur chaque site.
- Coût minimal : La fonction de coût comprend :
    - Le coût du stockage de chaque Fi sur un site Sj ;
    - Le coût de l'interrogation de F_i sur le site S_j ;
    - Le coût de la mise à jour de F_i sur tous les sites où elle est stockée ;
    - Et le coût de la communication de données.

Özsu et Patrick ont proposé un modèle d'allocation qui tente de minimiser le coût total du traitement et du stockage tout en essayant de respecter certaines restrictions de temps de réponse. Le modèle a la forme suivante : min (coût total) sous réserve de contrainte de temps de réponse, de contrainte de stockage et de contrainte de traitement.

3.3. Répertoire de données

Le schéma de la base de données distribuée doit être stocké et maintenu par le système. Ces informations sont nécessaires lors de l'optimisation des requêtes distribuées. Les informations de

schéma sont stockées dans un catalogue / dictionnaire de données / répertoire. Un répertoire est une base de métadonnées qui stocke un certain nombre d'informations telles que des définitions de schéma et de mappage, des statistiques d'utilisation, des informations de contrôle d'accès, etc.

Dans le cas d'un SGBD distribué, la définition du schéma se fait au niveau global (c'est-à-dire le schéma conceptuel global (GCS)) ainsi qu'au niveau des sites locaux (c'est-à-dire les schémas conceptuels locaux (LCSs)). GCS définit la base de données globale tandis que chaque LCS décrit les données sur ce site particulier. Par conséquent, il existe deux types de répertoires :

- Répertoire global (GD) qui décrit le schéma de base de données tel que les utilisateurs finaux le voient ;
- Et répertoire local (LD) qui décrit les mappages locaux et décrits le schéma de chaque site.

Ainsi, les composants de gestion de base de données locale sont intégrés au moyen de fonctions globales de SGBD [Özsu, 11].

4. Intégration de bases de données

L'intégration des bases de données est un processus technique permettant de combiner des données provenant de sources disparates pour visualiser les données en une seule vue unifiée.

L'intégration de la base de données peut être physique ou logique. Dans le premier, les bases de données sources sont intégrés et la base de données intégrée est matérialisée. Ceux-ci sont connus sous le nom d'entrepôts de données. L'intégration est facilitée par des outils d'extraction transformation chargeme (ETL) qui permettent l'extraction de données à partir de sources, sa transformation pour correspondre au GCS et son chargement (c'est-à-dire la matérialisation) [Wrembel, 06]. Dans l'intégration logique, le schéma conceptuel global (ou médiatisé) est entièrement virtuel et non matérialisé.

Ces deux approches sont complémentaires et répondent à des besoins différents. L'entreposage de données prend en charge les applications d'aide à la décision, appelées traitement analytique en ligne (OLAP). Les applications OLAP analysent des données historiques résumées provenant d'un certain nombre de bases de données opérationnelles au moyen de requêtes complexes sur des tables potentiellement très volumineuses. Par conséquent, les entrepôts de données collectent des données à partir d'un certain nombre de bases de données

opérationnelles et les matérialisent. Quand on effectue des mises à jour au niveau de bases de données opérationnelles, elles sont diffusées vers l'entrepôt de données, ce que l'on appelle la maintenance de la vue matérialisée [Vangenot, 03].

Dans le cas de l'intégration de données logiques, les données résident dans les bases de données opérationnelles et le GCS fournit une intégration virtuelle pour interroger les multiples bases de données. Dans ces systèmes, le GCS peut être défini de bas en haut en intégrant des parties des LCSs des bases de données locales. D'autre part, les requêtes des utilisateurs sont posées sur le schéma global, qui sont ensuite décomposés et envoyés aux bases de données opérationnelles locales pour traitement comme cela se fait dans des systèmes étroitement intégrés, la principale différence étant l'autonomie et l'hétérogénéité des systèmes locaux [Wiederhold, 92].

4.1. Méthodologie de la conception ascendante (Bottom-Up)

La conception ascendante implique le processus qui permet d'intégrer physiquement ou logiquement les informations des bases de données participantes pour former une seule multi base de données cohérente. La figure I.5 représente le processus d'intégration est divisé en plusieurs étapes.

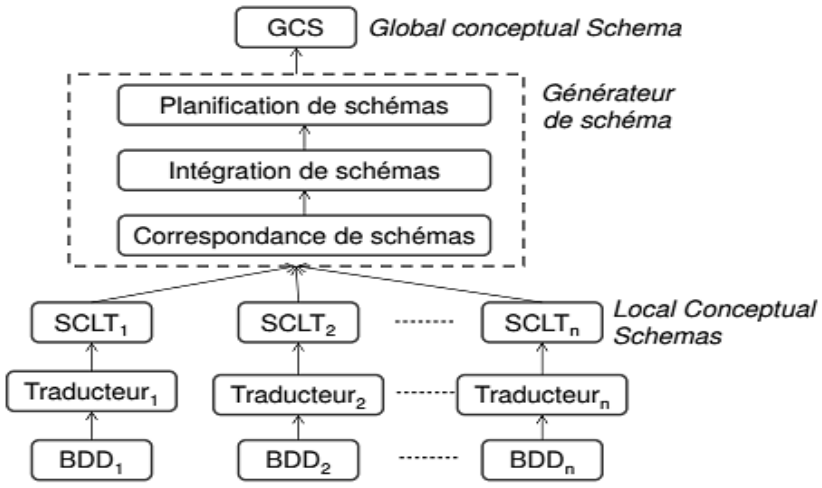


Figure I.5. Processus d'intégration des bases des données [Özsu, 11].

Comme indiqué sur la figure, le processus de génération de schémas est composé de trois étapes : correspondance de schémas, intégration de schémas et Planification de schémas.

4.1.1. Correspondance de schémas

Elle permet de déterminer les correspondances syntaxiques et sémantiques entre les éléments traduits. Si le schéma conceptuel global (GCS) est défini, les éléments traduits doivent lui correspondre, sinon on établit une correspondance entre les schémas conceptuels locaux traduits [guezouli, 22].

Les correspondances qui sont définies ou découvertes lors de la mise en correspondance de schémas sont spécifiées comme un ensemble de règles où [Devogele, 95] et [Özsu, 11] :

- Chaque règle (r) identifie une correspondance (c) entre deux éléments, telle que la correspondance (c) définit un élément en fonction d'un autre soit par égalité s'ils sont similaires, ou par une expression.
- Un prédicat (p) où la condition qui indique quand la correspondance peut se tenir, quand elle est vraie on applique la règle, sinon la règle sera ignorée.
- Une valeur de similarité (s) entre les deux éléments identifiés dans la correspondance qui est une valeur réelle $\in$ [0,1]. Elle peut être calculée ou spécifiée. Si les deux éléments comparés sont similaires (s = 1) et la correspondance se fera par égalité, sinon la correspondance se feront par modification d'au moins un des deux éléments.

4.1.2. Intégration de schémas

Dans le processus d'intégration de schémas la participation humaine est clairement primordiale pour définir les critères de construction de la description intégrée.

Cette étape consiste à créer le GCS, ce que l'on appelle l'intégration de schéma. Si le GCS était défini à l'avance, l'étape d'appariement déterminerait les correspondances entre celui-ci et chacun des LCS et il n'y aurait pas besoin de l'étape d'intégration. Si le GCS est créé à la suite de l'intégration de LCSs sur la base des correspondances identifiées lors de la correspondance de schéma, alors, dans le cadre de l'intégration, il est important d'identifier les correspondances entre le GCS et le LCSs [Devogele, 95] et [Özsu, 99].

4.1.3. Planification de schémas

Le processus de planification de schémas est divisé en deux étapes : création de planification et maintenance de planification [guezouli, 22].

- La création de planification consiste à créer des requêtes permettant de placer les données provenant des schémas locaux dans le schéma global.
- La maintenance de planification permet de détecter et de corriger les incohérences générées par l'évolution du schéma.

5. Traitement de requête distribue

Le traitement des requêtes est plus difficile dans les environnements distribués que dans les environnements centralisés, car il existe un grand nombre de paramètres, qui affectent les performances des requêtes distribuées. En particulier, les requêtes distribuées peuvent être fragmentées et/ou répliquées, induisant ainsi des frais généraux de communication. De plus, avec de nombreux sites auxquels accéder, le temps de réponse aux requêtes peut devenir très haut [Chevance, 01].

L'objectif du traitement d'une requête distribuée est de transformer une requête de haut niveau sur une base de données réparties en une stratégie d'exécution efficace exprimée en langage de bas niveau sur les bases de données locales. Typiquement, le traitement de requête passe en quatre phases de décomposition de requêtes, de localisation de fragments, d'optimisation globale, d'optimisation locale et d'exécution. La figure I.6 illustre ce processus [Chevance, 01].

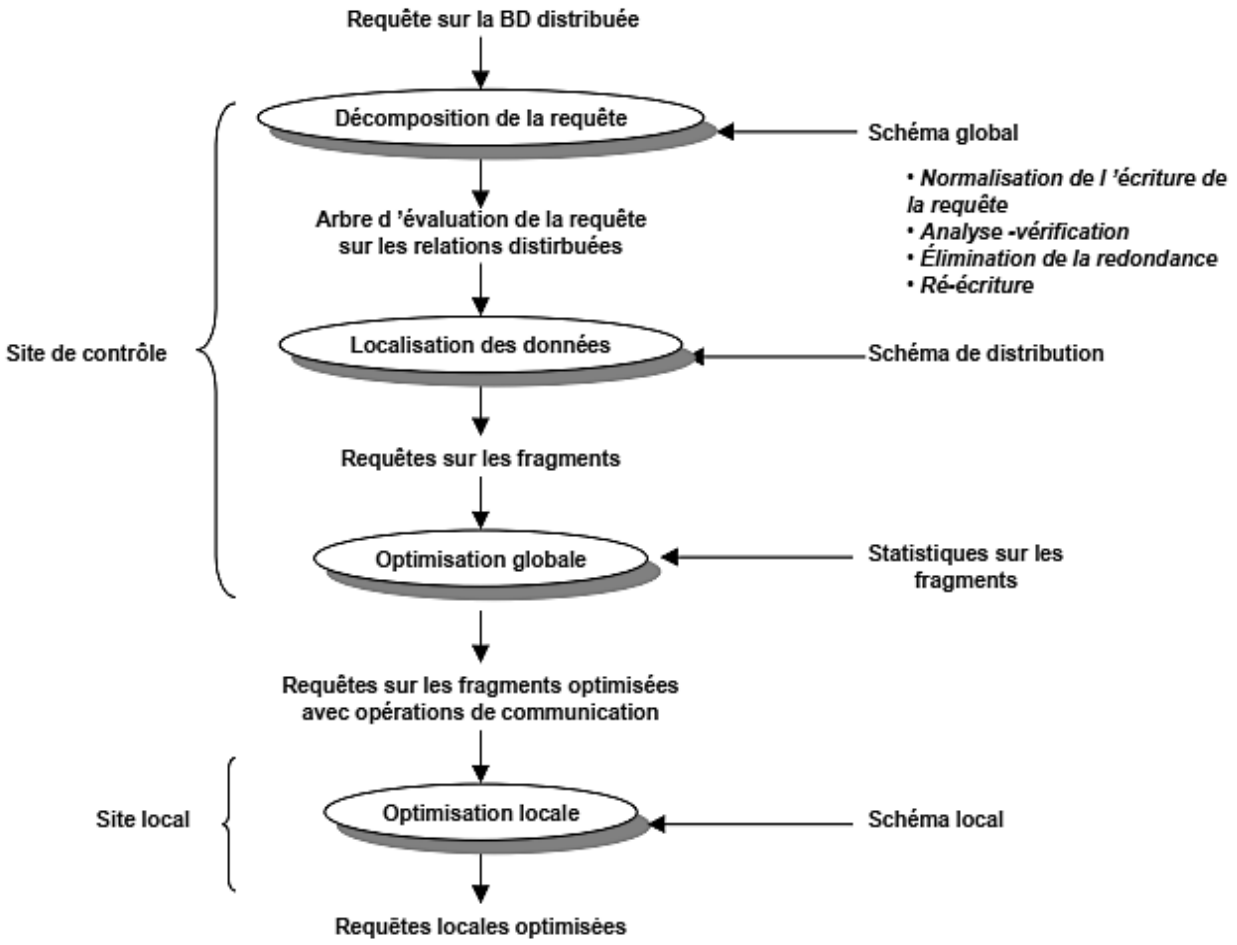


Figure I.6. Processus de traitement de requête distribuée [Chevance, 01].

5.1. Décomposition de la requête

La décomposition des requêtes peut être prise en compte en quatre étapes successives [Özsu, 11] :

- Tout d'abord, la requête de calcul est réécrite sous une forme normalisée qui convient à une manipulation ultérieure. La normalisation d'une requête implique généralement la manipulation des quantificateurs de requête et de la qualification de requête en appliquant une priorité d'opérateur logique.
- Deuxièmement, la requête normalisée est analysée sémantiquement pour les requêtes incorrectes soient détectées et rejetées le plus tôt possible. Une requête est sémantiquement incorrecte si ses composants ne contribuent en aucune façon à la génération du résultat. Dans le contexte du calcul relationnel, il est impossible de déterminer l'exactitude sémantique des requêtes générales. Cependant, il est possible de le

faire pour une grande classe de requêtes relationnelles, c'est-à-dire celles qui ne contiennent pas de disjonction et de négation. Ceci est basé sur la représentation de la requête sous forme de graphique, appelé graphique de requête ou graphique de connexion. Nous définissons ce graphique pour les types de requêtes les plus utiles impliquant des opérateurs de sélection, de projection et de jointure.

- Troisièmement, la requête correcte est simplifiée. Une façon de simplifier une requête consiste à éliminer les prédicats redondants. Il faut noter, que des requêtes redondantes sont susceptibles de se produire lorsqu'une requête est le résultat de transformations système appliquées à la requête utilisateur, de telles transformations sont utilisées pour effectuer un contrôle de données distribuées.
- Quatrièmement, la requête de calcul est restructurée en requête algébrique. La qualité d'une requête algébrique est définie en matière de performances attendues. La façon traditionnelle de faire cette transformation vers une meilleure spécification algébrique est de commencer par une requête algébrique initiale et de la transformer pour en trouver une bonne. La requête algébrique initiale est dérivée immédiatement de la requête de calcul en traduisant les prédicats et l'instruction cible en opérateurs relationnels tels qu'ils apparaissent dans la requête. Cette requête d'algèbre directement traduite est ensuite restructurée à l'aide de règles de transformation. La requête algébrique générée par cette couche est bonne dans le sens où les pires exécutions sont généralement évitées. Par exemple, une relation ne sera accessible qu'une seule fois, même s'il y a plusieurs prédicats sélectionnés.

Cependant, cette requête est généralement loin de fournir une exécution optimale, car les informations sur la distribution des données et l'allocation des fragments ne sont pas utilisées dans cette couche.

5.2. Localisation de données

L'entrée de la deuxième couche est une requête algébrique sur les relations globales. Le rôle principal de la deuxième couche est de localiser les données de la requête à l'aide des informations de distribution des données dans le schéma de fragments.

Cette couche détermine quels fragments sont impliqués dans la requête et transforme la requête distribuée en requête sur des fragments. La fragmentation est définie par des règles de

fragmentation qui peuvent être exprimées sous forme d'opérateurs relationnels. Une relation globale peut être reconstruite en appliquant les règles de fragmentation, puis en dérivant un programme, appelé programme de matérialisation, d'opérateurs d'algèbre relationnelle qui agit ensuite sur des fragments. La localisation comporte deux étapes [Özsu, 11] :

- Tout d'abord, la requête est mappée en une requête de fragments en substituant chaque relation par son programme de matérialisation ;
- Et deuxièmement, la requête de fragments est simplifiée et restructurée pour produire une autre "bonne" requête. La simplification et la restructuration peuvent se faire selon les mêmes règles que celles utilisées dans la couche de décomposition.

Comme dans la couche de décomposition, la requête de fragments finaux est généralement loin d'être optimale car les informations concernant les fragments ne sont pas utilisées.

5.3. Optimisation globale des requêtes distribuées

L'entrée de cette couche est une requête algébrique sur des fragments, c'est-à-dire une requête de fragments.

L'optimisation des requêtes consiste à trouver le meilleur ordre des opérateurs dans la requête, y compris les opérateurs de communication, qui minimise une fonction de coût. La fonction de coût, souvent définie en matière d'unités de temps, fait référence à l'utilisation de ressources informatiques telles que le disque, le coût du processeur et le réseau. Généralement, il s'agit d'une combinaison pondérée de coûts d'E/S, de CPU et de communication.

La détermination des coûts d'exécution avant l'exécution de la requête, c'est-à-dire l'optimisation statique, est basée sur les statistiques de fragments et les formules d'estimation des cardinalités des résultats des opérateurs relationnels. Ainsi, les décisions d'optimisation dépendent de l'allocation des fragments et des statistiques disponibles sur les fragments qui sont inclus dans le schéma d'allocation.

5.4. Optimisation locale des requêtes distribuées

La dernière couche est réalisée par tous les sites qui ont des fragments impliqués dans la requête. Chaque sous-requête s'exécutant sur un site, appelée requête locale, est optimisée en utilisant le schéma local du site et exécuté. À ce stade, les algorithmes à effectuer les opérateurs

relationnels peuvent être choisis. L'optimisation locale utilise les algorithmes de systèmes centralisés [Chevance, 01].

L'implémentation classique des opérateurs relationnels dans les systèmes de bases de données est basée sur le modèle d'itérateur, qui fournit un parallélisme en pipeline dans les arbres d'opérateurs. Il est un modèle pull simple qui exécute des opérateurs à partir du nœud opérateur racine (qui produit le résultat) aux nœuds feuilles (qui accèdent aux relations de base). Ainsi, les résultats intermédiaires des opérateurs n'ont pas besoin d'être matérialisés car les tuples sont produit à la demande et peut-être consommé par les opérateurs ultérieurs. Cependant, il nécessite que les opérateurs soient implémentés en mode pipeline, en utilisant une open-next-close interface. Chaque opérateur doit être implémenté comme un itérateur avec trois fonctions :

1. Open () : initialise l'état interne de l'opérateur, par exemple, allouer une table de hachage ;
2. Next () : produit et renvoie le tuple de résultat suivant ou null ;
3. Et Close () : nettoient toutes les ressources allouées, une fois que tous les tuples ont été traités.

Ainsi, un itérateur fournit la composante d'itération d'une boucle while, c'est-à-dire l'initialisation, l'incrémentation, la condition de fin de boucle et le nettoyage final. Un aspect important de l'optimisation des requêtes est l'ordre des jointures, car les permutations des jointures au sein de la requête peuvent entraîner des améliorations d'ordre de grandeur.

La sortie de la couche d'optimisation de requête est une requête algébrique optimisée avec des opérateurs de communication incluse sur des fragments. Il est généralement représenté et enregistré (pour les exécutions futures) [Özsu, 11].

6. Conclusion

Dans ce chapitre nous avons présenté les principales notions et concepts des bases de données distribuées. Puis, nous avons abordé à la structure de ce type des bases de données. Ensuite, nous avons expliqué les étapes de l'intégration des bases de données. Enfin, nous avons présenté de traitement de requête distribuée.

Dans le prochain chapitre nous commencerons à explorer un autre point important dans notre sujet de thèse en présentant les notions principales d'extraction des motifs fréquents.

Chapitre II

Extraction des Motifs Fréquents

1. Introduction
2. Processus de découverte des connaissances
   - 2.1. Définition de fouille de données
   - 2.2. Quels types de données à fouiller ?
   - 2.3. Les tâches et outils de fouille de données
     - 2.3.1. Les méthodes de visualisation et de description
     - 2.3.2. Les méthodes de classification et de structuration
     - 2.3.3. Les méthodes d'explication et de prédiction
3. Généralités sur le datamining distribué
   - 3.1. Pourquoi le datamining distribué ?
   - 3.2. Les données dans le datamining distribué
   - 3.3. Techniques de placement des tâches et des données
     - 3.3.1. Placement statique
     - 3.3.2. Placement dynamique
   - 3.4. Outils du calcul parallèle
     - 3.4.1. Les threads ou les fils
     - 3.4.2. Echange de messages
     - 3.4.3. Mesures de performance des programmes parallèles
4. Présentation générale des algorithmes d'extraction des itemsets fréquents
   - 4.1. Terminologie et notations
   - 4.2. Les algorithmes d'extraction des itemsets fréquents
   - 4.3. Algorithmes d'extraction distribuée des itemsets fermés fréquents
5. Conclusion

1. Introduction

Le data mining est un terme générique pour désigner une famille d'outils d'analyse particulièrement adaptés à l'exploitation des grandes masses de données. Le datamining permet notamment de rechercher des structures difficilement identifiables et des corrélations peu perceptibles avec les techniques d'analyses statistiques plus classiques.

Dans le cas de distribution de données sur diverses sources de données distribuées et leurs tailles importantes rendent les algorithmes du data mining classique inadéquat pour leurs traitements devenus de plus en plus complexes. Pour résoudre ce problème, on a vu apparaître la notion du "datamining distribué".

Dans ce chapitre, on commence par présenter le processus d'extraction des connaissances où l'on définit les notions de data mining, de tâches et d'outils de data mining. Ensuite, on présentera des généralités sur data mining distribué. Et enfin, on expliquera et discutera les algorithmes d'extraction des items sets fréquents distribués.

2. Processus de découverte de connaissances

L'eExtraction des Connaissances à partir des Données (ECD) est un processus itératif et interactif d'analyse d'un grand ensemble de données brutes pour extraire des connaissances exploitables par un utilisateur analyste qui y joue un rôle central. Ce processus est constitué de quatre phases qui sont : le nettoyage et l'intégration des données, le prétraitement des données, la fouille de données et enfin l'évaluation et la présentation des connaissances [Agard, 05].

L'étape principale du processus ECD est l'étape de data mining tel que l'opérationnalisation du processus ECD est réalisée par la fouille de données qui fait appel à des multiples méthodes d'issue des statistiques, de l'apprentissage automatique, de la reconnaissance des formes ou de la visualisation [Fayyad, 01] comme illustrée par la figure II.1.

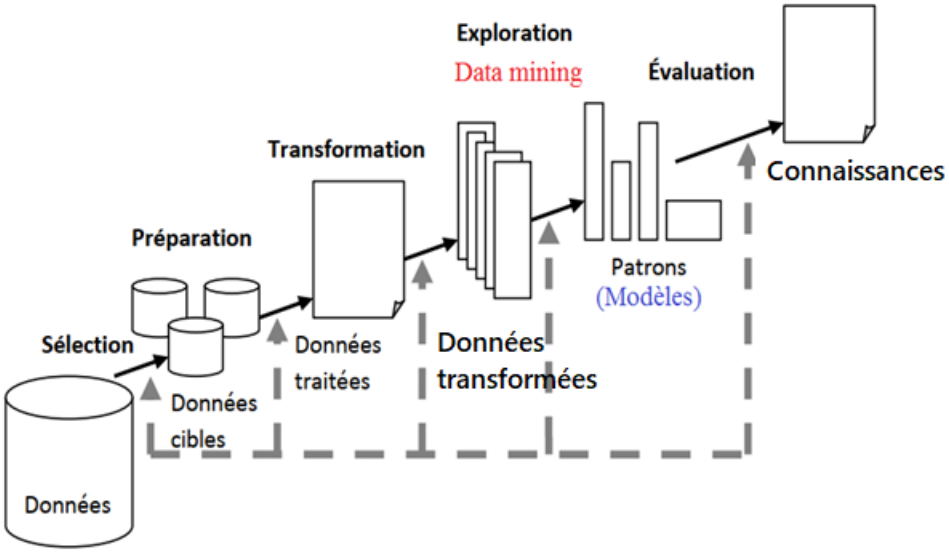


Figure II. 1. Processus d'extraction de connaissances à partir des données.

2.1. Définition de fouille de données

Le data mining ou la fouille de données, est l'ensemble des méthodes et des techniques destinées à l'exploration et l'analyse de bases de données informatiques (souvent très grandes), de façon automatique ou semi-automatique, en vue de détecter dans ces données des règles, des associations, des tendances inconnues ou cachées, et des structures particulières restituant l'essentiel de l'information utile tout en réduisant la quantité de données [Chaabane, 13].

Il s'agit de "fouille" visant à découvrir "de l'information cachée" que les données renferment et que l'on découvre à la recherche d'associations, de règles de classification et d'autres types de connaissances qui serviront à l'aide à la décision. Le datamining peut accomplir différentes tâches avec différents outils. C'est en résumé un processus itératif et interactif qui permet de découvrir de nouvelles connaissances, valides, utiles et compréhensibles, à partir de grandes masses de données [Fayyad, 01].

2.2. Quel type de données à fouiller ?

Le composant de base d'un processus de data mining est l'ensemble d'échantillons représentant les données à explorer. Chaque échantillon est présenté sous forme de ligne caractérisée par un ensemble d'attributs. Dans le cas des bases de données un échantillon est un enregistrement composé d'un ensemble de champs. Généralement, il convient de représenter les

enregistrements sous forme de points dans un espace de m dimensions où m est le nombre d'attributs [Djeffal, 14] comme sur la figure II.2.

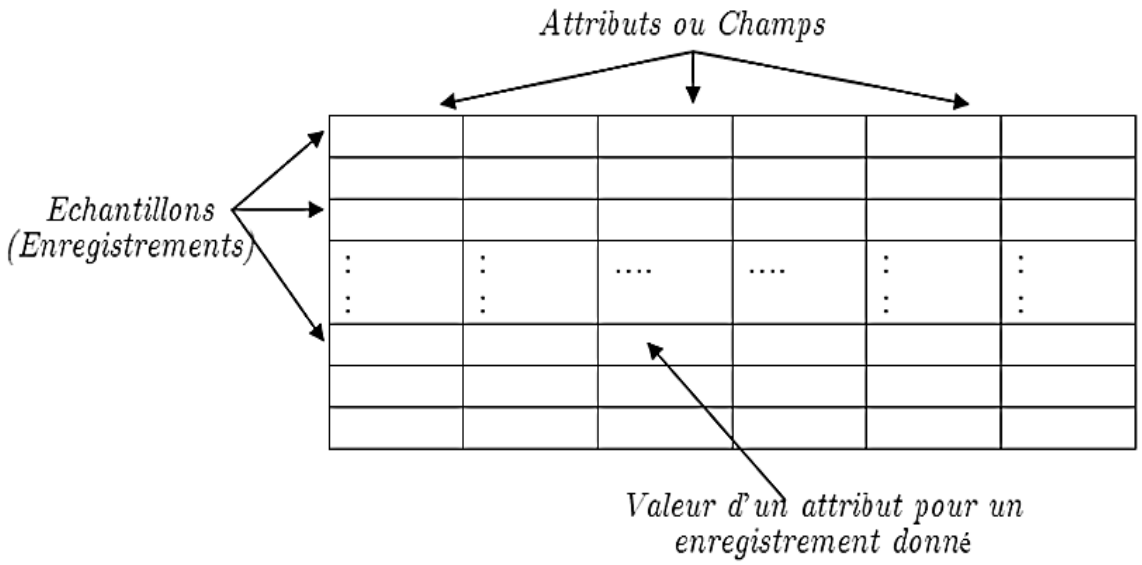


Figure II. 2. Le composant de base d'un processus de data mining [Djeffal, 14].

Théoriquement, plus le nombre d'échantillons est important, meilleur est la précision de l'analyse. Mais en pratique, beaucoup de difficultés peuvent être rencontrées avec les bases de données gigantesques (des milliards d'enregistrements ou des Gigas bytes). En effet, les bases de données de nos jours sont immenses au point où elles épuisent même les supports de stockage, et nécessitent pour être analysées les machines les plus puissantes et les techniques les plus performantes. Un premier problème avec les bases de données immenses, est celui de leur préparation à l'analyse, puisque la qualité des données analysées influence directement sur les résultats d'analyse. La préparation doit prendre compte d'un certain nombre de points [Djeffal, 14] :

- Les données doivent être précises : les noms doivent être écrits correctement, les valeurs doivent être dans les bons intervalles et doivent être complètes ;
- Les données doivent être enregistrées dans les bons formats : une valeur numérique ne doit pas être enregistrée sous format caractère, une valeur entière ne doit pas être réelle, ... ;
- Et la redondance doit être éliminée voir au moins minimisée.

Dans le cas d'un nombre limité d'échantillons, la préparation peut être semi-automatique ou même manuelle, mais dans notre cas d'immenses BDD la préparation automatique s'impose et des techniques automatiques de vérification et de normalisation doivent intervenir. Le problème majeur des BDD immenses apparaît lors de leur exploration, le temps d'exploration et la précision doivent être pris en compte.

Les enregistrements sont regroupés dans des tables et dans des bases de données de différents types et l'analyse effectuée et le choix de ses outils dépendent fortement du type de la base de données à analyser. En fait, les types bases de données les plus utilisés sont : les bases de données relationnelles, les bases de données transactionnelles, les systèmes avancés de bases de données ainsi que les bases de données spatiales [Djeffal, 14].

2.3. Les tâches et méthodes de fouille de données

La fouille de données est le cœur du processus d'ECD. Cette phase fait appel à de multiples méthodes issues des statistiques, de l'apprentissage automatique, de la reconnaissance des formes ou de la visualisation. Ces méthodes peuvent être divisées en trois catégories [Berry, 04] et [Zighed, 02] :

- Méthodes de visualisation et de description ;
- Méthodes de classification et de structuration ;
- Et méthodes d'explication et de prédiction.

2.3.1. Les méthodes de visualisation et de description

L'objectif de ces méthodes est de permettre à l'analyste d'avoir une compréhension synthétique de l'ensemble de ses données. Il s'agit donc principalement d'outils de synthèse d'informations. Cette synthèse peut s'exprimer par des indicateurs statistiques. Ce type comporte les méthodes de la représentation graphique [Berry, 04].

Le schéma de la figure II.3 représente un processus de data Mining orienté vers la visualisation et la description. Pour simplifier la présentation, le tableau des données (CREDITS) ne contient que 1000 clients [Zighed, 02].

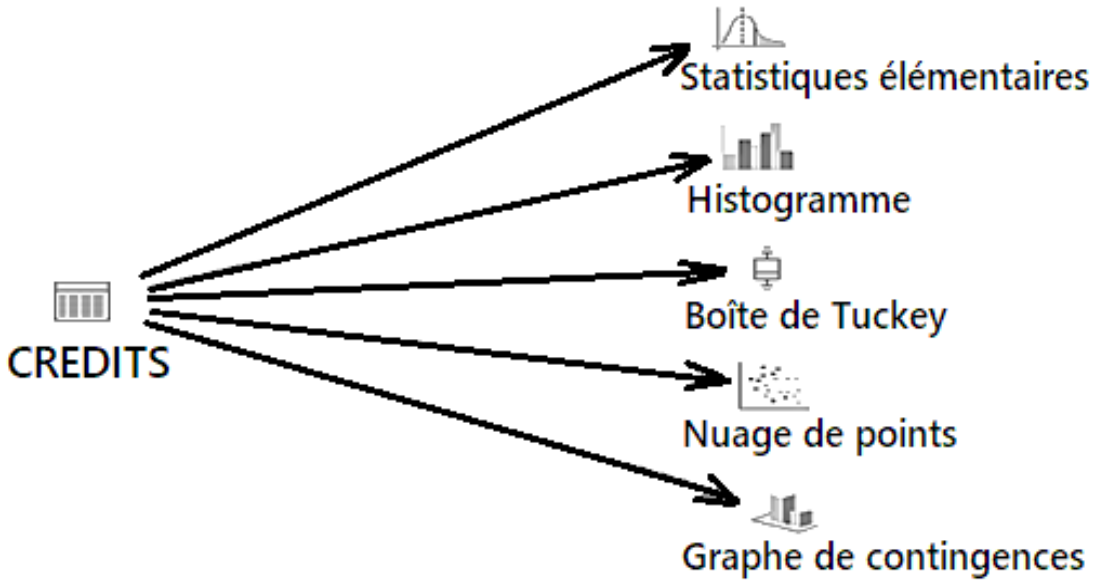


Figure II. 3. Traitement de description et de visualisation [Zighed, 02].

2.3.2. Les méthodes de classification et de structuration

L'objectif de ces méthodes est de permettre à identifier des groupes d'objets semblables au sens d'une métrique donnée, dont le clustering des objets en se basant sur leurs similarités. Ces types de méthodes utilisent l'apprentissage non supervisé car l'utilisateur ne sait pas a priori quelles classes, quels groupes ou quelles catégories il va obtenir ? Les principales techniques se répartissent en trois groupes [Zighed, 02] comme sur la figure II.4 :

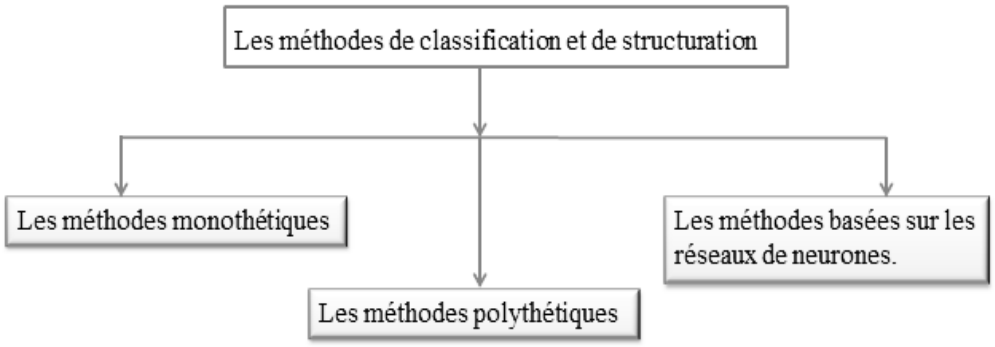


Figure II. 4. Les groupes de principales techniques des méthodes de classification et de structuration.

2.3.3. Les méthodes d'explication et de prédiction

Ces méthodes permettent de prédire si un élément est membre d'un groupe ou d'une catégorie donnée parce qu'elles utilisent l'apprentissage supervisé. Le processus de ces méthodes a deux étapes [Zighed, 02] :

- La première étape de construction du modèle à partir de l'ensemble d'apprentissage ;
- Et la deuxième étape utilisation du modèle, qui permet de tester la précision du modèle et l'utiliser dans la classification de nouvelles données. La figure II.5 représente les techniques utilisées dans ce type de méthodes.

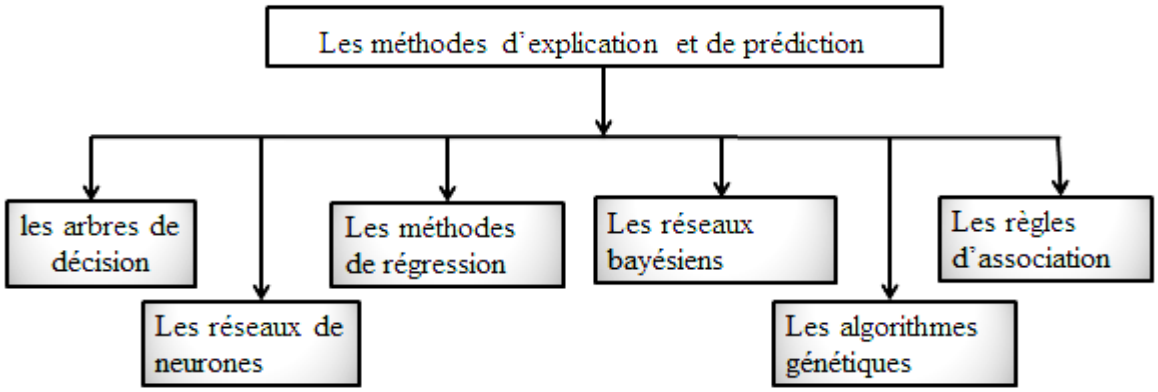


Figure II. 5. Les techniques utilisées dans les méthodes d'explication et de prédiction.

3. Généralités sur le datamining distribué

Le data Mining distribué est son application sur différentes architectures homogènes ou hétérogènes en utilisant les différents types de programmation parallèle ou distribuée. Formellement, il y a peu de différences entre les calculs parallèles et les calculs distribués : dans le datamining distribué, les calculs sont distribués et la communication se fait à travers le passage de messages ; tandis que dans le datamining parallèle, le calcul est parallèle sur des processeurs qui partagent la mémoire et/ou le disque dur [Fatiha, 11].

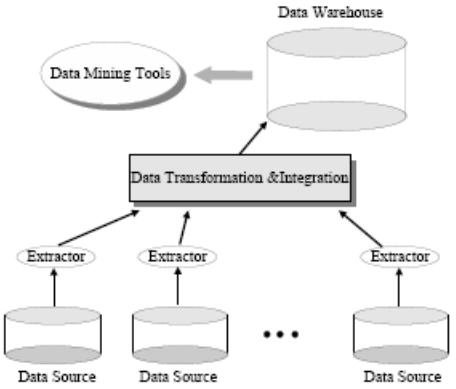
3.1. Pourquoi le datamining distribué ?

Le processus de découverte de connaissances dans des environnements distribués nécessite la collecte de données distribuées dans un entrepôt de données central pour le traitement comme

l'illustre la figure II.6.a. Cependant, elle est généralement soit inefficace, soit irréalisable pour les causes suivantes [Fatiha, 11] et [Park, 02] :

- **Coût de communication :** En raison de la nature distribuée des données et des ressources il existe certaines applications sont basées sur des données distribuées avec la nécessité d'obtenir un aperçu global à partir de l'ensemble de données distribuées. Le transfert de grandes quantités de données sur un réseau peut prendre beaucoup de temps aussi exiger une charge financière importante. Même un petit volume de données peut créer des problèmes dans les environnements réseau sans fil avec une largeur de bande limitée. Également, la communication peut être coûteuse, car les bases de données distribuées ne sont pas toujours constantes.
- **Coût de calcul :** Le coût de calcul à extraire les informations d'un entrepôt de données est plus grand que la somme du coût d'analyse des plus petites parties des données qui pourraient également être faites en parallèle.
- **Des données privées sensibles :** Il existe un ensemble d'application data mining qui traite des données sensibles, par exemple les dossiers médicaux et les dossiers financiers. Dans ces cas, on ne peut pas faire la collection centrale de ces données car elle met leur vie privée en danger. Dans certains cas (tels que les banques et les télécommunications) les données sont susceptibles d'appartenir à différents organismes qui veulent échanger des connaissances sans échanger des informations confidentielles brutes.
- **Efficacité et passage à l'échelle** : Avec la grande taille de données (Tera - octet), la question du passage à l'échelle « scalabilité » consiste à voir si l'algorithme peut traiter efficacement une grande masse de données. À travers cette distribution, on va construire les meilleurs modèles possibles [Mokeddem, 17].

On a besoin des architectures data mining qui donne une sollicitude particulière aux ressources distribuées de données, calcul, et communication pour réduire la charge de communication et réduire aussi l'énergie pour optimiser leur consommation. Le Data mining distribué (DDM) considère le data mining dans le plus large contexte comme indiqué dans les figure II.6b.



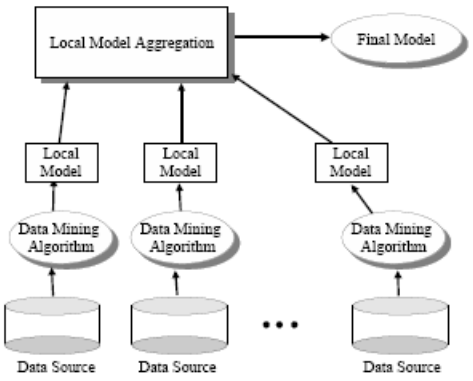


Figure II. 6a: Data Mining centralisé. **Figure II.6b :** Data Mining distribute [Fatiha, 11].

3.2. Les données dans le datamining distribué

La première étape dans le développement d'une solution du datamining distribué est l'identification de la manière dont les données sont distribuées. La plupart des algorithmes étant conçus pour le modèle relationnel, la table de données globales - qui n'a pas nécessairement une existence physique - est partitionnée d'une façon horizontale, verticale ou mixte [Mokeddem, 17].

3.3. Techniques de placement des tâches et données

Il existe des techniques pour placer les tâches et les données. Ce placement peut être statique ou dynamique.

3.3.1. Placement statique

Les tâches sont réparties entre les processus avant l'exécution de l'algorithme. Pour le partitionnement de données, il existe deux stratégies de placement statique [Grama, 03] :

- **Placement par bloc :** est une des méthodes les plus simples pour distribuer un tableau ou une matrice. On attribue des partitions uniformes contigües aux différents processeurs (processus).
- **Placement par bloc cyclique :** si la quantité de travail est différente dans les processeurs, le placement par bloc peut conduire à un déséquilibre de charge. On attribue les partitions (et les tâches associées) aux processeurs d'une manière cyclique de telle sorte que chaque processeur reçoit plusieurs blocs non adjacents. La raison pour laquelle le placement par

bloc cyclique est en mesure de réduire le déséquilibre de charge, est que tous les processus auront un échantillon de tâches et de données issues de différentes parties de la matrice. Par conséquent, même si les différentes parties de la matrice ont besoin de différentes quantités de travail, l'ensemble des travaux sur chaque processus s'équilibre.

3.3.2. Placement dynamique

C'est la répartition des tâches entre les processeurs durant l'exécution de l'algorithme. Si les tâches sont générées dynamiquement, elles doivent être placées dynamiquement. Si la taille de la tâche n'est pas connue, un placement statique peut potentiellement conduire à un déséquilibrage de charge, pour cela les placements dynamiques sont généralement plus efficaces. Les techniques de placement dynamique sont généralement classées en deux classes : centralisées ou distribuées [Grama, 03].

3.4. Outils du calcul parallèle

Les outils de data mining distribué est les mêmes que ceux dédiés au calcul parallèle. Parmi ces techniques les plus utilisées : les threads et l'échange de messages.

3.4.1. Les threads ou les fils (en français)

Un Thread ou « processus léger » est l'unité élémentaire de concurrence traitée par les systèmes d'exploitation. C'est un chemin d'exécution à l'intérieur d'un processus. Les threads d'un même processus partagent la mémoire virtuelle, ce qui rend les changements de contexte peu coûteux en temps. On distingue trois façons de structurer un programme utilisant des threads [Grama, 03] :

- **Maître-Esclave :** un thread maître rassemble et génère les tâches qui doivent être accomplies, et répartit ces taches sur des threads esclaves. Ce modèle est courant dans les programmes serveurs et à interface graphique.
- **Équipe de travail :** plusieurs threads traitent de la même façon les différentes parties de données. Cela reproduit le mode de travail du calcul parallèle classique et des processeurs vectoriels.
- **Travail à la chaîne :** ce modèle divise une tâche en une suite de tâches, et consiste à passer les résultats d'une étape au thread qui s'occupe de l'étape suivante.

3.4.2. Échange de messages

Un processus peut avoir plusieurs threads qui partagent le même espace d'adressage (mémoire partagée). Le modèle de passage de message est destiné à faire communiquer plusieurs processus qui ont des espaces d'adressage différents (mémoire distribuée). La communication consiste à des synchronisations entre processus ainsi que des mouvements de données d'un espace d'adressage à un autre. Ce modèle est offert à travers plusieurs outils tels que : le standard MPI (Message Processors Interface) et PVM (Parallel Virtual Machine) [Mokeddem, 17].

3.4.3. Mesures de performance des programmes parallèles

Paralléliser un programme consiste à faire exécuter plusieurs parties du code séquentiel en parallèle. L'idéal est de faire une conception parallèle de l'algorithme dès le début, ce qui n'est pas encore offert par les approches informatiques actuelles. Pourtant, il existe un ensemble de critères qui peuvent mesurer l'efficacité et la performance du programme parallèle [Mokeddem, 17] :

- **Temps d'exécution :**

  Le temps d'exécution séquentiel d'un programme est le temps écoulé entre le début et la fin de son exécution sur un ordinateur séquentiel. Le temps d'exécution parallèle est le temps qui s'écoule entre le moment où un calcul parallèle commence et le moment où le dernier élément de traitement termine son exécution. On note le temps d'exécution séquentiel T_s et le temps d'exécution parallèle T_p (architecture parallèle comportant P processeurs).

- **Accélération (SpeedUp) :**

  L'accélération S_p quantifie le gain produit par l'utilisation de P processeurs dans le but de juger les performances de la méthode du parallélisme. Elle constitue le rapport entre le temps d'exécution séquentiel optimal T_s et le temps d'exécution parallèle T_p : $S_p = T_s / T_p$ avec $1 \leq S \leq P$

- **Efficacité (Efficiency) :**

  L'efficacité E_p indique le taux d'utilisation des processeurs utilisés dans l'architecture parallèle : $E_p = S_p / P$.

- **Facteurs limitant l'accélération et l'efficacité :**

  L'idéal en programmation parallèle est d'arriver à une accélération linéaire utilisant *P* processeurs et une efficacité égale à 1. L'objectif final est d'être P fois plus rapide en utilisant *P* processeurs. Cependant l'accélération S_p habituellement inférieure à *P* pour plusieurs raisons, entre autres :

  a. La partie du code du programme qui ne peut pas être parallélisée.
  b. Le coût de communication réseau : la quantité et la fréquence d'envoi des messages entre les processeurs influent sur l'accélération et le parallélisme.
  c. équilibrage de charge : l'accélération est généralement limitée par la vitesse du processeur le plus lent. Ainsi, il est important de s'assurer que chaque processeur effectue la même quantité de travail dans la mesure du possible.

4. Présentation générale des algorithmes d'extraction des itemsets fréquents

La tâche de la découverte des items sets fréquents est un problème important du Data Mining. Elle permet la découverte de règles intelligibles et exploitables dans un ensemble de données volumineux, règles exprimant des corrélations (associations) entre items ou attributs dans une base de données transactionnelles ou relationnelles.

On peut dire que les algorithmes d'extraction des motifs fréquents opèrent généralement sur des bases de données transactionnelles pour extraire une connaissance sous forme de corrélations qui peuvent exister entre les différents attributs de la base.

4.1. Terminologie et Notations

Il existe des notations et des définitions importantes pour décrire l'approche d'extraction des motifs fréquents :

- **Base de données transactionnelle** : un ensemble D = $\{t_1, t_2, \ldots, t_n\}$ t_i tel que i= *1* ... *n* est une transaction.
- **Item (motif)** : Correspond à une valeur d'un attribut dans la base *D*, Soit $I = \{i_1, i_2, \ldots, i_m\}$ un ensemble fini d'items.
- **Transaction** : une transaction t est un sous ensemble de *I*.
- **Itemset** : un itemset *X* est un ensemble d'items tel que $X \subset I$.

- **K-itemset** : est un itemset de taille *K* ou itemset avec *K* items.
- **Support d'un itemset** : Le support d'un itemset *X* noté *Supp(X)* est la proportion de transactions de *D* contenant *X* : *Supp(X) = Freq(X) / |D|* où *Freq(X)* est le nombre des transactions dans *D* qui contient l'itemset *X*.
- **Itemset fréquent** : On appelle *X* un itemset fréquent si *Supp(X)* est supérieur ou égal à un seuil minimal *Minsupp*.
- **Itemset fermé fréquent** : Un itemset *X* est dit fermé si et seulement s'il n'existe aucun itemset *Y* tel que *Supp(X) = Supp(Y)* et $X \subset Y$. Les itemsets fermés dont le support est supérieur au support minimal sont dits fermés fréquents.
- **Base conditionnelle** : Une base conditionnelle d'un itemset *X* est l'ensemble des transactions de la base de données contenant *X*. De manière formelle, une transaction d'identificateur *TID*, $t = (TID, Y)$ appartient à la base conditionnelle de X si et seulement si $X \subseteq Y$.
- **Règle d'association** : Implication de la forme $A \rightarrow B$, ou $A, B \subset I, A \cap B = \emptyset$ Avec *A* et *B* deux itemsets. *A* est dit prémisse (ou cause) et *B* est dit conclusion (ou conséquence).
- **Support d'une règle** : Le support d'une règle $R : A \rightarrow B$ noté *Supp(R)* est la proportion de transaction de *D* contenant $A \cup B$, il est calculé par $Supp(R) = Freq(A \cup B) / |D|$ où $Freq(A \cup B)$ est le nombre des transactions dans qui contiennent $A \cup B$.
- **Confiance d'une règle** : La confiance d'une règle $R : A \rightarrow B$ est la proportion de transactions de *D* contenant l'ensemble *A* qui contient aussi *B* : $conf(A \rightarrow B) = Freq(A \cup B) / Freq(A)$ où $Freq(A \cup B)$ est le nombre de transactions dans *D* contenant $A \cup B$. On voit immédiatement que la confiance se définit aussi par un rapport de support : $conf(A \rightarrow B) = sup(A \cup B) / sup(B)$.
- **Règle d'association fréquente** : On appelle *R* une règle d'association fréquente si *conf* (*R*) est supérieur ou égale à un seuil minimal *Minconf*.

4.2. Les algorithmes d'extraction des itemsets fréquents

Selon le principe, on peut diviser les algorithmes d'extraction des règles d'association en trois familles : algorithmes génèrent tous les items sets fréquents, algorithmes génèrent seulement les items sets fréquents maximaux, algorithmes génèrent les items sets fermés fréquents.

4.2.1. Algorithmes générant tous les itemsets fréquents

Dans ce cas on fournit les algorithmes A priori [Agrawal, 94] et FP-Growth [Han, 00].

- **A-priori :**

Le premier algorithme d'extraction des règles d'association est l'algorithme A priori qui diviser en deux phases [Agrawal, 94] :

1. Détermination des itemsets fréquents (*support* >= *minSupp*)
2. Dérivation des règles d'association valides (*confiance* >= *minconf*)

A priori se base essentiellement sur la propriété d'anti-monotonicité qui existe entre les items sets. Cette propriété est utilisée à chaque itération pour diminuer le nombre de motifs à considérer. Elle se résume comme suit :

- Tous les sous-ensembles d'un item set fréquent sont fréquents.
- Tous les sur ensembles d'un item set non fréquent sont non fréquents.

L'algorithme A priori est une séquence d'étapes à suivre pour trouver l'ensemble d'éléments le plus fréquent. Cette technique utilise deux étapes « joindre » et « élaguer » pour réduire l'espace de recherche suivre une approche itérative. Un seuil de support minimum est donné dans le problème où il est supposé par l'utilisateur.

- **FP-Growth**

Cet algorithme est une amélioration de la méthode A Priori. Han et al [Han, 00] ont proposé la méthode Fp-Growth (Frequent pattern Growth) qui permette de générer un modèle fréquent sans qu'il soit nécessaire de générer des candidats. Un modèle fréquent représente la base de données sous la forme d'un arbre appelé arbre de modèle fréquent ou arbre FP.

Fp-Growth consiste d'abord à compresser la base de données en une structure compacte appelée FP-tree, puis à diviser la base de données ainsi compressée en sous-projection de la base de données appelées bases conditionnelles. Chacune de ces projections est associée à un item fréquent. L'extraction des items sets fréquents se fera sur chacune des projections séparément. Ce processus est réalisé donc en deux étapes :

1. Représenter la base de données dans une structure compacte ″FP-tree″ après deux scans seulement.
2. Extraire les items sets fréquents en accédant à l'arbre FP-tree.

FP-tree est une structure arborescente constituée d'un arbre, d'un index ou d'une table de liens.

- Arbre : mise à part la racine nulle, chaque nœud de l'arbre contient trois informations : l'item qui représente ce nœud « itemName » sa fréquence « itemCount », ainsi que le nœud suivant dans l'arbre.
- Index : contient la liste des items fréquents. À chaque item est associé un pointeur indiquant le premier nœud de l'arbre contenant cet item.

Pour extraire les items sets fréquents Han propose des propriétés et des lemmes [Han, 00]. Un des principes de base est la notion de « base conditionnelle » : Si X et Y sont deux items, le support de $X \cup Y$ dans la base de données est exactement le support de Y si on restreint la base aux transactions contenant X. Cette restriction est appelée la base conditionnelle de X. L'arbre construit est appelé FP-tree conditionnel de X. On note que pour tout items Y fréquent dans la base conditionnelle de X, $X \cup Y$ est un itemset fréquent dans la base d'origine. Lorsque FP-tree contient un seul chemin, on arrête la récursivité et on déduit les items sets fréquents en calculant toutes les combinaisons possibles des items dans le chemin avec l'item en question.

- **FP-Growth vs A-Priori :**

Dans la table II.1, on fait une comparaison entre l'algorithme FP-Growth et A priori.

Table II. 1. FP-Growth vs A-Priori.

Algorithme	Génération de motifs	Génération de candidats	Traitement
A Priori	A Priori génère un motif en associant les éléments en singletons, paires et triplées.	A Priori utilise la génération de candidats.	Le processus est comparativement plus lent que FP Growth, le temps d'exécution augmente de façon exponentielle avec l'augmentation du nombre d'ensemble d'éléments.
FP-Growth	La croissance FP génère un modèle en construisant un arbre FP.	Il n'y a pas de génération candidate.	Le processus est plus rapide par rapport à A Priori. La durée d'exécution du processus augmente linéairement avec l'augmentation du nombre d'ensemble d'éléments.

Au contexte distribué :

Dans le contexte distribué, on peut résumer les algorithmes existant dans la table II.2.

Table II. 2. Les algorithmes dans le contexte distribué.

Algorithme	Développement	Traitement	Evaluation des performances
Count Distribution (CD) [Agrawal, 96]	Proposé par R. Agrawal et J.C Shafer en 1996, cet algorithme basé sur l'algorithme A Priori est réalisé dans un environnement distribué.	• Chaque site traite sa partition locale de la base de transactions. • Calcule les supports locaux des candidats et les diffuse aux autres sites pour calculer le support global. • Les items sets fréquents sont déduits à partir de ce calcul global.	• N'exploite pas la mémoire globale du système de manière efficace. • La duplication de l'ensemble entier des candidats au niveau de chaque nœud.
Distributed Meaning of Association rules (DMA) [Cheung,96]	En 1996, Cheung et al. a proposé l'algorithme DMA comme une parallélisassions de l'algorithme A Priori.	• Introduit une technique d'élagage au niveau de chaque site et ne garde que les items sets fréquents. • Emploie les supports locaux des items sets fréquents sur chaque site pour décider si un itemset fréquent est lourd (localement fréquent dans une partition et globalement fréquent dans la base de données entière). • Génère ensuite les candidats à partir des items sets fréquents lourds. • Chaque item est associé un site du vote qui est le responsable des collectes de ses supports locaux pour éviter la duplication de messages pour un même itemset.	• Le nombre des itemset générés au DMA est plus réduit que CD à cause d'utiliser des items sets lourds qui sont localement et globalement fréquents dans un site. • DMA réalise une réduction dans le nombre des messages échanges de $O(n^2)$ à $O(n)$ (n'est le nombre de sites) grâce à la technique d'optimisation de communication qui consiste à déterminer un site du vote pour chaque item.
Fast Distributed Mining (FDM) [Cheung, 1996, Dec]	Daving Cheung et al. ont proposé l'algorithme FDM pour l'extraction des règles d'associations. Il est basé sur l'algorithme A	Après avoir généré l'ensemble global des candidats $CG_{(k)}$, le site : • échanger les supports de cet ensemble entre les sites pour trouver les items sets globalement fréquents. • Observer que certains ensemble de candidats dans $CG_{(k)}$ peuvent être élagués en utilisant des informations locales avant	• L'algorithme FDM minimise ainsi l'ensemble des candidats générés en se basant sur deux techniques l'élagage local et l'élagage global. • FDM réduit substantiellement le

	Priori.	que l'échange des supports commence. • L'élagage local consiste alors à supprimer l'élément X de l'ensemble de candidats sur le site S_i si X n'est pas localement fréquent par rapport à sa base de données locale. • Après l'élagage local, chaque site diffuse les supports des candidats restant aux autres sites. • A la fin, chaque site somme ces supports locaux pour générer les items sets globalement fréquents par rapport à la base de données globale, cette technique est appelée l'élagage global.	nombre de messages transmis entre les sites et le temps d'exécution par apport les algorithmes précédents.
Parallel FP-Growth [Li, 08]	Li et ses groupes ont proposé de paralléliser l'algorithme FP-Growth qui appelle Parallel FP-Growth (PFP) sur des machines distribuées.	PFP utilissait la technique de MapReduce pour paralléliser PF-Growth en cinq étapes : • Sharding : diviser de la base de données en parties successives et stocker les parties sur P ordinateurs différents. Cette division et distribution de données sont appelées sharding, et chaque partie est appelée shard[1]. • Comptage parallèle : Effectuer une passe MapReduce pour compter les valeurs de support de tous les éléments qui apparaissent dans la base de données. Chaque mappeur entre un fragment de DB. Cette étape découvre implicitement le vocabulaire des items I, qui est généralement inconnu pour une énorme BD. Le résultat est stocké dans la F-list. • Regroupement d'Items : diviser toute les éléments |I| de F-List en groupes Q. La liste des groupes est appelé liste du groupes (G-list), où chaque groupe reçoit un identifiant de group unique (gid). Comme la liste F et la liste G sont toutes deux petites et que la complexité temporelle est O(|I|), cette étape peut être effectuée sur un seul ordinateur en quelques secondes. • PF parallèle-croissance : L'étape clé de PFP qui comprit deux différentes fonctions importantes :	• L'algorithme FP-Growth distribué fonctionne de manière à diviser pour régner. • PFP partitionne le calcul de manière à ce que chaque machine exécute un groupe indépendant de tâches de minage. • Un tel partitionnement élimine les dépendances de calcul entre les machines, et donc la communication entre elles.

		• Mapper Génération de transactions dépendante du groupe : Chacun l'instance du mappeur est alimentée avec un fragment de base de données généré à l'étape 1. Avant de traiter les transactions dans le fragment un par un, il lit la liste G. Avec l'algorithme de mappage, il génère un ou plusieurs paires clés-valeur, où chaque clé est un identifiant de groupe et son la valeur correspondante est une dépendance de groupe générée transaction. • Réducteur FP-Croissance sur des fragments dépendant du groupe : quand toutes les instances de mappeur ont terminé leur travail, pour chacune group id, l'infrastructure MapReduce automatiquement regroupe toutes les transactions correspondantes dépendant du groupe dans un fragment de transactions dépendante du groupe. Chaque instance de réducteur est affectée au traitement d'un ou plus de fragments dépendants du groupe un par un. Pour chaque shard, l'instance de réducteur construit un arbre FP local et croissance de ses arbres FP conditionnels de manière récursive. Pendant le processus récursif, il peut générer des modèles découverts. • Agrégation : Agrégation des résultats générés à l'étape 4 comme résultat final.	

4.2.2. Algorithmes d'extraction distribuée des itemsets fermés fréquents

Dans ce type d'algorithmes, on dispose d'une base transactionnelle globale D partitionnée horizontalement en plusieurs bases locales D_i (i = 1..n). L'objectif est d'extraire les items sets fermés fréquents globaux IFFG, c.à.d. qui se trouvent dans la base de données globale D, en effectuant le traitement sur les partitions locales D_i. Les définitions et propriétés suivantes sont utilisées dans la majorité des travaux [Mokeddem, 17] :

a. **IFFL (Itemset fermé fréquent local)** : C'est est un itemset fermé fréquent dans la partition *Di*.

b. **IFFG (Itemset fermé fréquent global)** : C'est un itemset fermé fréquent dans la base de données globale *D*.

c. **Support global** : Le support global d'un itemset X dans la base de données globale est noté $\text{Supp(X)} = \sum_{i=1..n} Freq_i(X) / \sum_{i=1..n} |Di|$ où $Freq_i(X)$ est le nombre des transactions dans la partition, qui contiennent l'itemset *X*.

Les approches utilisées dans ces types des algorithmes sont : Le calcul distribué qui s'effectue d'une manière indépendante dans chaque site, ou bien il est synchronisé entre les sites.

- **Calcul distribué indépendant dans chaque site**

Cette approche consiste à fouiller les partitions de données localement d'une manière indépendante en appliquant un des algorithmes séquentiels d'extraction des items sets fermés fréquents, et combiner par la suite les résultats trouvés localement. Cette méthode vise à minimiser le coût de communication à travers des calculs distribués, sans aucune synchronisation entre les différents sites, néanmoins, elle doit instaurer des règles d'estimation des supports global lors de la combinaison des items sets fermés fréquents localement IFFL. Pour illustrer le problème d'estimation du support global, on reprend deux propriétés discutées dans beaucoup de travaux basés sur une telle approche [Alramouni, 06] et [Liu, 07] :

a. **Propriété 1.** Un item fermé fréquent global *X* est fréquent dans au moins une partition.

b. **Propriété 2.** Un item fermé fréquent global X est soit un item fermé fréquent local, ou bien un sous ensemble des IFFLs.

En effet, si on choisit d'extraire les items sets fermés fréquents indépendamment, à partir des différentes partitions de données horizontales disjointes, le problème principal réside essentiellement dans la ″restauration″ de l'information « cachée » dans les IFFL, qui auraient contribué à calculer les IFFG avec leur support exact.

- **Calcul distribué synchronisé entre les sites**

Cette approche consiste à concevoir une version distribuée de l'algorithme séquentiel de base, en synchronisant les calculs entre les différents sites. Le résultat de la fouille distribuée de données avec une telle approche est une simple union des IFFL trouvés localement. Le coût de communication entre les différents sites dépend étroitement de l'algorithme de base utilisé.

Généralement, malgré l'efficacité des algorithmes d'extraction distribuée des items sets fermés relatifs en matière de temps d'exécution et nombre d'item sets produits, ces algorithmes voient leurs performances se dégrader lorsque la taille des données augmente.

5. Conclusion

La découverte des motifs fréquents est l'un des domaines majeurs de la fouille de données. Dans ce chapitre les algorithmes relatifs à l'extraction des items sets fréquents dans un environnement distribué. L'objectif est de découvrir les corrélations qui peuvent exister entre différents attributs des données, selon certaines valeurs seuils, prédéfinies par l'utilisateur. Depuis l'algorithme A priori d'Agrawal et Shafer [Agrawal, 94], plusieurs approches ont été proposées, visant essentiellement l'optimisation des coûts de calcul et de stockage comme par exemple les algorithmes FP-Growth [Han, 00]. Ce dernier est basé sur une stratégie récursive avec l'utilisation d'une structure arborescente compacte, afin de représenter les bases transactionnelles. Malgré leur performance significative par rapport aux approches basées sur la stratégie « tester et générer », les arbres utilisés peuvent ne pas tenir en mémoire centrale.

Chapitre III

Les requêtes Skyline : outils d'optimisation des problèmes de décisions multicritères

---

1. Introduction
2. Définitions
    - 2.1. Définition 1 : principe de domination
    - 2.2. Définition 2 : Skyline
3. Algorithmes fondamentaux des requêtes Skyline
    - 3.1. Algorithmes sans index
    - 3.2. Algorithmes avec index
4. Types de requêtes Skyline
    - 4.1. Contraint Skyline
    - 4.2. Dynamic Skyline
    - 4.3. Spatial Skyline
    - 4.4. Group-by and join Skyline
    - 4.5. Thick Skyline
    - 4.6. ε-skyline
5. Skyline subspatiaux (subspace) et Cubes de Skyline (SkyCube)
    - 5.1. Skyline subspatiaux
    - 5.2. Cubes de Skyline
6. Matérialisation & Requêtes Skyline
7. Conclusion

---

1. Introduction

Notre travail vise à proposer une approche qui permet de manipuler le problème d'hétérogénéité de données multidimensionnelles pour la décision multicritère.

La requête Skyline est un outil puissant pour analyser les données multidimensionnelles et prendre des décisions multicritères. Elle permet d'extraire les meilleurs objets à partir d'un ensemble de données. Elle s'appuie sur le principe de domination de Pareto. Lorsque les dimensions sont hétérogènes, les requêtes Skyline retournent les meilleurs compromis associés à ces dimensions.

Dans ce chapitre, nous citons les définitions de base des requêtes Skyline**.** Par la suite, nous présentons les algorithmes fondamentaux des requêtes Skyline, plus précisément les différentes familles des requêtes Skyline. Enfin, nous détaillons les notions de Skyline subspatiaux et Cubes de Skyline.

2. Définitions

La requête Skyline est un paradigme populaire et puissant pour incorporer les préférences des utilisateurs dans des requêtes relationnelles et extraire des points intéressants à partir d'un ensemble de points [Borzsony et al, 01]. La différence par rapport aux approches classiques est qu'au lieu de trouver des vecteurs ou des points, Skyline trouve les maximums sur un ensemble de lignes ou l'ensemble de lignes de Pareto optimales. Ces lignes sont celles qui ne sont dominées par aucune autre ligne dans la même relation [Kalyvas, 17].

Formellement, étant donné un espace d-dimensionnel $D = \{d_1, \ldots, d_d\}$ et un ensemble D_s de points appartenant à D, un point $p \in D_s$ peut être représenté par $P = \{p.d_1,\ldots,p.d_j\}$, $1<= j<= d$, où $p.d_j$ est la valeur de la $j^{ième}$ dimension du point.

Supposons que le dataset D_s contient les points $D_s = \{p_1, \ldots, p_n\}$. La notation $p_i.d_j \geq 0$, avec $1 \leq j \leq d$ et $1 \leq i \leq n$, est utilisé pour désigner la j-ième valeur dimensionnelle du point p_i. Supposons que pour chaque dimension d_j il existe une relation d'ordre total, notée '<' ou '>' selon le choix de l'utilisateur. Dans nos exemples on utilisera la relation '<'.

2.1. Définition 1 : principe de domination

Étant donné les points p et r $\in D_s$, p domine r, noté $p \prec r$, si et seulement si $\exists\, j \in [1, d]$ tel que $p.d_j < r.d_j$ et $\forall\, i \in [1, d] - \{j\} : p.d_i \leq r.d_i$.

La dominance a la propriété d'une relation transitive. Autrement dit, si p domine r et r domine t, alors p domine également t. Formellement, la propriété de transitivité est présentée dans la proposition suivante.

2.1.1. Proposition 1 : Transitivité

Étant donné les points p, r et t $\in D_s$, si $p \prec r$ et $r \prec t$, alors $p \prec t$.

La transitivité peut être utilisée pour éliminer de toute considération ultérieure un seul point ou un groupe de points qui sont dominés par un point p qui à son tour est dominé par un nouveau point r.

2.1.2. Proposition 2 : Incomparabilité

Étant donné deux points p et r $\in D_s$, si $p \prec\succ r$ et simultanément $r \prec\succ p$, alors p et r sont incomparables sur D_s (i.e. $p \sim r$).

Cette propriété permet de déterminer si un ou plusieurs points sont des points Skyline. Un point de l'ensemble Skyline doit être incomparable à tous les autres points de l'ensemble.

2.2. Définition 2 : Skyline

Un point de données p $\in D_s$ est un point de Skyline ssi $\nexists\, r \in D_s$ tel que $r \prec p$.

On note que pour qu'un point soit un point Skyline, il n'est pas nécessaire de dominer un autre point dans l'ensemble de données. De plus, l'ensemble Skyline d'un jeu de données est unique.

Les notations mathématiques qui seront utilisées dans les points suivants sont résumées dans la table III.1.

Table. III. 1. Notations mathématiques utilisées.

Notation	Definition
D	espace d-dimensionnel
D_S	Ensemble de données d'entrée pour le calcul Skyline (ensemble de points)
d	Nombre de dimensions de D_S
di	one dimension ($1 \leq i \leq d$)
p, r, t	Points de données
s	Point Skyline
p.di	e dimension du point p
q	Requête
SDS	Ensemble de points Skyline de D_S
f	Fonction monotone
$p \prec r$	p domine r
$p \prec q\ r$	p domine r en ce qui concerne q
$p \prec D_s\ r$	p domine r dans l'ensemble de données D_S
$p\ q\ D_s\ q$	p domine r dans l'ensemble de données D_S en ce qui concerne q
$p\ \varepsilon\prec r$	p ε-domine r
$p \prec\succ r$	p ne domine pas r
$p \sim r$	p et r sont incomparables
SD_Sq	Skyline set S du jeu de données D_S par rapport au point de requête q

3. Algorithmes fondamentaux des requêtes skylines

Les algorithmes de calcul de Skyline peuvent être classés en deux catégories : les algorithmes sans index et les algorithmes indexés. Les premiers groupes n'utilisent pas de structure de données particulière et la recherche des Skyline se fait en analysant entièrement la base de données au moins une fois. Les algorithmes indexés ont de meilleures performances, car ils évitent l'accès à l'ensemble de données. Par contre, ils ont une applicabilité limitée en raison de la nécessité d'un ensemble de données indexées.

3.1. Algorithmes sans index

Dans ce point nous présentent les algorithmes suivants: l'algorithme à boucles imbriqués, l'algorithme Divide & Conquer, l'algorithme Sort Filter Skyline et l'algorithme Linear Elimination Sort for Skyline.

3.1.1. Algorithme à boucles imbriqués (Block Nested Loop BNL)

Il s'agit du premier algorithme de recherche de Skyline qui a été proposé dans le contexte de bases de données [Börzsönyi et al., 01]. L'idée principale de cet algorithme est de comparer tous les points de l'ensemble de données deux à deux (i.e. teste de dominance), tout en gardant une liste de points candidats au Skyline en mémoire centrale. En cas de saturation de la mémoire centrale, l'algorithme fait appel à un fichier temporaire stocké en mémoire secondaire dans lequel sont stockés tous les candidats non considérés. Ils seront traités lors d'une prochaine itération. Lors de la comparaison d'un point *p* avec les points stockés en mémoire centrale, trois cas de figure se présentent :

- Le point p est dominé par un point présent en mémoire centrale : p est alors directement éliminé de l'ensemble Skyline (i.e. il est dominé) et ne sera plus pris en compte pour le reste des calculs.

- Le point p domine un ou plusieurs points présents en mémoire centrale : les points dominés par p sont directement éliminés de la liste de points candidats au Skyline. Le point *p* quant à lui est inséré en mémoire centrale.

- Le point p est incomparable avec l'ensemble des points présents en mémoire centrale : dans ce cas, si la mémoire n'est pas pleine, *p* est alors inséré en mémoire centrale. Sinon, il est mis de côté dans le fichier temporaire cité précédemment et sera examiné à nouveau au cours de la prochaine itération de l'algorithme.

La complexité de l'algorithme dans le pire cas est quadratique $O(n^2)$ où n représente le cardinal de l'ensemble de données E.

3.1.2. Algorithme Divide & Conquer (D&C)

L'algorithme Divide & Conquer a été proposé par Börzsönyi [Börzsönyi *et al.*, 01]. Le principe général de cet algorithme consiste à travailler de manière récursive en divisant l'ensemble de données en plusieurs partitions de sorte à ce que chaque partition puisse tenir en mémoire centrale. Le Skyline partiel des points de chaque partition est calculé en utilisant un

algorithme de recherche de Skyline se basant sur un calcul en mémoire centrale (ex. l'algorithme BNL). Le Skyline final est ensuite obtenu en fusionnant tous les Skyline partiels.

La complexité de cet algorithme est de l'ordre de O (n:(log n) l-2 + n: log n) où n représente le cardinal de l'ensemble de données E, et l le nombre de dimensions.

3.1.3. Algorithme Sort Filter Skyline (SFS)

L'algorithme sort-filter-skyline (SFS) améliore les performances de BNL en prétriant l'ensemble de données d'entrée dans un ordre croissant selon une fonction de préférence monotone f, telle que la somme des coordonnées d'un point sur toutes les dimensions, ou optimisé comme entropie (en supposant dans les deux cas que les valeurs ont été normalisées en (0,1) non inclusif). Le prétri impose qu'un point p dominant un autre point q soit visité avant q. Cela garantit le comportement progressif de SFS et la réduction du nombre de comparaisons par paires entre les points. L'algorithme SFS examine les points de données par ordre croissant de leurs scores et conserve un tampon en mémoire contenant les points candidats skyline jusqu'à présent trouvés de la même manière que sur BNL. Au début, le tampon est initialement vide. Un point est lu à partir de l'ensemble de données trié et s'il n'est pas dominé par un point de Skyline dans le tampon y est inséré. Les tests de dominance en SFS sont effectués par une recherche exhaustive sur les points de Skyline existants. [Chomicki, 03]

La complexité de l'algorithme pour le meilleur des cas est O (dn + nlogn) et dans le pire des cas O (dn^2), où d est le nombre de dimensions et n la taille de l'ensemble de données. Cette exécution implique la phase de tri qui ordonne les points en fonction de leur volume [Kalyvas, 17] [Chomicki *et al.*, 03] et [Chomicki *et al.*, 05].

3.1.4. Algorithme Linear Elimination Sort for Skyline (LESS)

L'algorithme LESS, est une version optimisée de SFS, qui permet d'obtenir de meilleures performances moyennes. Comme avec SFS, il trie l'ensemble de données en fonction de la fonction de notation d'entropie, ce qui pousse les points non dominants au début de l'ensemble de données trié. L'algorithme implémente deux optimisations [Godfrey, 05].

La première optimisation de la première passe du processus de tri externe utilise un tampon appelé filtre d'élimination (EF), qui conserve un petit ensemble de points (des copies d'entre eux) qui ont les meilleurs scores d'entropie. Cet ensemble sera utilisé pour élaguer efficacement et le plus tôt possible les tuples dominés du jeu de données. Le jeu de données d'entrée est divisé en b blocs et chaque bloc de points est lu afin d'être trié. Lors du tri, l'algorithme compare les points du bloc avec ceux de l'EF. Si le point du bloc est dominé par un point dans l'EF, il est rejeté. Sinon, si le point est incomparable ou domine d'autres points dans l'EF, il est inséré (une copie de celui-ci) dans l'EF et les points de l'EF qui sont dominés sont rejetés [Godfrey, 05] et [Kalyvas, 17].

La deuxième optimisation combine la dernière passe du processus de tri externe (dernière étape de fusion des blocs b) avec la première passe du processus de filtrage de Skyline (SF) (c'est-à-dire la première passe du composant BNL de SFS), qui élimine les éléments restants. LESS n'est pas applicable dans les scénarios dans lesquels on n'a aucun contrôle direct sur l'algorithme utilisé pour trier les tuples [Godfrey, 05].

La complexité pour le meilleur des cas est O (kn) et dans le pire des cas O(kn^2), où n est le nombre de points et k le nombre de dimensions [Kalyvas, 17].

3.2. Algorithmes avec index

Dans ce point nous présentent les algorithmes suivants: l'algorithme Index, l'algorithme Nearest Neighbor et l'algorithme Sort and Limit Skyline,

3.2.1. Algorithme Index

Les auteurs ont proposé l'algorithme Index, qui partitionne l'ensemble des données de dimension d en listes ordonnées l de sorte qu'un point $p = (x_1^p, x_2^p, \dots x_l^p)$ est affecté à la $i^{ème}$ liste ($1 \leq i \leq l$), si et seulement si, sa coordonnée x_i^p sur la dimension i correspond à la plus petite valeur de p sur toutes les dimensions. Les points de chaque liste sont triés dans l'ordre croissant de leurs coordonnées minimales et indexés sous forme d'un arbre B-Tree. Ensuite, l'algorithme analyse les l listes de manière séquentielle et simultanée à partir des groupes de points de chaque liste. Tel qu'un groupe de points pour chaque liste, représente l'ensemble des points ayant la

même valeur sur la dimension correspondante. Il s'agit de calculer les Skyline locaux de chaque groupe de points en utilisant les B-Tree, puis de fusionner ces derniers avec les autres Skyline pour former le Skyline final [Tan *et al.*, 01].

3.2.2. Algorithme Nearest Neighbor (NN)

L'algorithme Nearest Neighbor (NN) [Kossmann *et al.*, 02] est fondé sur des requêtes de plus proche voisin qui sont implémentées à l'aide d'index de type R/R*-Tree. NN utilise une constatation géométrique très simple : les points les plus intéressants sont ceux qui sont proches de l'origine du repère. Au début, l'algorithme détermine le point ayant la plus petite distance de l'origine du repère. Ce dernier représentant forcément un point skyline p. Ensuite, il segmente le reste des points en l partitions (i.e. zones de l'espace), une partition étant ajoutée pour chaque coordonnée de p. Les prochains plus proches voisins sont ensuite itérativement trouvés dans chaque partition (régions susceptibles de contenir d'autres points candidats) [Ugarte Rojas, 14].

3.2.3. Algorithme Sort and Limit Skyline (SaLSa)

L'algorithme SaLSa (Sort and limit skyline algorithm) [Bartolini, 06] est une amélioration de SFS et LESS. SaLSa permet d'éviter le balayage de l'ensemble de données trié. Comme SFS et LESS, il n'a pas de structure d'index et est le premier algorithme qui exploite les valeurs d'une fonction de notation monotone (limitation) pour trier l'ensemble de données et limiter efficacement le nombre de points à lire et à comparer en utilisant une valeur de seuil.

La suggestion de l'auteur est une fonction de tri optimale, qui ordonne les points en fonction de la valeur $fmin(p) = (min_{i\in[1,d]}p.d_i, sum(p))$, qui est la valeur minimale de coordonnées d'un point parmi toutes les dimensions et $sum(p) = \sum_{i=1}^{d} p.d_i$ i est le deuxième élément de tri qui fonctionne comme une règle de rupture d'égalité.

Soit S l'ensemble actuel des points Skyline, pour chaque point $p_i \in S$ soit $p_i = max\{p_i, d_j\}$, qui est la valeur de coordonnée maximale d'un point. La valeur de seuil utilisée pendant le processus de balayage de filtre pour vérifier si tous les points du reste de l'ensemble de données trié sont dominés, afin que l'algorithme s'arrête, est définie comme $p_{stop} = \arg min_{i\in s}\{p_i\}$. C'est-

à-dire que p_{stop} est égal à la valeur minimale de p_i calculée jusqu'à présent sur la base des points de skyline existants.

L'algorithme pendant le processus d'analyse des filtres lit et examine les points un à la fois. Chaque fois qu'un nouveau point est lu, il est comparé à la liste de Skyline actuelle. S'il est dominé par un point, il est ignoré, sinon il est inséré dans la liste Skyline et l'algorithme vérifie son déclencheur de terminaison. Si le seuil actuel p_{stop}est inférieur ou égal à la valeur f_{min} du point, l'algorithme se termine et renvoie l'ensemble des points de skyline. Cette condition d'arrêt, garantit que tous les points de données examinés ultérieurement ne doivent pas faire partie de la liste skyline, évitant ainsi d'analyser l'ensemble des données [Kalyvas, 17].

4. Types de requête skyline

Dans cette section, nous présentons des différents types de requête skyline. Le type de Skyline diffère selon le cas d'utilisation. Pour cela on trouve Contraint Skyline, Skyline Dynamique, Spatial Skyline, Group-by et Join Requêtes Skyline, Thick Skyline, ε-Skyline.

4.1. Contraint Skyline

Il existe des cas où une requête Skyline peut renvoyer trop d'objets. Cela peut se produire si la dimensionnalité de l'ensemble de données est importante ou si l'ensemble de données est anti-corrélé. De plus, les utilisateurs peuvent être intéressés à étudier un sous-espace particulier plutôt que l'ensemble de l'espace de données. Pour les raisons précédentes, l'utilisateur peut spécifier des contraintes sur certaines dimensions pour exprimer ces restrictions. Les requêtes de contraintes sont très utiles pour la maintenance de Skyline en présence de suppressions ou d'insertions de points [Kalyvas, 17].

La solution de ce type de problèmes est les requêtes de skyline contraintes ou les utilisateurs sont intéressés à trouver les points de skyline d'un sous-ensemble de l'ensemble de données d'origine, qui satisfait une ou plusieurs contraintes. Étant donné un ensemble de contraintes, une requête Skyline contrainte renverra les points les plus "intéressants" de l'ensemble de données défini par ces contraintes. Chaque contrainte est généralement exprimée

sous la forme d'un intervalle le long d'une dimension de l'ensemble de données [Papadias, 03] et [Papadias, 05].

Par exemple, l'utilisateur intéressé par des maisons "intéressantes" dans la plage de distance de la station de métro entre 400 et 1250 et la plage de prix de 100 à 1500. Pour l'exemple maison-métro, la requête Skyline contraint renverra les points H8, H11, H3, H2 (Figure III.1) qui sont inclus dans la région ombrée et sont des points Skyline dans cette région.

SELECT * FROM Houses WHERE ((distance ≥ 400 AND distance ≤ 1250) AND (price ≥ 100 AND price ≤ 1500)) SKYLINE OF price MIN, distance MIN

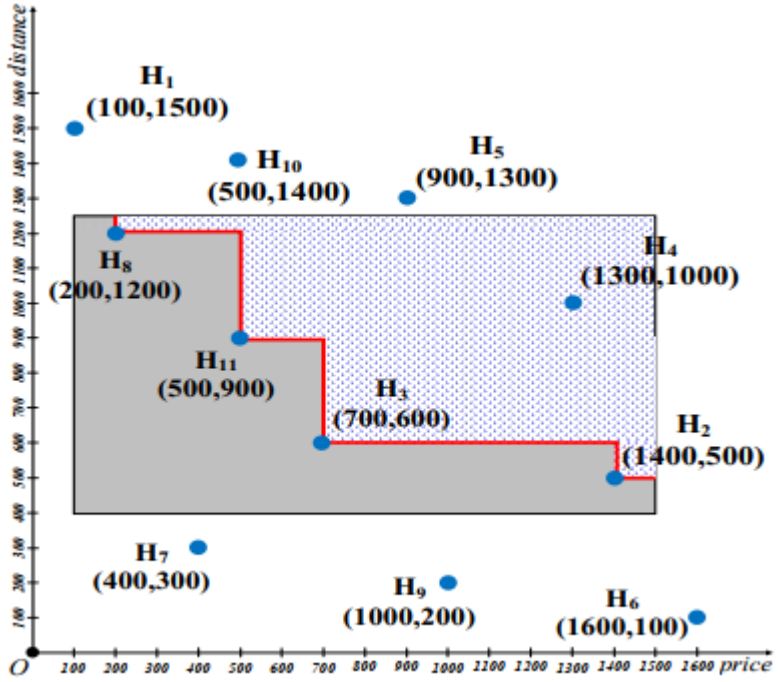


Figure III. 1. Skyline contrainte. [Kalyvas, 17].

- **Définition 1 : Région contrainte**

Étant donné un ensemble de données D_s à d dimensions, une région contrainte C= {c_1, c_2,…c_d} est déterminée par d sous-contraintes c_i où chacune exprime un intervalle le long de chaque dimension de l'ensemble de données. C'est-à-dire c_i= {$c_i$min, $c_i$max} où $c_i$min et $c_i$max sont les valeurs de restriction d'intervalle minimale et maximale sur la i-ème dimension.

- **Définition 2 : Skyline contrainte**

Étant donné un dataset Ds et une région contrainte C ⊆ Ds, un Skyline contrainte contiendra tous les points p ∈ C ($p.d_i \in ci$, ∀ i ∈ [1, d]) où ∀p,r ∈ C, ∃ j ∈[1, d] tel que $p.d_j<r.d_j$ et ∀ i ∈ [1, d]-{j}:$p.d_i \leq r.d_i$.

- **La différence entre Skyline contrainte et Skyline avec contraintes**

En général, une requête Skyline contrainte est calculée sur l'ensemble de données restreint par les contraintes qui ont été placées, tandis que la requête Skyline avec contraintes est calculée sur l'ensemble de données, puis l'ensemble résultant est limité par les contraintes.

4.1. Skyline Dynamique

La requête Skyline dynamique est un type de requête dans laquelle les coordonnées de chaque point sont données par un ensemble de fonctions de distance (dynamiques), qui considèrent la distance entre un point de requête/référence donné q et un point p de l'ensemble de données d'origine [Maamra, 22]. Une requête Skyline dynamique d'un espace de données d-dimensionnel DS spécifie un nouvel espace de données d' dimensionnel DS' basé sur l'espace d'origine et représenté comme un système de coordonnées interne. Pour y parvenir la transformation spécifie m (m ≤ d) fonctions de dimension *f*. Chaque fonction prend comme paramètres une ou plusieurs coordonnées d'origine de chaque point et les mappes dans une nouvelle coordonnée dynamique unique. C'est-à-dire que chaque point p de l'espace de données d'origine de dimension est mappé à un nouveau point de dimension d' p' = (f_1(p), … , $f_{d'}$(p)) où chaque f_i est appelé une fonction distance. Alors le skyline dynamique appliqué sur DS avec le respect des fonctions f_i spécifiées par un point d'interrogation q renvoie le Skyline original du nouvel espace d'-dimensionnel transformé DS' [Papadias, 05].

Pour simplifier la définition de skyline dynamique, on suppose, sans perte de généralité, que DS et DS' ont la même dimensionnalité (d = d'). De plus, pour un point de requête donné q, chaque fonction de distance est définie comme la distance obsolète entre la valeur de la i-ième dimension du point p de l'ensemble de données DS et la valeur de la i-ième dimension du point de requête q, $f_i(p) = |q.\ di-p.d_i|$ [Kalyvas, 17].

- **Définition 1 : Dominance dynamique**

Étant donné un ensemble de données D_s, un point de référence de requête q dans l'espace du travail et deux points p et r $\in D_s$, le point p domine dynamiquement le point r par rapport au point de requête q, noté $p \prec^q r$ si et seulement si $\exists\, j \in [1, d]$ tel que $|q.d_i - p.d_j| < |q.d_i - r.d_j|$ et $\forall\, j \in [1, d] - \{j\} : |q.d_i - p.d_j| \leq |q.d_i - r.d_j|$.

- **Définition 2 : Skyline dynamique**

Étant donné un point de référence de requête q dans l'espace du travail, l'ensemble de Skyline dynamique de D_s par rapport au point de requête q, noté S_{DS}^q , se composent des points de l'ensemble de données qui ne sont pas dynamiquement dominées par aucun autre point. Autrement dit, $S_{DS}^q = \{p \in DS \mid \nexists r \in DS : r \prec^q p\}$.

4.2. Spatial Skyline

La requête de Skyline spatial (SSQ) peut être considérée comme un cas particulier plus restreint des requêtes Skyline dynamiques. Il considère plusieurs points de requête en même temps et s'appuie sur l'existence d'un espace euclidien multidimensionnel pour dériver des structures de délimitation géométriques, telles que la coque convexe et le diagramme de Voronoï pour réduire l'espace de recherche. Étant donné un ensemble de données DS et un ensemble de points de requête Q, une requête de ligne de skyline spatiale récupère les points de DS qui ne sont pas spatialement dominés par aucun autre point de DS par rapport à Q. Plus précisément, un point p ∈ DS domine spatialement un point r ∈ DS par rapport à Q, si et seulement si p est plus proche d'au moins un point de requête q ∈ Q par rapport à r et a dans le meilleur des cas la même distance que r au reste des points de requête, c'est-à-dire qu'aucun autre objet n'est plus près de tous les points de requête donnés simultanément [Sharifzadeh, 06] et [Sharifzadeh, 09].

Notations géométriques

- **Ensemble convexe :** un ensemble S de points, qui existent sur un plan au-dessus de $\mathbb{R}$, est appelé ensemble convexe si et seulement si pour deux points quelconques p,r ∈ S, le segment (ligne) qui les relie réside entièrement dans S (c'est-à-dire tous les points d'un cercle ou d'un hexagone).

- **Coque convexe :** La coque convexe d'un ensemble S de points sur ℝ, est l'intersection de tous les ensembles convexes contenant S.

Un contre-exemple d'une coque convexe impliquerait la ligne pointillée de la figure III.2 [Kalyvas, 17]. Si le segment de la ligne rouge qui appartient entre les maisons H4 et H6 était remplacé par la ligne pointillée qui contient la maison H2, l'ensemble S ne serait pas un ensemble convexe puisque la ligne qui relie les maisons H4 et H6 ne résiderait pas à S.

- **Diagramme de Voronoï :** Étant donné un ensemble S de n points sur ℝ qui existent sur un plan, le diagramme de Voronoï de S, est la subdivision du plan en n cellules, où chaque cellule contient un seul point de S, appelé générateur.

La propriété importante est que tout point (sauf le générateur) dans une cellule particulière sera toujours plus proche du point qui génère cette cellule (Figure III.3) [Kalyvas, 17].

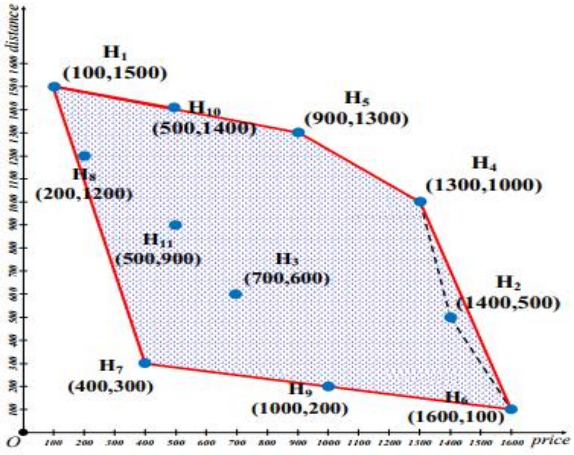


Figure III.3 Coque convexe du jeu de données maison-station de métro.

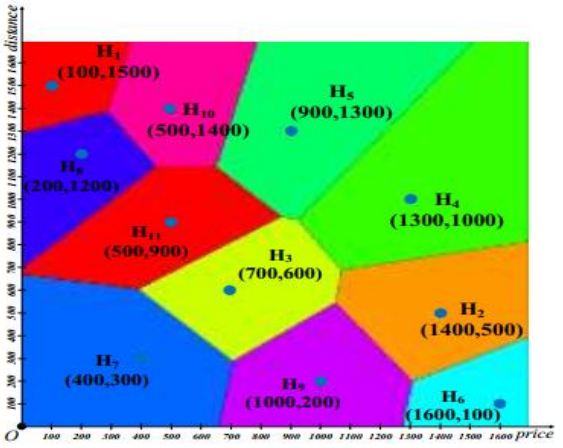


Figure III. 2 Diagramme de Voronoï de l'ensemble de données maison-station de métro.

4.3. Group-by et Join Requêtes Skyline

4.3.1. Group-by Requêtes Skyline

Pour illustrer le requête Group-by Skyline on utilise l'exemple de maison-station de métro, un troisième attribut est inséré dans l'ensemble de données d'origine qui représente le nombre de

chambres dans chaque maison (Table III.2). De cette façon, un acheteur potentiel peut trouver des Skyline individuels en fonction du nombre de chambres. C'est-à-dire regrouper les maisons par le nombre de leurs chambres et ensuite calculer le skyline de chaque groupe. Dans ce cas, la cardinalité des valeurs distinctes des chambres sera égale au nombre d'individuels qui seront trouvés. Dans la figure III.4 sont illustrés les skyline individuels de chaque groupe en fonction du nombre de chambres [Papadias, 05].

Table. III. 2. Dataset de maision station de métro avec nombre de chambres

Maisons	prix (par thousand €)	Distance (m)	No. des chambres
H1	100	1500	1
H2	1400	500	3
H3	700	600	2
H4	1300	1000	3
H5	900	1300	2
H6	1600	100	3
H7	400	300	1
H8	200	1200	1
H9	1000	200	2
H10	500	1400	1
H11	500	900	1

Table. III . 3. Group-by Skyline.

Maisons	prix (par thousand €)	Distance (m)	No. des chambres
H1	100	1500	1
H7	400	300	1
H8	200	1200	1
~~H10~~	~~500~~	~~1400~~	~~1~~
~~H11~~	~~500~~	~~900~~	~~1~~
H3	700	600	2
~~H5~~	~~900~~	~~1300~~	~~2~~
H9	1000	200	2
H2	1400	500	3
H4	1300	1000	3
H6	1600	100	3

Une requête Group-by Skyline basée sur la logique précédente peut être exprimée à l'aide de SKYLINE OF et de l'identifiant GROUP BY comme suit :

SELECT * FROM Maisons SKYLINE OF prix MIN, distance MIN GROUP BY chambres

Pour donner une définition formelle de dominance Groupe by on a :

Étant donné une instance de table relationnelle DS (dataset), dans un espace de dimension d avec des attributs numériques égaux et un schéma A = (A_1, A_2, …, A_d), la notation p[A_i] représente la valeur d'un tuple p dans l'attribut A_i. De plus, étant donné un ensemble G ⊂ A d'attributs de DS qui seront utilisés pour le regroupement et une instance g de G, DS (g) est définie comme l'ensemble de tuples de DS qui appartiennent à l'instance de groupe de g. Soit : $DS(g) = \{p \in D \mid \forall A_i \in G, [A_i] = [A_i]\}$

- **Définition 1 : dominance groupée**

Étant donné un ensemble S ⊂ A (S ∩ G = ∅) qui contient cette fois les attributs Skyline (dont la dominance sera vérifiée), un tuple p domine un autre tuple r par rapport à S, (noté $p \succ_s r$) si et seulement si ∃ A_j ∈ S tel que p[A_j] < r[A_j] et ∀ A_i ∈ S -{A_j} : p[A_i] ≤ r[A_i].

- **Définition 2 : Skyline Group-by**

Finalement, la requête Group-by Skyline contiendra tous les tuples p qui ne sont pas dominés Group-by par un autre tuple r par rapport à S et c'est : $(DS, S) = \{p \in DS \mid \nexists r \in DS, r \succ S\ p\}$

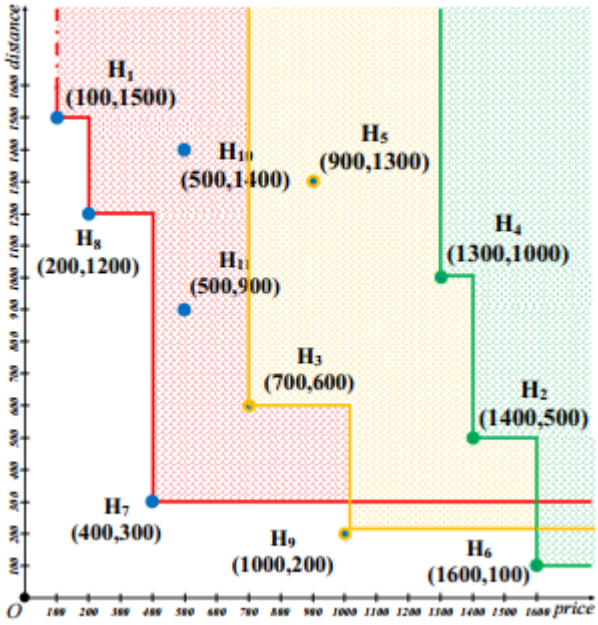


Figure III. 4. Group-by Skyline.

4.3.2. Requêtes Skyline sur les jointures

De nombreuses méthodes ont été développées pour les requêtes Skyline-join dans des environnements distribués et non distribués. Les données sont stockées dans plusieurs tables nécessitant l'existence d'opérations de jointure entre elles pour calculer le Skyline final. Puis, proposer des approches efficaces pour partager le coût de traitement de la jointure avec le coût de calcul le skyline. Plus précisément, en supposant l'existence de deux tables (ou plus) (qui ont un attribut de jointure commun) et en appliquant une opération de jointure sur elles. Il y a un cas où de nouveaux points Skylines peuvent apparaître qui ne sont pas dans Skyline des tables individuelles [Maamra, 22].

Les auteurs ont proposé une approche de jointure tri-fusion dans laquelle ils regroupent les tuples en trois groupes en fonction des valeurs des attributs de jointure et en fonction de la présence ou non de skylines locaux dans leur groupe et de points de skyline dans l'ensemble du tableau. Cela implique une première phase d'élagage de chaque table qui est réalisée avec l'utilisation d'un R-tree. Ensuite, chaque tuple de chaque groupe est trié en fonction de sa valeur d'attribut de jointure. L'étape suivante implique une troisième table (dans ce cas) qui hébergera

l'opération de jointure. Chaque groupe est inséré individuellement et fusionné avec les tuples existants en effectuant en plus une vérification de dominance pour chaque tuple afin de calculer le skyline final de la relation de jointure [Jin, 06].

4.4. Thick Skyline

Une ligne de Skyline épaisse étend la ligne de Skyline conventionnelle (les auteurs utilisent le terme ligne de Skyline mince) en renvoyant les points de Skyline conventionnels et en plus leurs voisins proches non-skyline qui existent à l'intérieur ε-distance, qui sont similaires mais pas aussi bons que les points de Skyline. Cette approche peut aider l'utilisateur dans les cas de recherche du voisin le plus proche où la cardinalité de l'ensemble de données est élevée et les points de l'ensemble de données forment des groupes. Dans leur travail, les auteurs étendent le concept de Skyline à Skyline généralisé en ajoutant une contrainte spécifique à l'utilisateur, définie comme le ε-voisin de tout point de skyline, dans l'espace de recherche de Skyline. Une ligne de Skyline épaisse est composée d'un sous-ensemble des points de Skyline généralisés [Jin, 04].

- **Définition 1 : Skyline généralisé**

Étant donné un ensemble de données d-dimensionnel DS et un ensemble SL = $\{s_1, s_2, s_3, \dots\}$, qui contient les points de Skyline conventionnels de DS, le Skyline généralisé GL est composé des points de Skyline conventionnels et en plus des points non Skyline qui existent dans leur voisinage à une distance ε. Soit $GL = SL \cup \{p \mid p \in DS \wedge p \in si + \varepsilon, \forall i, 1 \leq i \leq d\}$.

- **Définition 2 : Skyline épais**

Étant donné un jeu de données d-dimensionnel DS, un Skyline épais est composé des points de Skyline de Skyline généralisé (appelés points de skyline denses), qui ont dans leur voisinage (défini par ε) un ou plusieurs autres points de Skyline (strictement), et en plus les points de Skyline de Skyline généralisé (nommés points de skyline hybrides) qui ont dans leur voisinage un ou plusieurs autre(s) point(s) de Skyline et quelques points non Skyline. Ainsi, une ligne de Skyline épaisse contient tous les points de skyline de la ligne de Skyline généralisée à l'exception

de ceux qui ne contiennent aucun autre point dans leur voisinage (points d'horizon périphériques) tels que définis par la ligne de Skyline généralisée.

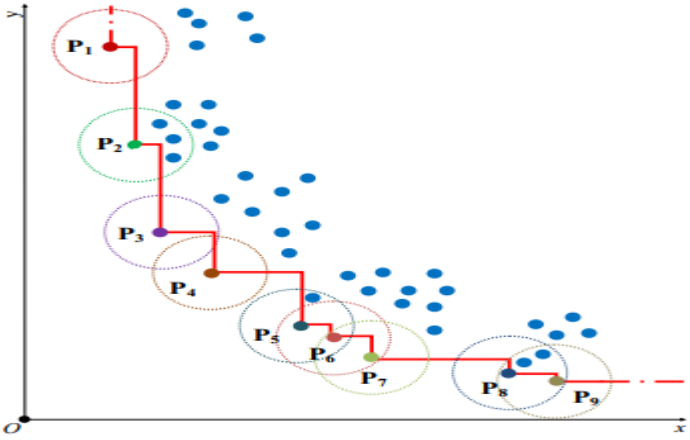


Figure III. 5. Points de Skyline denses, hybrides et périphériques.

La Figure III.5. montre les principales différences entre les points de Skyline denses, hybrides et périphériques dans un plan composé de points aléatoires, car dans l'ensemble de données maison-station de métro, cela ne serait pas évident. Les points p_1, p_3 et p_4 sont des points de Skyline de contour puisque ce sont des points de Skyline mais ils ne contiennent aucun point dans leur voisinage. Les points p_7 et p_6 sont des points de Skyline denses car ce sont des points de Skyline et contiennent un autre point de Skyline à proximité. Les points p_2, p_8, p_9 et p_5 sont des points de Skyline hybrides car ce sont des points de Skyline et contiennent d'autres points non de Skyline à proximité. De plus, les points p_9 et p_8 contiennent un autre point de Skyline dans leur voisinage.

4.5. ε-Skyline

Selon des considérations précédentes, les auteurs ont proposé la ligne de ε- Skyline qui permet aux utilisateurs de contrôler le nombre de points de Skyline de sortie (en les augmentant ou en les diminuant en fonction d'un paramètre ε approprié que l'utilisateur définit), fournit un système de classement intégré et intègre des facteurs de pondération pour chaque dimension [Xia, 2008].

- **Définition1 : ε-domination**

Étant donné un dataset DS de d-dimensions, un vecteur de pondération $W = \{W_i \mid i \in [1, d], 0 < w_i < 1\}$, un paramètre $\varepsilon \in [-1, 1]$ et deux points $p,r \in DS$, p ε domine r, noté $p \prec^{\varepsilon} r$ si et seulement si $\exists\, j \in [1, d]$ tel que $p.d_j < r.d_j$ et $\forall\, i \in [1, d]-\{j\}$: $p.d_i * w_i \leq r.d_i * w_i + \varepsilon$.

- **Définition 2 : ε-Skyline**

Le ε-Skyline d'un dataset DS contient tous les points $p \in DS$ qui ne sont ε-dominés par aucun autre point du dataset.

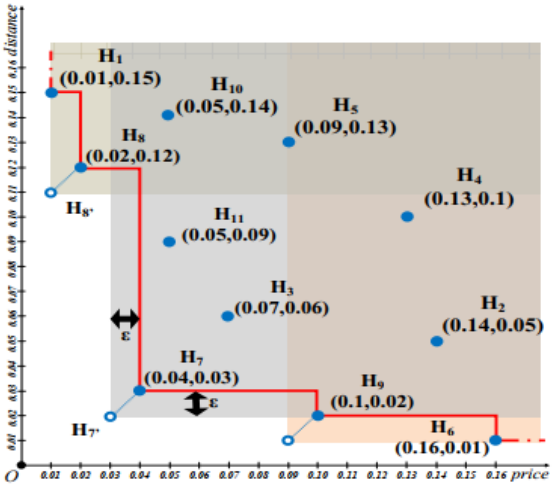


Figure III. 6. Région de dominance de H_7' avec ε=0.01.

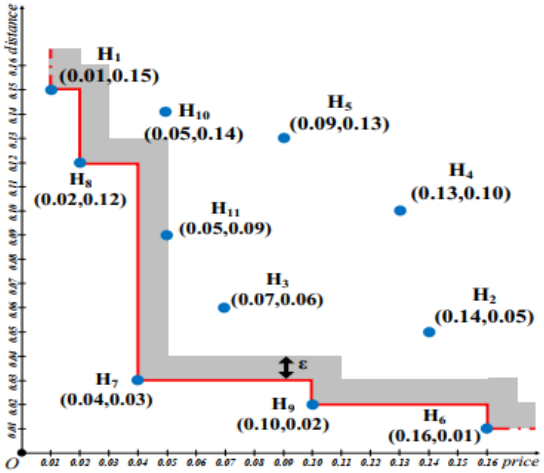


Figure III. 7. ε-skyline avec ε=-0.01.

Un ε-Skyline peut varier de manière monotone d'un ensemble vide à l'ensemble de données en fonction de la valeur de ε. Une ligne de ε-Skyline avec ε = 0 représente le Skyline conventionnelle. Un ε-Skyline avec la valeur ε = -1 renverra l'ensemble de données complet et avec la valeur ε = 1 l'ensemble vide. Plus généralement, le cas d'un ε-Skyline, avec ε > 0 est illustré à la figure III.6. Dans ce cas ε = 0,01 et comme indiqué, la région de dominance de (c'est-à-dire) H_7 sera visualisée lorsque le point H_7 a été déplacé vers l'emplacement de $H_{7'}$ pour remplir la ε-domination. Ce point de cheminement H_7 ε-domine le point H_9 qui pour cette raison ne sera pas dans l'ensemble de Skyline final. Similaire H_8 ε-domine H_4 et H_9 ε-domine H_6. Le Skyline finale définie pour ε = 0,01 sera S = {H_7, H_8}. Le cas d'un ε-Skyline, avec ε < 0 est illustré à la figure III.7. Dans ce cas, le ε-Skyline contiendra les points de Skyline conventionnels et en plus les points qui se trouvent dans la zone ombrée, comme se produit avec le point H_{11}[Kalyvas, 17].

5. Skyline subspatiaux (subspace) et Cubes de Skyline (SkyCube)

Les différents utilisateurs peuvent être intéressés par différentes dimensions / attributs de données et peuvent donc souhaiter récupérer la ligne de Skyline en comparant uniquement un sous-ensemble spécifique de toutes les dimensions / attributs. De plus, spatiale complète dans un espace de grandes dimensions peut renvoyer trop de points intéressants à l'utilisateur qui ne lui permettront pas de prendre une décision appropriée. Ce problème révèle un scénario différent dans lequel une requête est placée sur moins de dimensions que celles de l'espace complet.

5.1. Skyline subspatiaux

Formellement, étant donné un ensemble de points de dimension d, une requête skyline peut être émise sur n'importe quel sous-ensemble des dimensions d. Ce sous-ensemble sera appelé sous-espace et la requête skyline correspondante sur ces dimensions subspace requête Skyline. Les méthodes liées au calcul de Skyline sous-spatial peuvent être classées en deux catégories. Le premier pré-calcule les Skyline de sous-espace pour tous les sous-espaces et organise les résultats dans une structure similaire aux cubes de données dans les environnements d'entrepôt de données, appelée SkyCube, permettant de répondre immédiatement à toute requête de Skyline de sous-espace. La deuxième méthode calcule les Skyline du sous-espace à l'aide de structures d'index [Kalyvas, 2017].

- **Définition 1 : Dominance subspace**

Étant donné deux points p de d-dimension et $r \in D_s$ avec $p = (p_1, p_2, \ldots, p_d)$ et $r= (r_1, r_2, \ldots, r_d)$ et un paramètre k qui spécifie un sous-espace de k- dimension DS_k de Ds avec $k \leq d$, p domine r dans le sous-espace de dimension k si et seulement si $\exists\ j \in [1, k]$ tel que $p.d_j < r.d_j$ et $\forall\ i \in [1, k]-\{j\}$: $p.d_i \leq r.d_i$.

- **Définition 2 : Subspace Skyline**

Étant donné un ensemble de données Ds de d dimension, un point de données $p \in D_s$ et un paramètre k ($k \leq d$) qui spécifie un sous-espace k dimensionnel DS_k de D_s, on dit que le point p est un point d'horizon du sous-espace ssi $\nexists\ r \in D_s$ tel que $r \prec p$ dans le sous-espace DS_k.

5.2. Cubes de skylines

Yuan et al., (2005) ont proposé le SkyCube qui pré-calcule et stocke tous les Skyline de sous-espace possible, fournissant un temps de réponse minimum aux requêtes. Cette structure réunit l'ensemble de tous les Skyline dans tous les sous-espaces non vides possibles (appelés aussi cuboïdes). Il est alors possible de rechercher efficacement des points dominants selon différentes combinaisons de critères. De plus, grâce à cette structure, il devient possible d'observer le comportement des points Skyline à travers l'espace multidimensionnel et ainsi d'analyser et de comprendre les différents facteurs de dominance [Kalyvas, 2017].

Le Skycube, étant inspiré du cube de données [Gray et al., 1997], souffre des mêmes inconvénients de coût de calcul et d'explosion de l'espace de stockage. Ainsi il est naturel d'essayer d'en proposer des représentations concises et des algorithmes efficaces associés. On dresse ci-dessous un bref état de l'art des algorithmes de calcul du Skycube [Ugarte Rojas, 2014].

Dans [Yuan et al., 2005] deux approches ont étés proposées :

- Bottom-Up Skycube (BUS) : L'idée de base de l'algorithme BUS est de calculer chaque cuboïde du Skycube niveau par niveau et de bas en haut, où les cuboïdes fils sont fusionnés pour former (en partie) les cuboïdes pères de niveau supérieur.

- Top-Down Skycube (TDS) : À l'inverse de BUS, l'algorithme TDS permet de calculer chaque cuboïde du SkyCube niveau par niveau et de haut en bas en étendant le principe de l'algorithme D&C.

Pei et al. [Pei et al., 2007] ont proposé Stellar, une méthode de calcul du Skycube qui évite de rechercher l'ensemble des Skyline pour chaque cuboïde. En utilisant les notions de Groupe Skyline et de Sous-espace décisif, Stellar garantit qu'en conservant uniquement les groupes Skyline et les sous-espaces décisifs associés, il est possible de retrouver tous les points Skyline. Cette représentation est donc plus concise que le Skycube.

L'algorithme Orion [Raïssi et al., 2010] propose une représentation concise du Skycube. Il réduit considérablement les tests de dominance dans un sous-espace donné en identifiant les points Skyline qui peuvent être dérivés à partir des Skyline d'autres sous-espaces. En se basant sur la connexion de Galois, l'algorithme réduit également le nombre de sous-espaces à explorer. En effet, un opérateur de fermeture a été défini pour construire une représentation concise du Skycube.

6. Conclusion

Les requêtes Skyline ont pour but de sélectionner les meilleurs éléments dans un espace de recherche en contexte de décision multicritère. Quel que soit le type de skyline à développer, plusieurs méthodologies ont été proposées pour la sélection multi critère.

Généralement une recherche Skyline trouve les maximums sur un ensemble de lignes ou l'ensemble de lignes de Pareto optimales. Ces lignes sont celles qui ne sont dominées par aucune autre ligne dans la même relation. Ce chapitre nous avons présenté les algorithmes fondamentaux de requête Skyline. Dans le cadre de notre travail, nous appuierons sur l'approche de requête Skyline dynamique. Nous allons présenter dans la deuxième partie de ce manuscrit (partie contributions), notre processus de sélection de requête Skyline pour le traitement du problème de recherche et sélection des meilleurs éléments dans un environnement complexe et distribué.

Partie 02
Contributions

Chapitre IV

Traitement d'hétérogénéité de données dans un environnement distribué

1. Introduction
2. Aperçu général de l'approche proposée
3. Approches existantes pour la gestion des réservations de voyage
   - 3.1.Motivations
   - 3.2.Description
   - 3.3.Travaux existants dans ce domaine
4. Architecture proposée pour le système de réservation
   - 4.1.Gestionnaire des tâches
     - 4.1.1. Définition de l'agent
     - 4.1.2. Définition du système multi-agents
     - 4.1.3. Scénario de communication entre agents
   - 4.2.Stations de travail
   - 4.3.Clients
5. Approche proposée pour analyser le système de réservation
   - 5.1. Base des résultats et souhaits de réservation
   - 5.2.Moteur de souhait et exploration de données
     - 5.2.1. Base générique pour les règles d'association exactes
     - 5.2.2. Base informative pour les règles d'association approximatives
   - 5.3.Algorithmes de génération des bases
     - 5.3.1. Algorithme de base générique pour les règles d'association exactes
     - 5.3.2. Réduction transitive de la base informative pour les règles d'association approximatives
   - 5.4.Implémentation
6. Conclusion

1. Introduction

Notre travail permet d'optimiser le calcul et le stockage distribués de structures de données complexes, utiles et dynamiques pour la fouille de données. L'enjeu est de pouvoir manipuler et analyser un très grand volume de données, dans un environnement distribué, en optimisant l'utilisation des ressources réseaux et en maximisant l'interaction avec un utilisateur. Où le contexte d'utilisation de l'application proposée est celui de l'aide à la décision ou l'analyse de données.

Ce chapitre présente notre première contribution concernant le traitement d'hétérogénéité de données dans un environnement distribué. On présente une nouvelle approche du traitement de requête dans un système distribué et complexe dédié à la fouille de données.

Plus précisément, nous présentons un aperçu général de l'approche proposée. Par la suite, nous citons les approches existant pour la gestion des réservations de voyage. Puis, nous présentons l'architecture proposée pour le système de réservation. Enfin, nous détaillons l'approche proposée pour analyser le système de réservation.

2. Aperçu général de l'approche proposée

Notre approche permet d'explorer aisément des bases de données hétérogènes en traitant le problème d'hétérogénéité selon deux niveaux : hétérogénéité niveau requête et hétérogénéité niveau sources de données.

L'approche proposée s'appuie sur le système multi agent et les méthodes de fouille de données dans un environnement distribué [Maamra, 18]. Notre approche est basée sur la construction d'architecture dans le domaine de réservation de voyage, où on a proposé une nouvelle base de données de préférences des utilisateurs, elle permet de faciliter le processus d'aide à la décision.

Avant de présenter en détail l'approche proposée, nous présentons dans un premier temps le domaine d'application. Par la suite, nous décrivons les différentes étapes de construction de l'architecture dédiée à la gestion de réservation de voyage.

3. Approches existantes pour la gestion des réservations de voyage

Afin de valider notre contribution pour le traitement du problème d'hétérogénéité des sources de données distribuées, il est nécessaire de spécifier le domaine d'application sur lequel on crée des sources de données distribuées et hétérogènes. En effet, on a choisi l'un des domaines de la Business intelligent, il s'agit du domaine d'étude des réservations des services de voyage à distance ou tout simplement les réservations de voyage.

Dans ce cadre, on va présenter les motivations de choix de ce domaine. Par la suite, on va donner sa description.

1.1. Motivations

Le domaine des réservations de voyage est un domaine en plein essor, qui consiste à analyser et rechercher les préférences des utilisateurs qui peuvent toucher la décision des entreprises, permet de découvrir les tendances marketing et trouver des compromis multicritères. Le choix du domaine des réservations de voyage est motivé par les trois raisons suivantes :

- ***La distribution d'information*** **:** Les données de domaine de réservation de voyage sont situées dans un environnement distribué, où chaque information est dans une base de données déférente (hôtels, restaurants, aéroport, etc.). Cette distribution des sources peut provoquer des problèmes de complexité lors de l'exploration de ces données.
- ***La diversité des sources de données*** **:** Étant donné que les informations sont distribuées sur différentes bases de données, cela signifie que la source de données est différente c'est-à-dire plusieurs degrés d'hétérogénéité (représentation de données, type de requête, type de serveur, etc.). Cette diversité des sources peut provoquer des problèmes d'hétérogénéités lors de l'exploration de ces données.
- ***La complexité d'information*** **:** La complexité est au cœur des systèmes d'information (SI) d'aujourd'hui. Les grandes organisations ont à gérer des SI de plus en plus vastes et hétérogènes à tout point de vue : des centaines d'applications, des milliers de serveurs, des centaines de milliers d'échanges et de transactions. Les utilisateurs de plus en plus nombreux et répartis sur la planète, qu'ils soient collaborateurs, partenaires, clients ou fournisseurs. La pression des évolutions imposées aux

entreprises, l'intégration toujours plus forte des processus intra et inter-entreprises et la poussée de la dématérialisation dans tous les domaines de l'économie ne font qu'accélérer ce phénomène et confèrent un rôle de plus en plus critique au SI.

1.2. Description

La présence de millions d'entreprises de services touristiques à travers le monde conduit à l'émergence des plateformes de gestion électronique des réservations qui permettent aux clients de connaître en temps réel l'état des stocks des différents fournisseurs de produits touristiques (compagnies aériennes, hôtels, loueurs de voitures, voyagistes, etc.). Ces plateformes de gestion électronique des réservations jouent le rôle d'intermédiaire entre l'entreprise et le client (les clients dans ces systèmes sont les agences) qui permettent de faciliter et accélérer le processus de réservation à distance. Mais, il augmente le coût de réservation.

Les agences touristiques permettent de garantir le confort et la sécurité des voyageurs. Cependant, la réservation manuelle est très pénible et consomme beaucoup de temps.

Par exemple, les services d'hébergement, on aimerait choisir le meilleur hôtel dans des emplacements de choix, avec des installations modernes, un environnement propre et des tarifs raisonnables.

Les systèmes informatiques de réservation (CRS) vous permettent de réserver une ressource ou un service à temps pour une personne ou un groupe. Ces systèmes sont intégrés dans l'immobilier (salle, salle de conférence, court de tennis, salle de conférence), le transport (voiture, train, bateau, avion), les outils (scie, grue) sous forme de fourniture (location, prêt) à partir d'une plage horaire déterminée avec, selon le cas, l'option d'une ou plusieurs personnes.

Ce système de réservation est souvent associé à un système de paiement qui s'adapte à de nombreux domaines : location (court de tennis, villa, voiture, chambre), achat et vente d'objets usagés (livre, maison, voiture, jeux), les rendez-vous (consultation médicale, concert, exposition, réunion) ou les transports (siège, cabine). Les domaines sont souvent en relation avec l'utilisation d'objets qui demandent un entretien fréquent ou coûteux ou soumis à une forte réglementation afin d'assurer la sécurité des utilisateurs ou des usagers et la disponibilité de l'objet. La durée d'utilisation de l'objet de la réservation est généralement courte tandis que le nombre d'objets mis à disposition est nombreux. [Wikipedia, 22]

1.3. Travaux existants dans ce domaine

Généralement, les travaux existants dans ce domaine sont des systèmes simples. Ils sont constitués d'une seule base de données pour traiter les réservations simples. Parmi ces travaux, nous citons :

En 2010, l'auteur [McTavish, 10] a proposé un système intelligent de recherche et de réservation d'hôtels basé sur un agent qui se chargera de rechercher et réserver les hôtels pour divers utilisateurs. Le système offrait aux utilisateurs la possibilité de saisir certains critères de recherche, puis d'effectuer la recherche sur la base de ces critères et de réserver la base d'hôtel la plus appropriée sur la confirmation de l'utilisateur. Celles-ci sont réalisées à l'aide de JADELEAP sur l'appareil mobile de l'utilisateur.

En 2014, Oloyede [Oloyede, 14] propose un système de réservation de bus au Nigeria. Il était développé par l'Extensible Hypertext Markup Language (XHTML), PHP Hypertext Preprocessor (PHP), Ajax, Cascading Style Sheet (CSS) et JavaScript. Le système dispose d'une base de données relationnelle modélisée par Structure Query Langage (SQL).

En 2014, Ogirima [Ogirima, 14] propose un système de gestion hôtelière informatisé en ligne (HMS). Le système proposé dispose une base de données relationnelle modélisée par Structure Query Language (SQL). L'auteur décrit son système et présente une comparaison entre HMS en ligne et le système de gestion hôtelière manuelle. Le résultat indique que les utilisateurs préfèrent la gestion hôtelière en ligne qui permet de simplifier l'utilisation et l'accessibilité, et offrir une sécurité dans la zone d'étude.

En 2014, Bemile [Bemile, 14]. a proposé un système de réservation d'hôtel en ligne qui permet aux clients de faire des demandes en ligne et de réserver des services en fournissant les détails requis. Le système proposé permet aux clients potentiels d'obtenir les informations correctes, telles que les tarifs des chambres et l'emplacement de l'hôtel. Il adopte la visite virtuelle de l'hôtel Shangri-La à Singapour. Les outils de développement de ce système sont Apache, MySQL et PHP

Bhagat [Bhagat, 18] a développé un système de réservation de parking perceptif avec la technologie IOT. Le système de Bhagat se compose de différents modules comme le serveur, la base de données, l'application utilisateur et l'arrangement des emplacements de stationnement. Le

mécanisme de la fente de stationnement est un capteur Arduino UNO et ultrasons. Arduino est utilisé pour gérer le capteur à ultrasons et la porte d'entrée. Le capteur à ultrasons est utile pour détecter la position de la voiture. Lorsque l'utilisateur scannera le code QR sur la place du stationnement, il sera facturé et paie pour une durée qui est déjà mentionnée par lui-même au moment de la réservation de la place via l'application Android. Un utilisateur doit d'abord télécharger l'application Android sur son téléphone mobile Android. L'utilisateur doit faire un enregistrement avec un identifiant spécifique (en utilisant le numéro de carte AADHAR ou le numéro de licence) en utilisant l'application. Les informations sur les places de stationnement sont stockées dans la base de données qui est toujours mise à jour. Les serveurs gèrent et mettent à jour ces informations et envoient les notifications aux utilisateurs après chaque réservation des places de stationnement. Pour gérer les emplacements de stationnement et le système de base de données des utilisateurs, les auteurs fournissent un site Web pour l'administration du système.

Pour montrer l'originalité de notre travail par rapport aux travaux similaires, on a réalisé une étude comparative qui va porter sur trois critères de comparaison : méthode de recherche et de réservation, modèle multidimensionnel et application du domaine.

Dans cette étude comparative, on a également sélectionné un ensemble des travaux similaires de domaine de décision multicritère à base de requête Skyline dans la période 2014-2018. Les résultats de comparaisons qualitatives sont présentés dans le tableau suivant.

Table. IV. 1. Comparaison entre différents systèmes.

Critères / Travaux	Système de Réservation En Ligne		
	Méthode de recherche et de réservation	**Modèle multidimensionnel**	**Application du domaine**
McTavish (2010)	Intelligent agent	Non	hotel search and booking
Oloyede (2014)	Request SQL	Non	Bus reservation
Ogirima (2014)	Request SQL	Non	Online computerized Hotel Management
Bemile (2014)	Request SQL	Non	Hotel Reservation
Bhagat (2018)	Arduino UNO and Ultrasonic sensor	Non	perceptive car parking booking

Dans ce contexte, on présentera dans cette thèse notre proposition de système de réservation des sources de données distribuées et hétérogènes. Il est donc nécessaire de définir l'architecture générale qui permet de satisfaire les besoins de l'utilisateur, c'est-à-dire le client est autorisé à réserver un voyage de n'importe où et à tout moment selon les conditions requises et dans un court laps de temps.

Figure IV.2. Architecture proposée pour le système de réservation

Notre architecture comprend une base de données de résultats agrégés des demandes des clients, c'est-à-dire que lorsque l'utilisateur dans le cas de la demande une réservation, la réponse est immédiate. La présence de ces résultats collectés permet d'accélérer la réponse. Cette base de données gérée par un gestionnaire de tâches qui se connecte d'un côté aux postes de travail qui gèrent des entrepôts de données distribuées et hétérogènes pour les services touristiques (hôtels, les aéroports, les restaurants, ainsi que les guides touristiques, etc.) et d'autre coté avec les clients. La figure IV.1 montre l'architecture proposée de notre système.

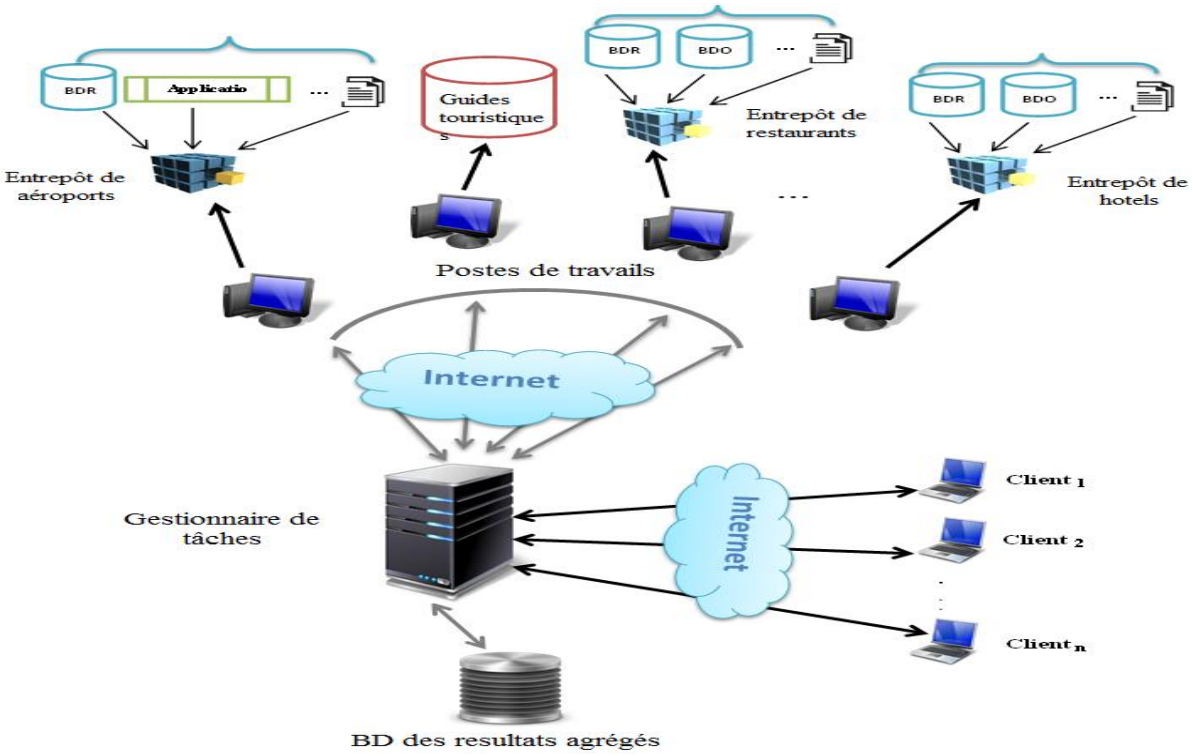


Figure IV.1 Architecture pour un système de réservation.

2.1. Gestionnaire des tâches

Le gestionnaire des tâches de notre proposition est un système multi-agents intelligent (au lieu de l'agent humain) pour effectuer des activités de recherche et de réservation similaires qui peuvent améliorer la vitesse de la recherche et réduire considérablement les coûts de recherche. Mais la question ici est pourquoi un système multi-agents, et quel est l'agent ?

2.1.1. Définition de l'agent

Un agent est une entité physique ou virtuelle, qui possède les caractéristiques suivantes [Ferber, 1997] :

- Qui est capable d'agir dans un environnement,
- Qui peut communiquer directement avec d'autres agents,
- Qui est animé par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'il cherche à optimiser),
- Qui a des ressources propres,
- Qui est capable de percevoir (mais de manière limitée) son environnement,
- Qui n'a qu'une représentation partielle de cet environnement (et éventuellement aucune),
- Qui a des compétences et offre des services,
- Qui peut éventuellement se reproduire,
- Dont le comportement tend à répondre à ses objectifs, compte tenu des ressources et de l'expertise dont il dispose, et en termes de perception, de représentations et de communications qu'il reçoit,
- Les agents jouissent d'autonomie : Cela signifie qu'ils ne sont pas motivés par des commandes de l'utilisateur (ou d'un autre agent), mais par un ensemble de tendances qui peuvent prendre la forme d'objectifs individuelle à atteindre.

Il a la possibilité de répondre oui ou non aux demandes des autres agents. Il a donc une certaine liberté de manœuvre, ce qui le différencie de tous les concepts similaires, qu'ils appellent "objets", "modules logiciels" ou "processus". Mais l'autonomie n'est pas seulement comportementale, c'est aussi une question de ressources. Pour agir, l'agent a besoin d'un certain nombre de ressources : énergie, CPU, quantité de mémoire, accès à certaines sources

d'information, etc. Ces ressources sont à la fois ce qui rend l'agent non seulement dépendant de son environnement, mais aussi, en étant capable de les gérer [Ferber, 97].

2.1.2. Définition du système multi-agents

Un système multi-agents (SMA) est un système composé des éléments suivants [Ferber, 97] :

- Un environnement E, c'est-à-dire un espace ayant généralement une métrique.
- Un ensemble d'objets O. Ces objets sont localisés, c'est-à-dire que, pour tout objet, il est possible, à un instant donné, d'associer une position en E.
- Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par des agents.
- Un ensemble d'agents, qui sont des objets particuliers (A, O), qui représentent les entités actives du système.
- Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.
- Un ensemble d'opérations Op qui permettent aux agents As de percevoir, de produire, de consommer, de transformer et de manipuler des objets O.
- Opérateurs pour représenter l'application de ces opérations et la réaction du monde à cette tentative de changement, que l'on appellera les lois de l'univers.

Notre gestionnaire de tâches est un système multi-agents composé des agents suivants : Agent Client, Agent de Stock, Agent de Station, Agent d'Agrégation et Agent de Réponse.

2.1.3. Scénario de communication entre agents

Dans ce point, on présente les principaux comportements des différents agents. Dans ce cas, on présente les principaux comportements des différents agents. On s'est basé sur l'architecture modulaire pour modéliser l'architecture interne d'agent. Ce type favorise l'amélioration, l'ajout et la réutilisation des principaux modules de chaque agent [Merzoug, 18].

La nature de chaque agent de notre système multi agent se distingue par des modules internes, chaque module est dédié à une fonction spécifique requise pour l'accomplissement du but de l'agent. Notre système SMA est basé sur cinq agents (Agent Client (AC), Agent Stock

(AS), Agent Station (ASt), Agent Agrégateur (AA), Agent Réponse (AR)), La figure IV.2 montre le scénario général de notre système multi agent (SMA).

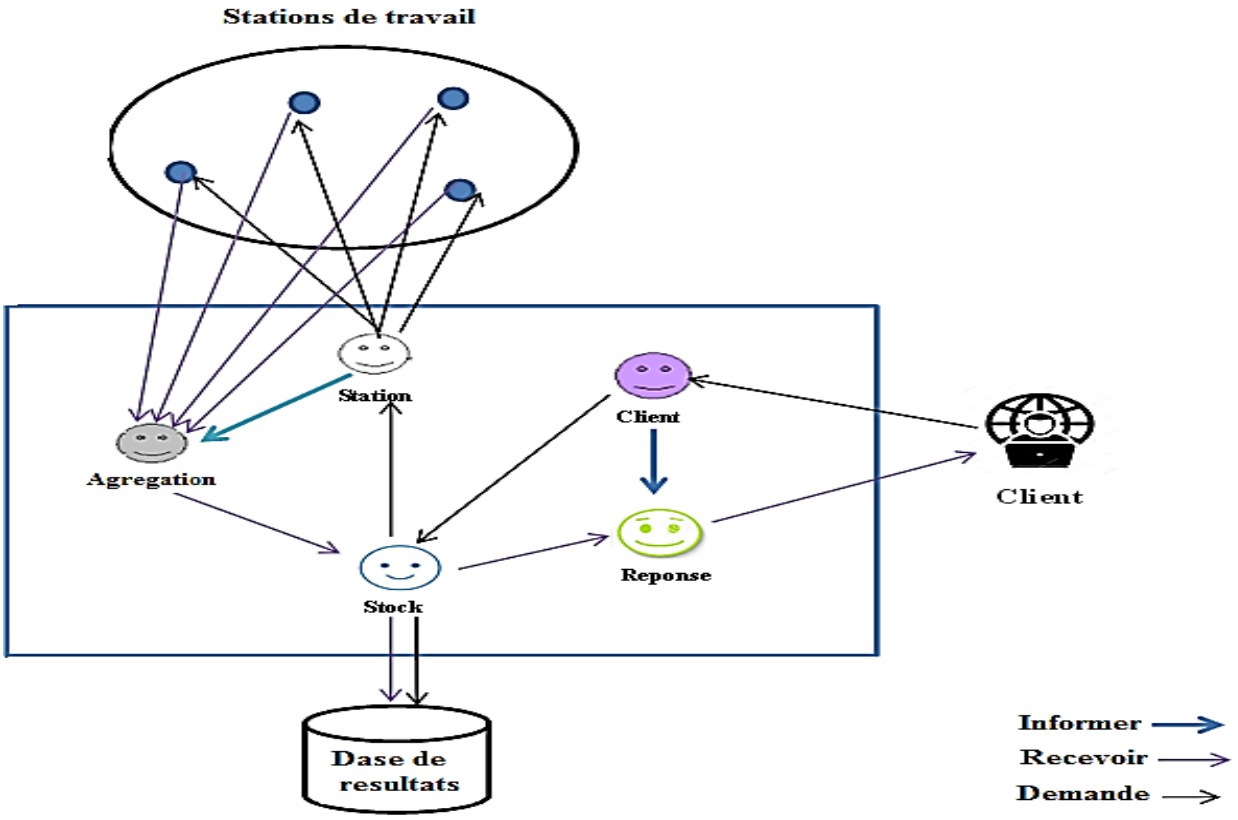


Figure IV. 2. Scénario général de notre système multi agent.

Agent Client

Le comportement d'agent client est programmé de façon capable d'acquérir les requêtes des utilisateurs. L'agent fournit aux utilisateurs une interface graphique pour interagir avec le système. Il est responsable de l'acquisition des demandes des utilisateurs et identifie les conditions de choix. II envoie ce message au l'agent de stock. Ensuite, il informe l'agent de réponse pour être prêt à recevoir la réponse de la requête de l'utilisateur. Cet agent utilise une interface d'entrée / sortie pour reçoit des demandes des clients tandis que l'interface d'entrée/ sortie externe sert à communiquer avec un agent de stock et l'agent de réponse. La figure suivante présente l'activité de cet agent.

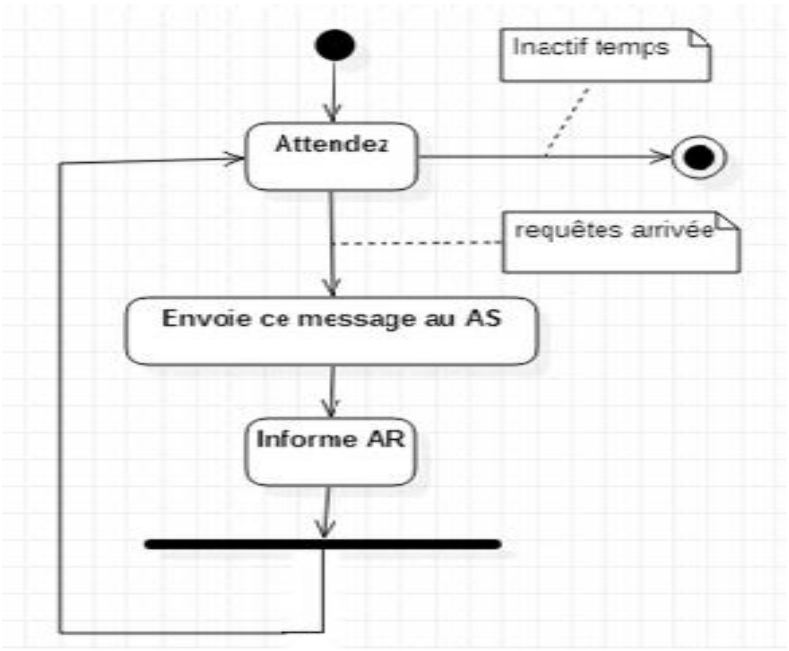


Figure IV. 3.Diagramme d'activité d'agent AC.

Les principaux modules de l'agent Client :

- *Un module de communication* : Ce module offre une interface de communication humain/agent pour aider l'utilisateur à saisir ses besoins. Il offre une méthode pour la gestion des communications avec les autres agents.
- *Un module de traitement* : Dans ce module l'agent transmet la demande à l'agent de Stock et informe l'agent de réponse qu'il a une demande pour être prêt à recevoir la réponse de la requête de l'utilisateur.

Agent Stock

L'agent Stock fournit une interface de transaction entre l'agent Client et l'agent Stations (AS). L'agent Stock recherche la réponse de requête client dans la base des réponses est préparée à l'avance, si la réponse existe dans le stock, il envoie à l'agent de Réponse. Sinon, il envoie la requête à l'agent de Stations. La figure IV.4 présente l'activité de cet agent.

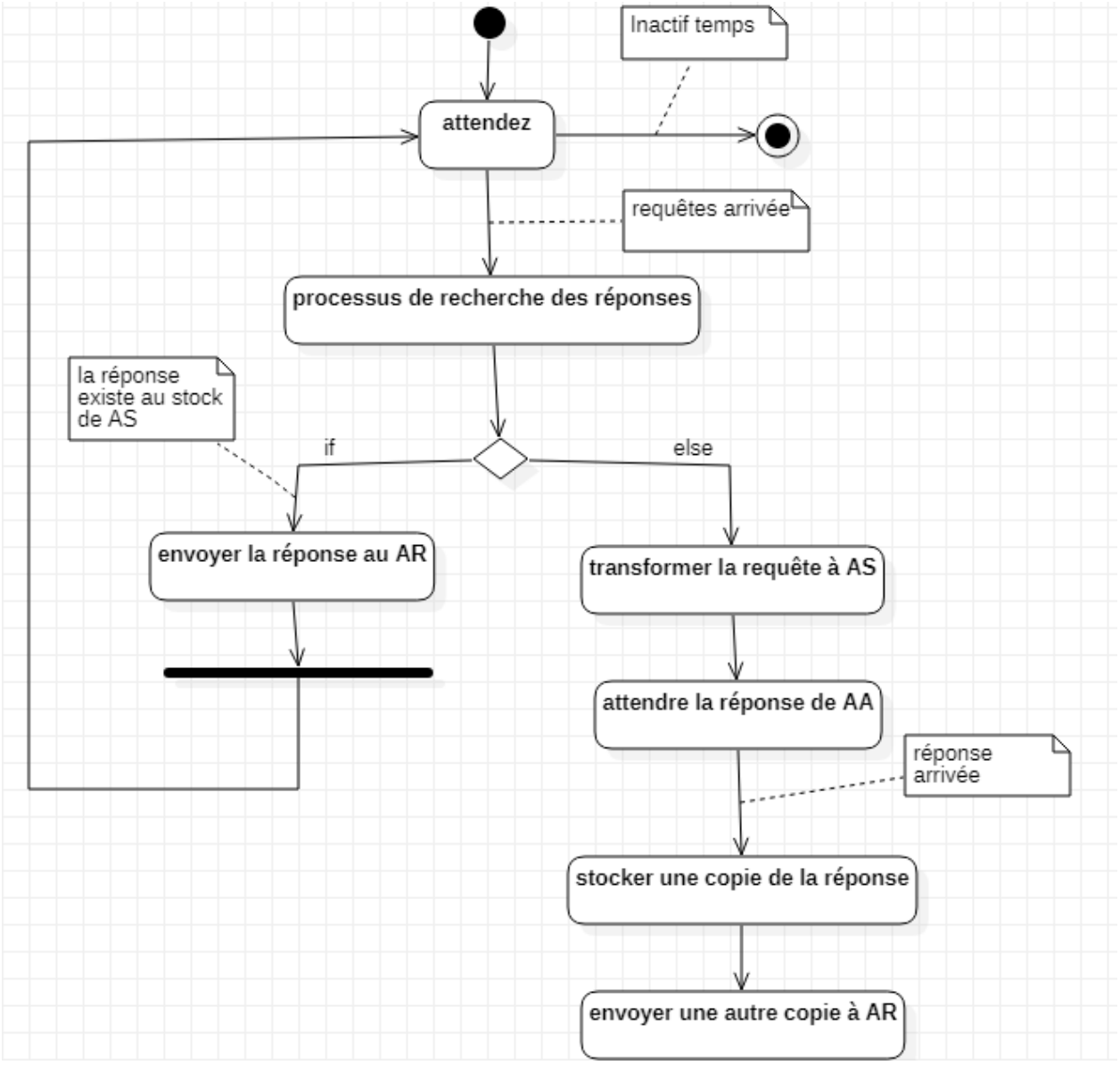


Figure IV. 4. Diagramme d'activité d'agent AS.

Les principaux modules de l'agent stock :

- *Un module de communication* : Ce module est une interface de communication qui supporte l'interaction agent/agent, Il offre en particulier une interface qui assure la communication avec l'agent Client, l'agent Réponse, l'agent Stations et l'agent d'Agrégation. Cette interface permet d'acquérir la requête et de préparer la réponse. Par la suite, il envoie la requête au module de traitement.
- *Un module de traitement* : Il est basé sur le résultat de l'agent de Stock dans la base des réponses. si la réponse existe il l'envoie à l'AR. Sinon, il transforme la requête à l'agent de

Stations et il reste dans l'état d'attend la réponse de l'AA. Après avoir reçu la réponse finale, il stocke une copie et envoie une autre à l'agent de Réponse.

Agent Stations

L'agent de Stations est programmé de telle façon à diffuser la requête d'utilisateur aux stations de travail pour préparer la réponse de cette requête. D'autre part, il informe l'agent d'Agrégation pour attendre la réponse. L'agent de Stations offre une interface d'entrée / sortie interne pour communiquer avec l'agent AS et une autre interface d'entrée / sortie externe pour communiquer avec l'AA. La figure IV.5 montre les détails du comportement de l'agent AS.

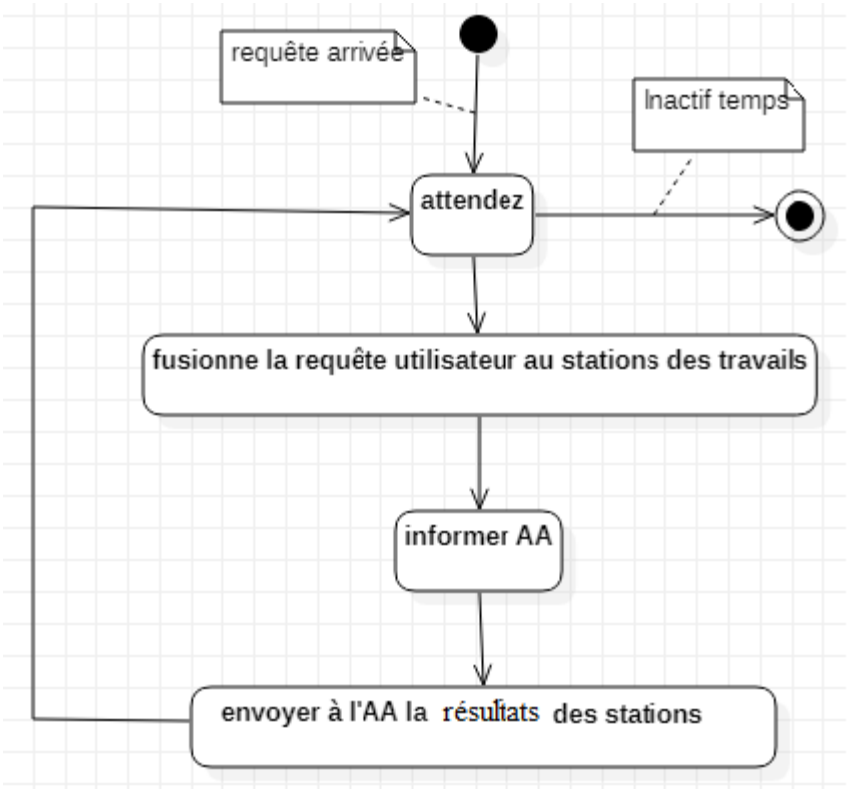


Figure IV.5. Diagramme d'activité d'agent des Stations.

Les principaux modules de l'agent Stations :

- *Un module de communication* : Ce module est une interface de communication qui supporte l'interaction agent/agent. Il offre en particulier une interface qui assure la communication entre l'agent des Stations et entre l'agent de Stock l'agent des Stations et l'agent d'Agrégation. Cette interface permet d'acquérir la requête et de préparer la réponse. Par la suite, il envoie la requête au module de traitement.

- *Le module de traitement :* Dans ce module, l'agent des stations diffuse la requête dans les stations des travaux pour trouver les réponses. Il envoie un message à l'agent d'Agrégation pour attend les résultats des stations. L'agent des Stations envoie les résultats à l'AA lorsqu'ils sont prêts.

Agent Agrégation

Le comportement d'agent d'Agrégation est programmé de façon qu'il soit capable d'acquérir les résultats des stations des travaux par fournir une interface d'interaction avec l'agent des Stations. Il agrège ces résultats et prépare la réponse finale. À travers une interface d'interaction, l'agent d'Agrégation envoie la réponse finale à l'agent de Stock. La Figure IV.6 présente l'activité de l'agent d'Agrégation.

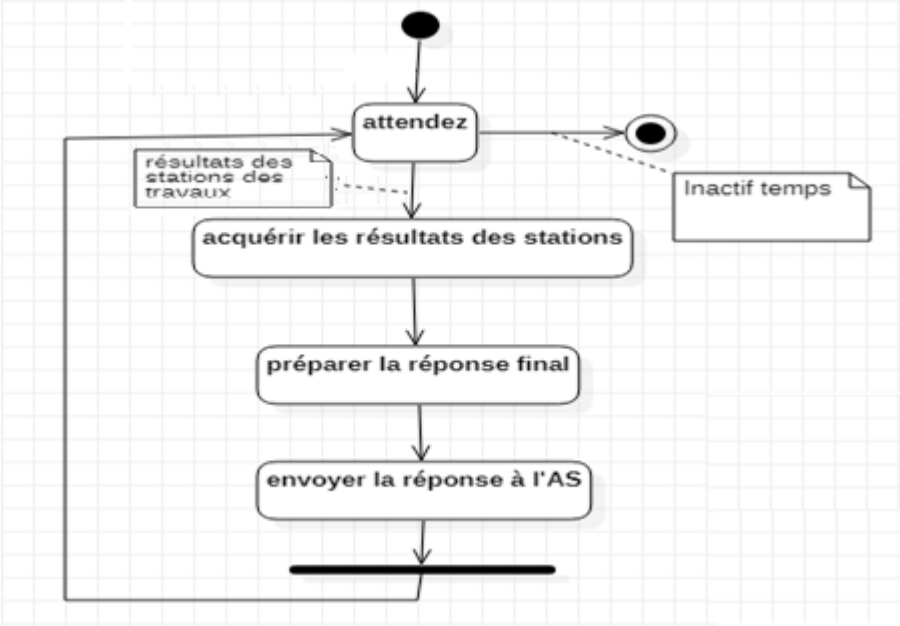


Figure IV.6. Diagramme d'activité d'agent d'Agrégation.

Les principaux modules de l'agent d'Agrégation :

- *Un module de communication :* Ce module est une interface de communication qui supporte l'interaction agent/agent. Il offre en particulier une interface qui assure la communication entre l'agent des Stations et avec l'agent de Stock. Cette interface permet d'acquérir les résultats de la requête client et de préparer la réponse finale.

- *Le module de traitement :* L'agent d'Agrégation dans ce module reçoit les déférentes réponses de requête de client de la part des stations des travaux. Il prépare la réponse finale par un module de préparation. Puis, il l'envoie à l'agent de Stock.

Agent Réponse

Le comportement d'agent Réponse est programmé de manière à lui permettre d'obtenir les réponses des requêtes utilisateurs. L'agent fournit aux utilisateurs une interface graphique pour interagir avec le système. Il est responsable de présenté des réponses des requêtes utilisateurs. Cet agent utilise une interface d'entrée / sortie sert à la communication avec un agent de Stock pour reçoit les réponses des requêtes utilisateurs. La figure IV.7 présente l'activité de cet agent.

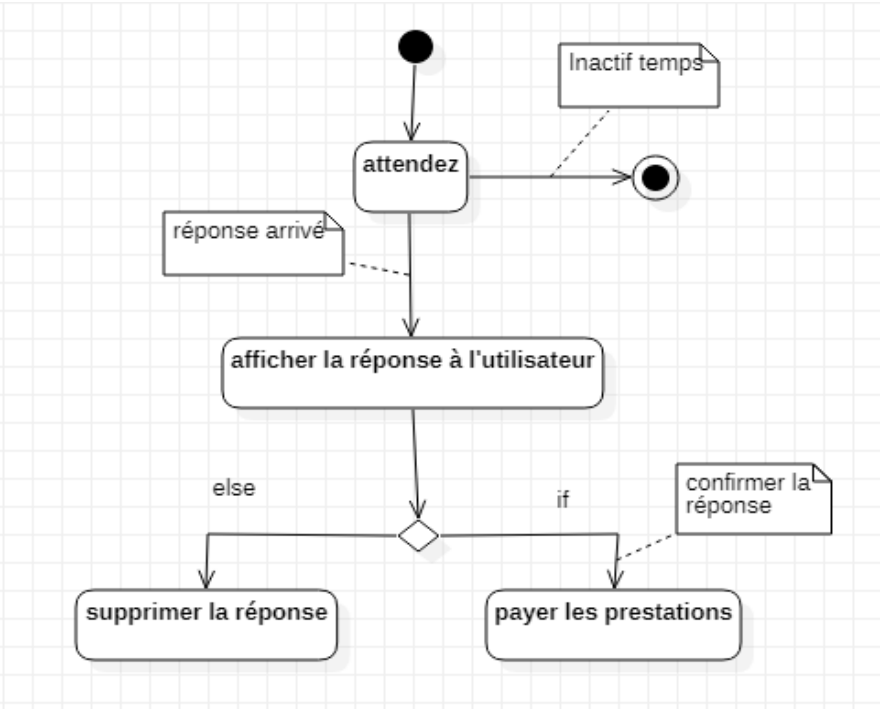


Figure IV.7. Diagramme d'activité d'agent de Réponse.

Les principaux modules de l'agent Réponse :

- *Un module de communication :* Ce module offre une interface de communication humain/agent pour l'affichage des résultats aux utilisateurs. Il fournit également une interface d'interaction agent/agent qui assure la communication avec l'agent de Stock. Cette interface permet d'acquérir les réponses des requêtes clients.

- *Le module de traitement :* L'agent de Réponse dans ce module reçoit les réponses des requêtes, puis les affichent aux utilisateurs. Si l'utilisateur confirme la réponse, l'agent de Réponse demande les frais de confirmation. Sinon, il annule la réponse de la requête de cet utilisateur.

Le scénario général de fonctionnement du système proposé est décrit dans les étapes suivantes :

1. Le client accède à l'interface web pour demander la réservation de voyage.
2. L'agent Client reçoit le message de client, et envoie ce message au stocke et informe l'agent de Réponse.
3. L'agent Stock reçoit le message de l'agent client, puis recherche le BR, si le résultat existe en envoyant la réponse à l'agent de Réponse, sinon il envoie le message à l'agent Station.
4. L'agent Station fusionne le message dans les postes des travaux et informe l'agent d'Agrégation.
5. L'agent Agrégation reçoit les résultats et prépare la réponse finale, puis envoie cette réponse à l'agent de Stock.
6. L'agent Stock envoie une copie de résultat à l'agent Client et stocke une autre dans la base de données des résultats.
7. L'agent Réponse reçoit le message de l'agent de stockage et affiche la réponse au client, puis demande la confirmation de cette réponse.

4.2. Stations de travail

Les stations de travail de notre proposition sont des entrepôts de données hétérogènes et distribuées. Pour extraire les connaissances demandées par le client, chaque station contient un serveur gérant l'entrepôt de données. Tandis que la recherche et la sélection des réponses basé sur l'utilisation de requête Skyline qui permet la sélection des meilleurs éléments selon le besoin du client (voir son application dans le chapitre suivant). Dans notre proposition, on a quatre postes de travail. Chaque poste de travail contient les informations des services des voyages différentes à partir des bases de données distribuées et hétérogènes. Ces informations sont collectées dans son propre entrepôt de données.

4.3. Clients

Dans notre système, les clients sont divisés en deux types Personnes et Agences de voyages. Les agences de voyages ont des privilèges, par exemple, ils peuvent réserver pour plusieurs personnes dans diverses conditions. Dans les deux cas, après l'inscription, le client reçoit la lettre de confirmation d'inscription, qui comprend le coût du voyage selon sa demande, confirmant son inscription pour payer les frais dans les vingt-quatre heures. S'il ne paie pas les frais, sa demande sera annulée.

5. Approche proposée pour analyser le système de réservation

Les règles associatives sont des règles extraites d'une base de données transactionnelles et qui décrivent des associations entre certains éléments. Elles sont utilisées dans notre proposition de réservation des services de voyage où la principale application pour analyser le choix des utilisateurs (Wishes Analysis) dont le principe est l'extraction d'associations entre services sur les choix utilisateurs. Le but de la méthode proposée est d'étudier ce que les clients ont tendance à être dans les réservations de voyage pour des informations sur la nature des clients et comment ils se rapportent au choix de certains services?

L'extraction de règles d'association 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 via le data mining [Fayyad, 96]. La plupart des approches proposées pour l'extraction des itemsets fréquents reposent sur les quatre phases suivantes : préparation des données, extraction des ensembles fréquents d'attributs, génération des règles d'association et interprétation des résultats. Alors que, le cadre proposé aide à faciliter l'extraction de règles d'association.

Dans notre contribution, on a utilisé la sémantique pour le problème de l'extraction de règles d'association basée sur la fermeture de la connexion de Galois. On distingue deux types de bases pour les règles d'association, base générique pour les règles d'association exactes et base informative pour les règles d'association approximatives. Elles sont définies à partir des itemsets fermés fréquents et leurs générateurs. Ce sont des ensembles de taille réduite qui minimisent le

nombre de règles d'association générées tout en maximisant la quantité et la qualité des informations convoyées.

L'union de ces deux bases constitue donc un ensemble générateur minimal non redondant pour toutes les règles d'association valides, leurs supports et leurs confiances. Dans notre proposition le moteur de souhait et l'exploration de données sont présents le générateur de règles d'association valides.

2.1. Base des résultats et souhaits de réservation

Le but est de proposer une base de résultats dans notre système et analyser les préférences des utilisateurs pour aider les entreprises à fournir des meilleures offres pour les clients, et assister les clients à obtenir leurs souhaits. Cette base est formée la sauvegarde d'une copie de sélection d'utilisateurs.

Les données de réservation se composent principalement de données non structurées de la présentation des souhaits des entreprises et du souhait du client. En fonction de l'objectif spécifique de l'analyse, le filtrage des données est effectué sur les données brutes qui sont ensuite analysées pour la compréhension et les prédictions sur la structure et la dynamique des souhaits de la communauté.

L'analyse des souhaits et la gestion de la réputation sont des applications où le traitement du langage naturel est appliqué aux sites de réservation.

Dans notre étude, on a proposé un moteur de souhait et d'exploration de données, qui est basée sur les règles d'association pour découvrir les similitudes entre les produits dans des données saisies sur une grande échelle. De plus, on a appliqué des techniques d'analyse graphique pour identifier les offres de voyages et les services requis au sein des communautés. La figure IV.8 présente un cadre proposé pour l'analyse du système de réservation

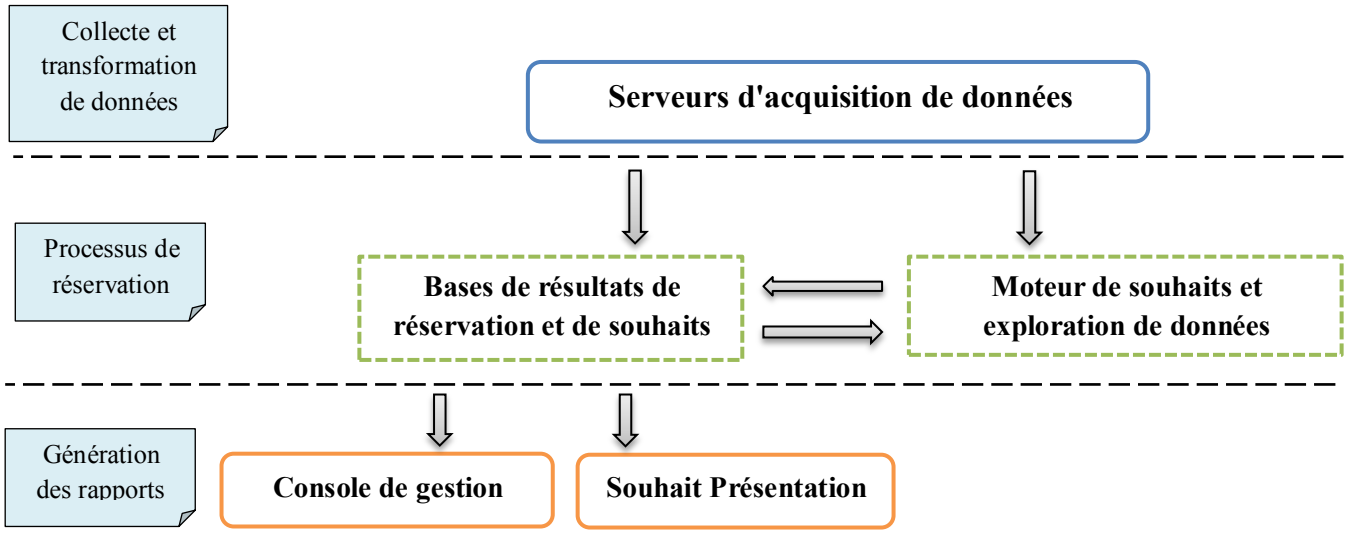


Figure IV.8. Cadre proposé pour l'analyse de système de réservation.

5.2. Moteur de souhait et exploration de données

Le moteur de souhait est décomposé en deux parties : base de règles d'association exactes et base des règles d'association approximatives. Pour cela, on doit donner d'abord quelques définitions.

- **Définition 1 (Règles d'association)**

Une règle d'association est une règle d'implication entre deux itemsets à laquelle sont associées la mesure de support. Elle définit la portée de la règle, et la mesure de confiance, qui définit la précision de la règle dans le contexte d'extraction. Le support et la confiance indiquent l'utilité et la pertinence de la règle et doivent donc être pris en considération lors de la définition des règles d'association redondantes. Une règle d'association r : $l_1. \rightarrow l_2$ de support s et de confiance c est notée r : $l_1^{s,c} \rightarrow l_2$. Une règle d'association r ∈ E est redondante si la règle r peut être déduite ainsi que son support s et sa confiance c de l'ensemble E \ r **[**Pasquier, 2000**].**

- **Définition 2 (Règles d'association redondantes)**

Soit un ensemble E de règles d'association. Une règle r : $l_1^{s,c} \rightarrow l_2$\in E est redondante si et seulement si E \ {r : $l_1^{s,c} \rightarrow l_2$} \models r : $l_1^{s,c} \rightarrow l_2$. Exemple 1 hôtel → restaurant, il est souhaitable que seules les règles d'association non redondantes minimales, qui sont les règles les plus utiles et les plus pertinentes, soient extraites et présentées à l'utilisateur. Une règle d'association est redondante si elle convoie la même information ou une information moins

générale que l'information convoyée par une autre règle de même utilité et de même pertinence. Une règle d'association r est non redondante minimale s'il n'existe pas une autre règle d'association r' possédant le même support et la même confiance, dont l'antécédent est un sous-ensemble de l'antécédent de r et la conséquence est un sur-ensemble de la conséquence de r **[**Pasquier_2, 00**].**

- **Définition 3 (Règles d'association non redondantes minimales)**

Soit l'ensemble AR des règles d'association extraites du contexte. Une règle d'association r : $l_1 \rightarrow l_2 \in$ AR est non redondante minimale s'il n'existe pas de règle d'association r' : $l'_1 \rightarrow l'_2 \in$ AR telle que support(r) = support(r'), confiance(r) = confiance(r'), $l'_1 \subseteq l_1$ et $l_2 \subseteq l'_2$.

En conséquence, ces règles constituent la base générique pour les règles d'association exactes et la base informative pour les règles d'association approximatives, et sont générées à partir des itemsets fermés fréquents et leurs générateurs.

5.2.1. Base générique pour les règles d'association exactes

Les règles d'association exactes, de la forme $r : l_1 \Rightarrow (l_2 \setminus l_1)$, sont des règles entre deux itemsets fréquents l_1 et l_2 dont les fermetures sont identiques : $\gamma(l_1) = \gamma(l_2)$. En effet, de $\gamma(l_1) = \gamma(l_2)$ on déduit que $l_1 \subset l_2$ et $support(l_1) = support(l_2)$, et donc $confiance(r) = 1$. Puisque l'itemset maximal parmi ces itemsets (qui possèdent le même support) est l'itemset $\gamma(l_2)$ (évident), tous les sur-ensembles stricts de l_1 qui sont des sous-ensembles de $\gamma(l_2)$ possèdent le même support, et les règles entre deux de ces itemsets sont des règles exactes **[**Pasquier, 00**].**

Soit l'ensemble $G\,\gamma_{(l2)}$ des générateurs de l'itemset fermé fréquent $\gamma\,(l_2)$. Par définition, les itemsets minimaux qui sont des sur-ensembles stricts de l_1 et des sous-ensembles de $\gamma(l_2)$ sont les générateurs $g \in G\,\gamma_{(l2)}$. On conclut que les règles de la forme $g \rightarrow (\gamma(l_2) \setminus g)$ entre les générateurs $g \in G_{\gamma(\ l2)}$ et l'itemset fermé fréquent $\gamma\ (l2)$ sont les règles d'antécédents minimaux et de conséquences maximales parmi les règles entre les sur-ensembles stricts de l_1 et les sous ensemble de $\gamma(l_2)$. La généralisation de cette propriété à l'ensemble des itemsets fermés fréquents définit la base générique constituée de toutes les règles d'association exactes non redondantes, selon la Définition 2, d'antécédents minimaux et de conséquences maximales **[**Pasquier, 00**].**

- **Définition 4 (Base générique pour les règles d'association exactes)**

Soit l'ensemble *FF* des itemsets fermés fréquents extraits du contexte et pour chaque itemset fermé fréquent *f* l'ensemble G_f des générateurs de *f*. La base générique pour les règles d'association exactes est :$BG = \{r : g \rightarrow (f \setminus g) \mid f \in FF \wedge g \in G_f \wedge g \neq f\}$. La condition $g \neq f$ est nécessaire car les règles entre un générateur *g* d'un itemset fermé fréquent *f* tel que $g = f$ sont de la forme $g \rightarrow \emptyset$ et n'appartiennent pas à l'ensemble des règles d'association valides (règles non informatives) **[**Pasquier, 2000**].**

5.2.2. Base informative pour les règles d'association approximatives

Les règles d'association approximatives, de la forme $l_1 \rightarrow (l_2 \setminus l_1)$, sont des règles entre deux itemsets fréquents l_1 et l_2 tel que la fermeture de l_1 est un sous-ensemble de la fermeture de l_2 : $\gamma(l_1) \subset \gamma(l_2)$. Les règles d'association approximatives non redondantes d'antécédent l_1 minimal et de conséquence ($l_2 \setminus l_1$) maximale sont déduites de cette caractérisation.

Soit l'itemset fermé fréquent f_1 qui est la fermeture de l_1 et le générateur unique g_1 de f_1 tels que $g_1 \subseteq l_1 \subseteq f_1$. Soit l'itemset fermé fréquent f_2 qui est la fermeture de l_2 et le générateur unique g_2 de f_2 tels que $g_2 \subseteq l_2 \subseteq f_2$. La règle $g_1 \rightarrow (f_2 \setminus g_1)$ entre le générateur g_1 et l'itemset fermé fréquent f_2 est la règle non redondante d'antécédent minimal et de conséquence maximale parmi les règles entre un itemset de l'intervalle $[g_1, f_1]$ qui contient tous les sur-ensembles de g_1 qui sont des sous-ensembles de f_1 et un itemset de l'intervalle $[g_2, f_2]$. En effet, le générateur g_1 est l'itemset minimal dont la fermeture est f_1, ce qui signifie que l'antécédent g_1 de cette règle est minimal, et la conséquence $(f_2 \setminus g_1)$ est maximale puisque f_2 est l'itemset maximal de l'intervalle $[g_2, f_2]$. La généralisation de cette propriété à l'ensemble des règles entre deux itemsets l_1 et l_2 définit la base informative qui est donc constituée de toutes les règle d'association approximatives non redondantes selon la Définition 1 d'antécédents minimaux et de conséquences maximales **[**Pasquier, 2000**].**

- **Définition 5 (Base informative pour les règles approximatives)**

Soit l'ensemble *FF* des itemsets fermés fréquents et l'ensemble *G* de leurs générateurs extraits du contexte. La base informative pour les règles d'association approximatives est : $BI = \{r : g \rightarrow (f \setminus g) \mid f \in FF \wedge g \in G \wedge \gamma(g) \subset f\}$. La base informative ne représente aucune perte

d'information car toutes les règles d'association approximatives valides dans le contexte peuvent être déduites ainsi que leurs supports et leurs confiances à partir des règles qui la constituent. De la définition de la base informative il est possible de déduire la réduction transitive de celle-ci qui constitue elle-même une base pour les règles d'association approximatives. Les règles transitives de la base informative sont de la forme $r : g \rightarrow (f \setminus g)$ pour un itemset fermé fréquent f et un générateur fréquent g tels que $\gamma(g) \subset f$ et $\gamma(g)$ n'est pas un prédécesseur immédiat de f : $\exists f' \in FF$ tel que $\gamma(g) \subset f' \subset f$, noté $\gamma(g) \int f$ **[**Pasquier, 2000**].**

- **Définition 6 (Réduction transitive de la base informative)**

Soit l'ensemble *FF* des itemsets fermés fréquents et l'ensemble *G* de leurs générateurs extraits du contexte. La réduction transitive de la base informative pour les règles d'association approximatives est : $RI = \{r : g \rightarrow (f \setminus g) \mid f \in FF \wedge g \in G \wedge \gamma(g) \int f\}$.

Il est possible de déduire toutes les règles d'association de la base informative ainsi que leurs supports et leurs confiances, et donc toutes les règles approximatives valides, à partir des règles de la réduction transitive. Cette réduction permet de diminuer le nombre de règles extraites en conservant les règles dont la confiance est la plus élevée puisque les règles transitives procèdent par construction des confiances inférieures aux règles non transitives **[**Pasquier, 2000**]**.

5.3. Algorithmes de génération des bases

La construction de la base générique pour les règles d'association exactes à partir de l'ensemble des itemsets fermés fréquents et leurs générateurs est présentée dans l'Algorithme1. De plus la construction de la réduction transitive de la base informative pour les règles d'association approximatives à partir de l'ensemble des itemsets fermés fréquents et leurs générateurs est présentée dans l'Algorithme 2. Les différentes abréviations utilisées sont présentées dans la table suivante.

Table. IV. 2. Différentes abréviations utilisées.

FF_k	Ensemble de *k*-groupes fréquents des *k*-générateurs. Chaque élément de cet ensemble possède trois champs : *générateur*, *fermé* et *support*
BG	*BG* Ensemble des règles d'association exactes de la base générique.
$Succ_g$	Ensemble des itemsets fermés fréquents qui sont des successeurs immédiats de la fermeture du générateur *g* considéré.
RI	Ensemble des règles d'association approximatives de la réduction transitive de la base informative.

5.3.1. Algorithme de base générique pour les règles d'association exactes

L'algorithme commence par initialiser l'ensemble *BG* avec l'ensemble vide. Chaque ensemble FF_k de *k*-groupes fréquents est ensuite examiné successivement. Pour chaque *k*-générateur $g \rightarrow FF_k$ de l'itemset fermé fréquent γ (*g*) pour lequel g est différent de sa fermeture γ (*g*), la règle r : $g \rightarrow (\gamma(g) \setminus g)$, dont le support est égal au support de g et γ (*g*), est insérée dans *BG*. L'algorithme retourne finalement l'ensemble *BG* qui contient toutes les règles exactes informatives entre un générateur et sa fermeture.

Algorithm 1 Base générique pour les règles d'association exactes.

Input: FF_k;

Output: *BG*;

1: **Initialize:** $BG \leftarrow \emptyset$;

2: **for** group FF_k **do**

3: **for** k-generators g $\in FF_k$ **where** $g \neq \gamma(g)$ **do**

4: $BG \leftarrow BG \cup \{(r : g \rightarrow (\gamma(g) \setminus g), \gamma(g).support)\}$;

5: **end for**

6: **end for**

7**: return** *BG*;:

5.3.2. Réduction transitive de la base informative pour les règles d'association approximatives

L'algorithme commence par initialiser l'ensemble *RI* avec l'ensemble vide. Chaque ensemble FF_k de *k*-groupes fréquents est ensuite examiné successivement dans l'ordre des valeurs de *k* croissantes. Pour chaque *k*-générateur $g \in FF_k$ de l'itemset fermé fréquent $\gamma(g)$, l'ensemble $Succ_g$ des successeurs de la fermeture de $\gamma(g)$ est initialisé avec l'ensemble vide et les ensembles S_j des *j*-itemsets fermés fréquents qui sont des sur-ensembles de $\gamma(g)$ pour $|\gamma(g)| < j \leq \mu$ sont construits. Les ensembles S_j sont ensuite considérés dans l'ordre croissant des valeurs de *j*.

Pour chaque itemset $f \in S_j$ dont aucun successeur immédiat de γ (*g*) dans $Succ_g$ n'est un sous-ensemble, *f* est inséré dans $Succ_g$ et la confiance de la règle $r : g \rightarrow (f \setminus g)$ est calculée. Si la confiance de *r* est supérieure ou égale au seuil minimal de confiance *minconfiance*, la règle *r* est

insérée dans *RI*. Lorsque tous les générateurs de taille inférieure à μ ont été considérés, l'algorithme retourne l'ensemble *RI*.

Algorithm 2 Génération de la réduction transitive de la base d'information.

Input: FF_k of the frequent k-groups of the k-generators;

Output: *RI*;

1: **Initialize:** $RI \leftarrow \emptyset$;
2: **for** ($k \leftarrow 1$; $k \leq \mu - 1$; k++) **do**
3: **for** k-generators g $\in FF_k$ **do**
4: $Succ_g \leftarrow \emptyset$;
5: **for** ($j \leftarrow |\gamma(g)|$; $j \leq \mu$; j++) **do**
6: $S_j \leftarrow \{f \in FF \mid f \supset \gamma(g) \wedge |f| = j\}$;
6: **end for**
7: **for** ($k \leftarrow 1$; $k \leq \mu - 1$; k++) **do**
8: **for** frequent closed itemset f $\in$ Sj **do**
9: **if** ($\nexists\, s \in Succ_g \mid s \subset f$) **then**
10: $Succ_g \leftarrow Succ_g \cup f$;
11 : $r.confiance \leftarrow f.support \,/\, g.support$;
12 : **if** $r.confiance \geq minconfiance$ **then**
13 : $RI \leftarrow RI \cup \{(r : g \rightarrow (f \setminus g), r.confiance, g.support\}$;
14 : **end if**
15 : **end for**
16 : **end for**
17: **end for**
18: **end for**

5.4. Implémentation

L'implémentation de notre proposition est composée de deux grandes parties : un système multi agent pour gérer les bases de données distribuées et complexes et la deuxième partie pour analyse les données qui aide à la décision, dans cette partie, on a proposé une base des résultats

qui permet de réduire la quantité des données et d'être plus précis dans le but de faciliter l'extraction des connaissances par l'extraction les itemsets fermé fréquents.

Réellement, on n'a pas pu implémenter cette approche pour les raisons suivantes :

- On n'a pas pu obtenir de véritables bases de données dans ce domaine pour vérifier l'efficacité de cette proposition, et il est difficile de générer de grandes quantités de données complexes et distribuées dans des ordinateurs à faible puissance (mémoire et processeurs).
- L'utilisation de fouille de données est besoin de calculateurs puissants pour trouver des résultats corrects et dans un temps raisonnable.

On peut dire que les résultats de l'étude seront satisfaisants et acceptables par rapport aux travaux précédents pour deux raisons :

Premièrement, la proposition de la base des réponses, qui est collectée à partir des différents entrepôts de données. Ce qui nous a permis de réduire la quantité de données destinées aux processus d'analyse. Après l'extraction des connaissances, on peut :

- Prévoir les demandes du client ;
- Décrire le comportement actuel des données et/ou ;
- Et prédire le comportement futur des données.

Tout cela en temps de traitement très court et il donne des connaissances plus précises.

Deuxièmes, l'utilisation de système multi agent permet de se dérouler un ensemble de processus informatiques en même temps. SMA est un système composé de : **[**Tanasescu, 04**]**

- Un environnement,
- Un ensemble d'objets de l'environnement qui peuvent être perçus, créés, détruits et modifiés par des agents.
- Un ensemble d'agents capables de percevoir, de produire, de consommer, de transformer et de manipuler les objets de l'environnement.
- Un ensemble de communications réunissant des agents.
- Enfin un administrateur chargé de contrôler l'activation des agents

Cette information nous confirme l'importance d'un système multi-agents dans notre projet car il réalise la mobilité constante de l'information sous le système.

3. Conclusion

Dans ce chapitre, nous avons présenté notre proposition dédiée à la fouille de données à partir des bases des données distribuées et hétérogènes. Cette approche d'extraction de données est basée sur un système multi agent qui permet d'assurer une intégration sémantique des bases des données distribués et hétérogènes et offre une satisfaction des souhaits des clients dans un temps raisonnable.

En effet, on a proposé dans un premier temps une architecture du système distribué et complexe (ex. système de réservation) et on a détaillé tous les composants de cette architecture. Par la suite, on a développé un cadre d'analyse le système dépend de règles d'association pour extraire les itemsets fermés fréquents à partir de la base des résultats.

Ensuite, on a détaillé la conception architecturale de chaque agent, et plus particulièrement le diagramme d'activité pour chaque agent pour exprimer le comportement de chaque agent et mieux expliquer le mode de fonctionnement de chaque agent.

Enfin, on a détaillé l'approche proposée dédiée à la fouille de données ou utilisé la méthode des règles d'associations qui permet d'extraire les itemsets fermés fréquents à partir de la base de résultats proposée. Pour faciliter l'extraction des règles, on a utilisé la méthode de Pasquier qui divise le moteur de souhait en deux parties : base générique pour les règles d'association exactes et base informative pour les règles d'association approximatives. De plus, on a présenté les algorithmes pour exprimer ce moteur proposé.

Chapitre V

Approche proposée pour la décision multicritère basée sur la requête Skyline

---

1. Introduction
    2. Approche de décisions multicritères basée sur la requête Skyline
3. Motivation et la de modélisation de l'architecture proposée
    - 3.1. Architecture de détection des éléments distribués et hétérogènes
    - 3.2. Description et modélisation de notre proposition
    - 3.3. Point de vue détaillé de l'architecture proposée
    - 3.4. Point de vue fonctionnel du système
    - 3.5. Commentaires des utilisateurs
4. Approche Skyline proposée pour le tourisme
    - 4.1. Formulation du problème
    - 4.2. Requête Skyline de réservation
5. Implémentation et évaluation des résultats
    - 5.1. Implémentation
    - 5.2. Collecte de données
    - 5.3. Analyse des performances
    - 5.4. Comparaison et résultats
6. Synthèse des résultats des expérimentations
7. Conclusion

---

1. Introduction

Les requêtes Skyline sont utilisées dans de nombreuses applications nécessitant des décisions multicritères sans utiliser de fonctionnalité cumulative pour identifier les meilleurs résultats en fonction des préférences des utilisateurs. Lors de la classification des tuples la requête Skyline est indépendante de la définition d'une fonction de score. L'explication de la requête Skyline et son concept nécessite des exemples pratiques. Par exemple, la réservation de tourisme en ligne permet aux clients de sélectionner leurs services touristiques en fonction de leurs besoins et de leur situation financière.

On présente dans ce dernier chapitre, la deuxième contribution qui vise à proposer une nouvelle approche de décision multi critère pour le traitement du problème des données complexes distribuées et hétérogènes lors de l'exploration de ce type de donnée. Il s'agit approche de décision multicritère basée sur la requête Skyline. Cette deuxième contribution est organisée en deux parties :

La première partie, sert à présenter l'approche proposée par la définition du modèle, de rechercher et sélectionner les meilleurs éléments en fonction des besoins des clients d'une part et le processus de recherche et de sélection d'autre part.

La deuxième partie, décrit l'étude expérimentale des algorithmes proposées de recherche et sélection qui a été implémenté en utilisant le langage de programmation Java sous l'environnement de développement Eclipse Hélios en tant que prototype de la proposition avec un réseau privé étaient composés de trois nœuds de calcul. Les données sont enregistrées dans la base de données MySQL. On va présenter les performances de notre étude en termes du nombre et de qualité des éléments sélectionnés ainsi que le temps de recherche et de sélection. Pour cela, on établira une comparaison des résultats avec la sélection classique.

2. Approche de décision multi critère basé sur la requête Skyline

La requête Skyline [Stephan, 01] est une approche populaire pour minimiser l'espace de recherche afin de sélectionner seulement quelques objets de données présentant certaines caractéristiques. De nombreuses méthodes ont été développées pour les requêtes de jointure Skyline dans des environnements distribués et non distribués, où les données sont stockées dans plusieurs tables nécessitant l'existence d'opérations de jointure entre elles pour calculer le Skyline

final, puis proposer des approches efficaces pour partager le coût de traitement de jointure avec le coût de calcul de Skyline [Vlachou, 06], [Sun, 08], [Jin, 06], [Kalyvas, 17] et [Zhang, 16].

En bref, les études antérieures sur Skyline se limitaient à une ou plusieurs relations dans des environnements distribués et non distribués. De plus, ces approches n'étaient pas adaptées au fait que les systèmes IoT sont naturellement distribués. Notre travail se concentre sur l'exploitation de la nature distribuée de l'IoT pour utiliser une jointure de dynamique Skyline locale au niveau de la passerelle, puis pour utiliser une jointure de dynamique Skyline globale au niveau du serveur pour un exemple typique (c'est-à-dire services de réservation). L'objectif principal consiste à de rechercher et de sélectionner les meilleurs éléments disponibles en fonction des requêtes des utilisateurs.

Notre approche proposée se concentre sur l'application du principe des requêtes Skyline aux services de réservation pour aider les utilisateurs à sélectionner le meilleur service disponible dans un contexte IoT utilisant une architecture distribuée. Il est procédé à une sélection parmi les meilleurs éléments d'un pack de base de données prise en compte de deux facteurs : la qualité des données et le temps minimal de recherche et de sélection.

L'objectif principal de cette étude est de proposer une approche efficace pour rechercher et sélectionner les meilleurs éléments dans le contexte multi décision (comme exemple services de réservation) en fonction des besoins des clients dans les plus brefs délais [Maamra, 22].

3. Motivation et la de modélisation de l'architecture proposée

3.1. Architecture de détection des éléments distribués et hétérogènes

Dans ce travail, la structure proposée est divisée en trois sections. Dans la première section, la demande des utilisateurs est d'abord introduite dans le système d'interface. Dans la deuxième section, le serveur global utilise une Skyline de sélection globale (GSS) en rassemblant les résultats de tous les serveurs locaux et fournit ensuite la réponse finale en fonction des besoins spécifiés des clients.

De plus, le serveur de cette architecture est relié à une base de données des éléments préalablement établie à partir des sélections des clients ; cette base de données est essentiellement utilisée pour analyser les différentes exigences des éléments préférées par l'utilisateur en utilisant

des méthodes telles que l'exploration de données. Cette approche aide à prendre des décisions pour améliorer les services pour gagner autant de clients que possible.

Dans la troisième section, le Skyline de sélection locale (LSS) est utilisée au niveau du serveur local. Cette section est constituée de passerelles distribues dans le réseau et chaque passerelle connectée à un serveur local en Cloud computing. Chaque serveur local gère un entrepôt de données de différents éléments selon le domaine d'application, et chaque serveur local répond localement aux demandes des utilisateurs. Ainsi, dans cette étude, on peut réduite la quantité de données à traiter lors de la recherche et les meilleurs éléments sont sélectionnés. La figure V.1 montre l'architecture proposée. Les détails des trois sections sont fournis ci-dessous :

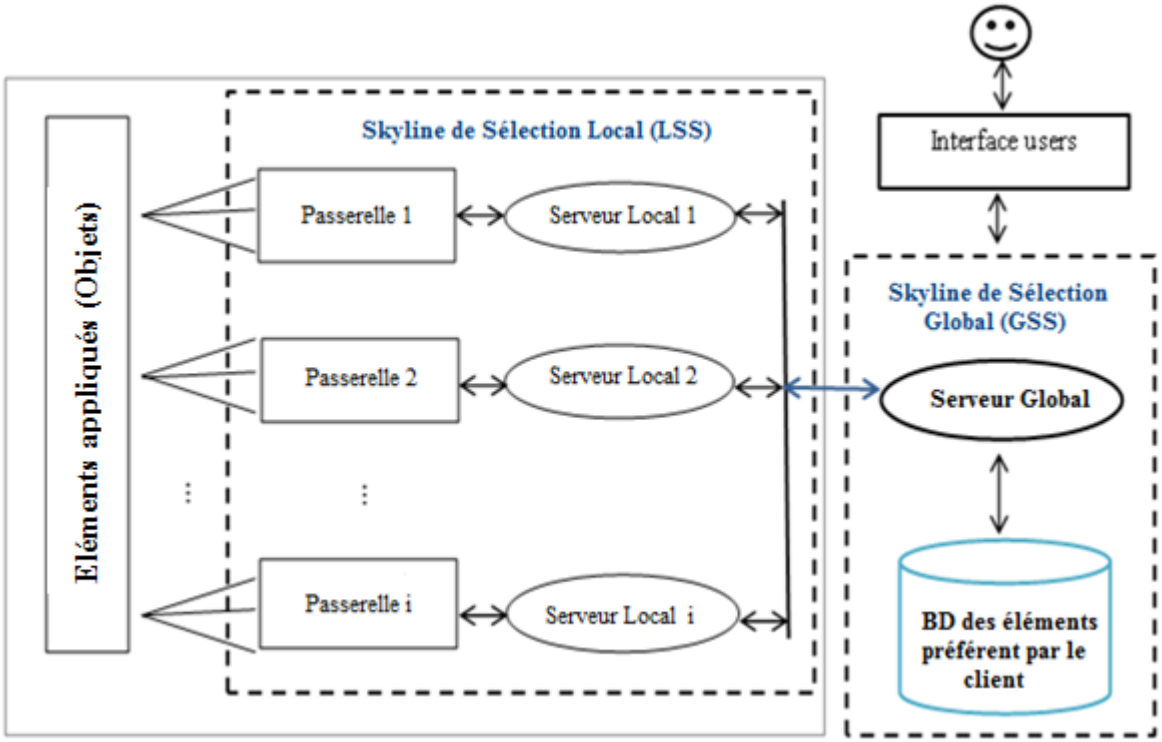


Figure V. 1. Architecture de détection des meilleur éléments au préfèrent d'utilisateur.

Section 1 - Les demandes et les réponses des utilisateurs sont introduites dans le système d'interface.

Section 2 - Le GSS comprend un serveur global et une base de données.

- *Serveur global:* Ce serveur traite les demandes des consommateurs, combine les résultats de tous les serveurs locaux, fournit la réponse finale aux utilisateurs et stocke les sélections préférées des utilisateurs.
- *Base de données des choix préférés des utilisateurs:* La base de données est liée au serveur global et est utilisée pour analyser les différents éléments préférés par les utilisateurs à l'aide de l'exploration de données.

Section 3 - Le LSS comprend les serveurs locaux, les passerelles et les éléments appliquées selon le domaine d'application.

- *Serveurs locaux:* Ils sont responsables de l'envoi des requêtes du serveur et de la collecte des données à partir des passerelles. De plus, ils gèrent un entrepôt de données de différentes éléments.
- *Passerelle:* La passerelle est responsable de la navigation des données des éléments appliquées au serveur local dans le Cloud et inversement. Il prétraite et filtre également les données avant les transférer vers le serveur local. Cette étape réduit la quantité de données à traiter et à stocker. Outre, la passerelle gère les mises à jour des capteurs des éléments appliquées.
- *Eléments appliquées :* sont des objets tels que les services touristiques (hôtels, avions, voitures et restaurants) dans le domaine de réservation de voyage, les produits de marketing numérique qui aide les entreprises découverte de tendances pour le marketing, outils d'apprentissage à distance, etc. Un capteur est monté sur chacun de ces objets pour la collecte de données. Ces objets transférez les données sur un réseau à l'aide d'actionneurs. Les capteurs associés à ces objets envoient leurs mesures aux passerelles correspondantes.

3.2. Description et modélisation de notre proposition

Le problème abordé dans notre étude est de savoir comment rechercher et sélectionner un sous-ensemble des meilleurs éléments dans un espace de recherche en fonction des conditions des clients à partir de grands groupes des éléments appliqués offerts sur Internet.

L'utilisateur détermine la caractéristique critique (condition d'utilisateur) qui aide à sélectionner les éléments qui correspond à la situation du client. Par conséquent, ce problème

devrait être modélisé de manière adéquate pour aider à représenter le problème de la recherche et à sélectionner efficacement les éléments les plus efficaces parmi les éléments appliqués dans l'espace de recherche. Par conséquent, on propose une technique efficace pour résoudre le problème qui a été abordé.

Notre modèle montre comment un utilisateur envoie une demande et comment la recherche et la sélection de la meilleure solution des éléments appliqués sont effectuées (comme sur la figure V.2).

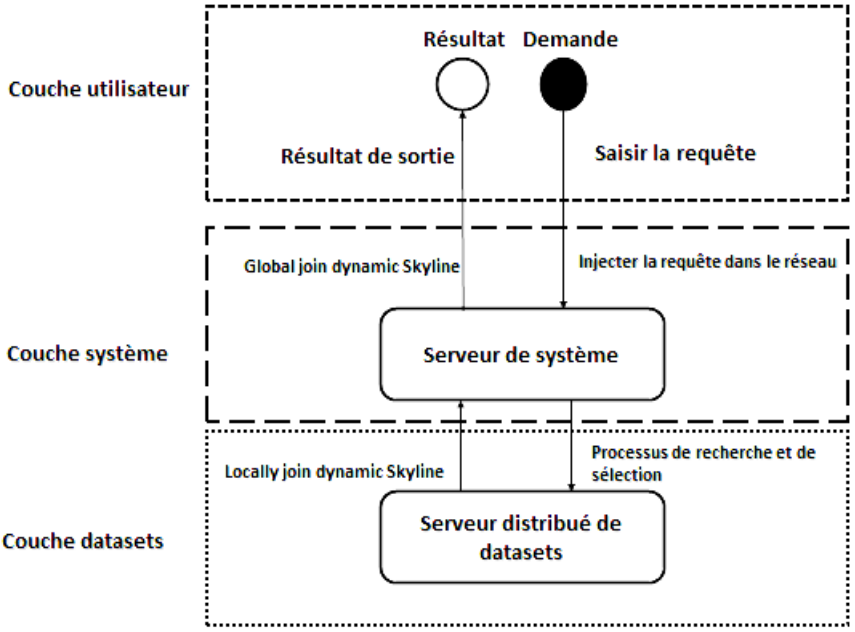


Figure V. 2. Processus proposés configurant les requêtes pour rechercher et sélectionner les meilleurs éléments.

Ce processus contient trois couches: la couche Utilisateur, la couche Système et la couche datasets.

- *Couche utilisateur :* Dans cette couche, l'utilisateur envoie une demande spécifique au couche système. Lorsque toutes les étapes de la demande sont effectuées, l'utilisateur reçoit les meilleurs résultats et selecte son préfère.
- *Couche système* : Dans cette couche, le serveur global injecte la demande de l'utilisateur dans le réseau pour accéder aux serveurs de base de données disponibles pour rechercher et sélecter. Après avoir reçu les résultats de chaque serveur local, le serveur global

fusionne les résultats sélectionnés. Ensuite, il sélectionne le meilleur résultat et l'affiche à l'utilisateur pour déterminer son préfère.

- *Couche des datasets :* Dans cette couche, le serveur local de chaque base de données recherche et sélectionne les informations appropriées pour le client. Puis, chaque serveur local envoie ses résultats au serveur global.

3.3. Point de vue détaillé de l'architecture proposée

On présente dans la figure V.3, les principales constructions du diagramme de classes via l'outil Visual Star UML [5.02] pour les détails de l'architecture proposée illustrée.

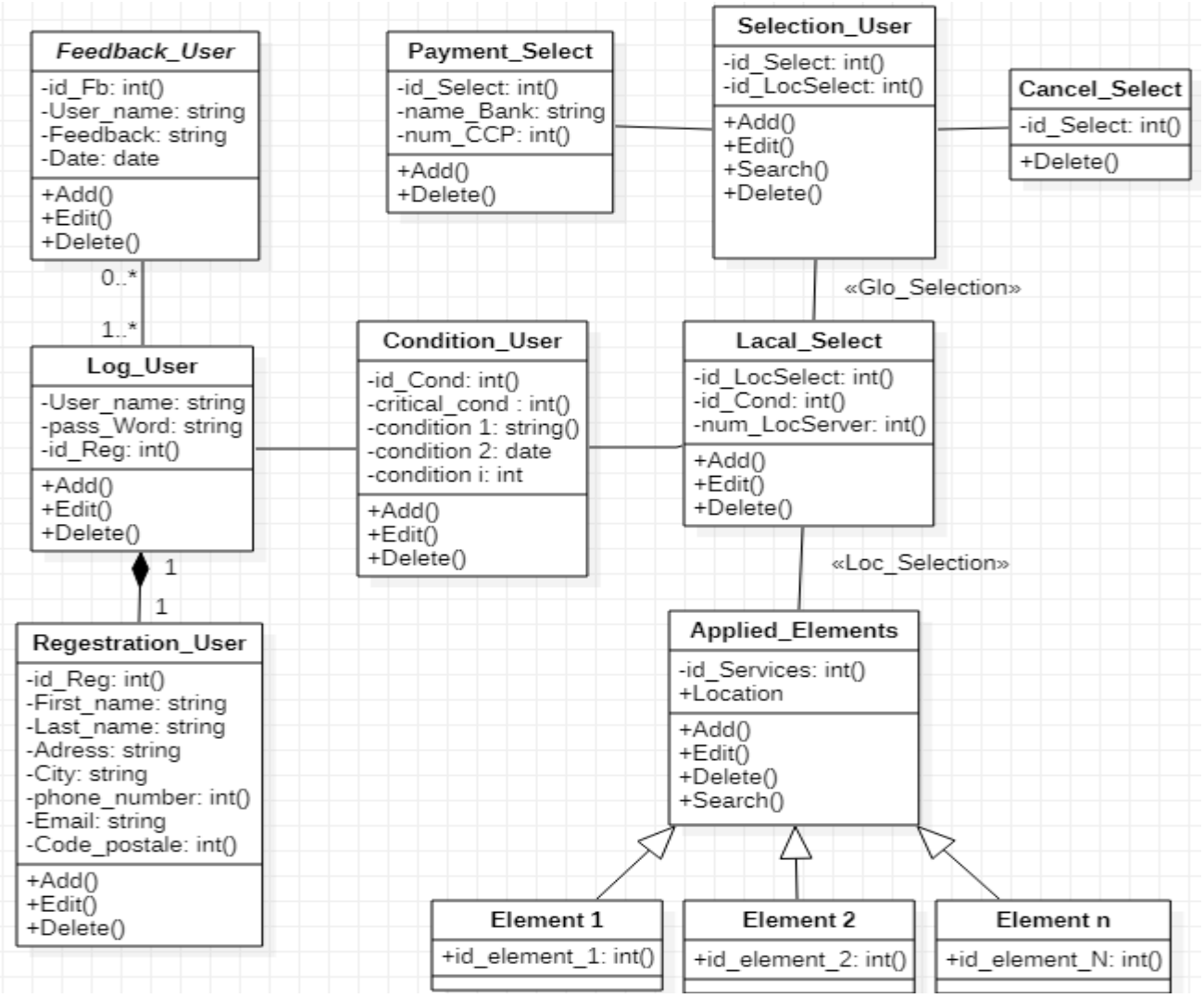


Figure V. 3. Détails de l'architecture proposée.

3.4. Point de vue fonctionnel du système

Dans ce point, On présente un point de vue "fonctionnel" de l'architecture proposée. Grâce à des cas d'utilisation, on obtiendra suffisamment d'informations sur les composants du système pour définir les limites du système comme sur la figure V.4.

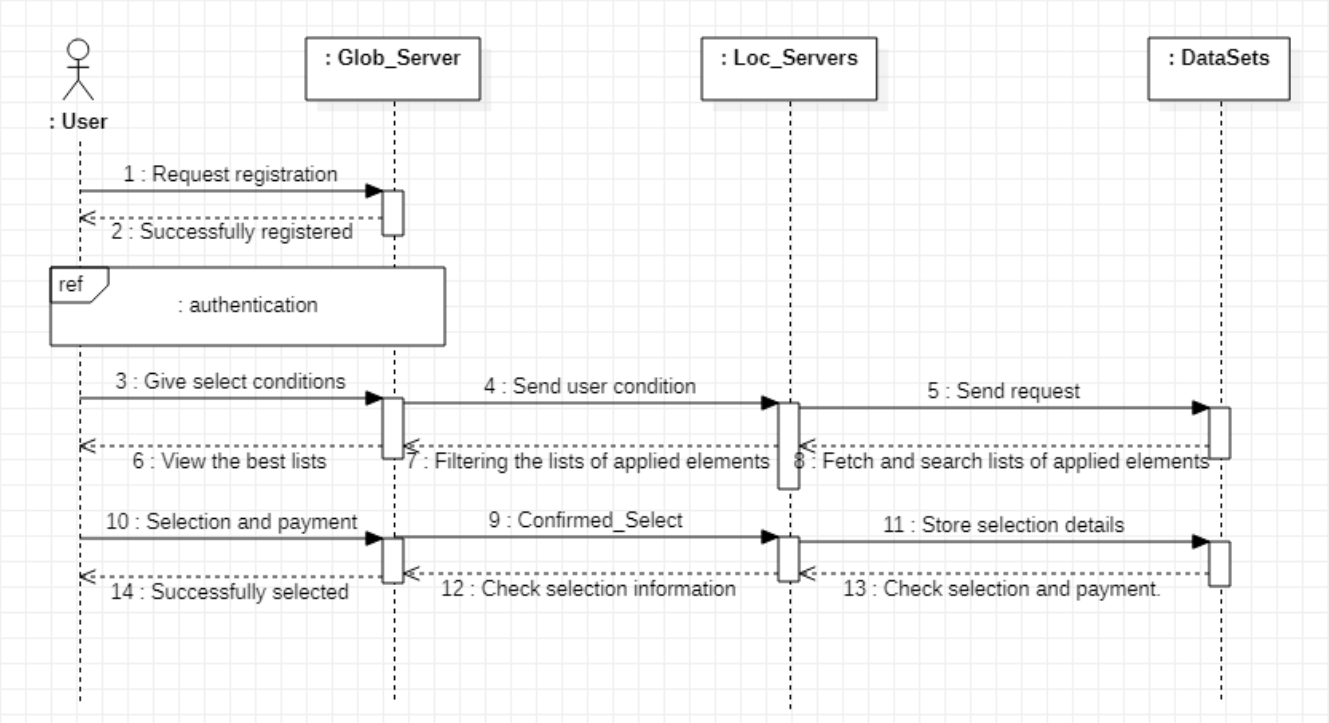


Figure V. 4. Diagramme de séquence de sélection des éléments appliqués.

3.5. Commentaires des utilisateurs

En ce qui concerne la réponse, la sélection et le commentaire de l'utilisateur sont stockés dans la base de données des sélections préférées des clients; cette base de données est utilisée pour analyser les différents éléments préférés de l'utilisateur. La figure V.5 montre le diagramme de séquence de la réponse proposé.

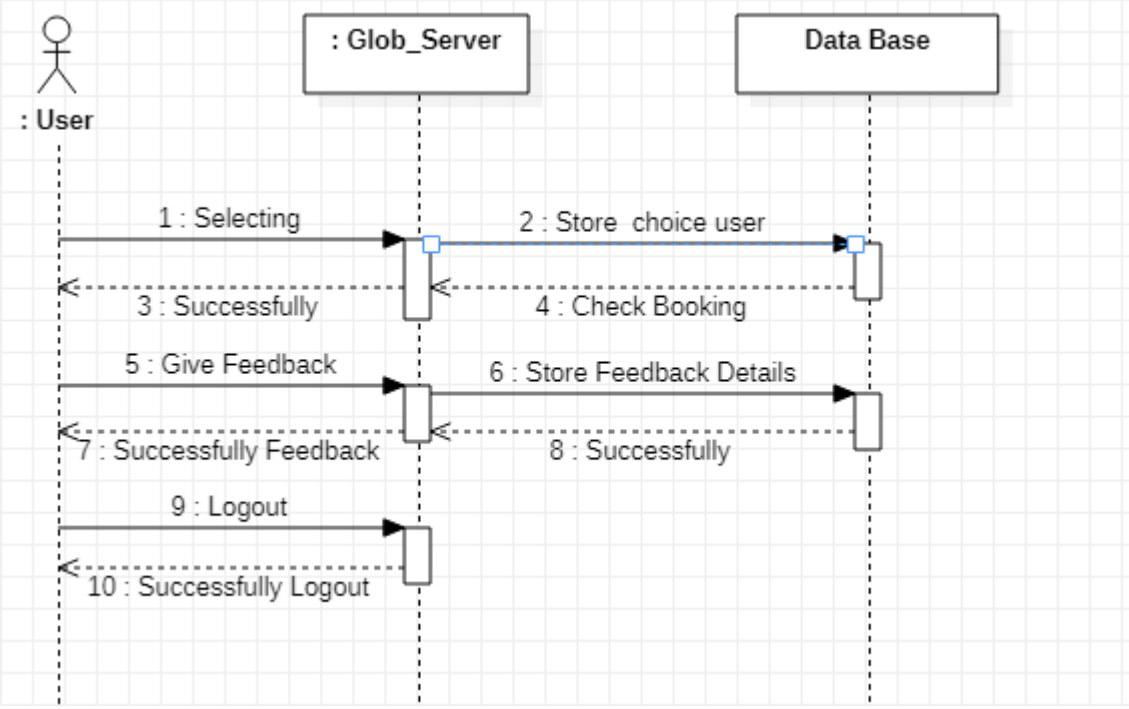


Figure V. 5. Diagramme de séquence de la réponse.

4. Approche Skyline proposée pour le tourisme

La requête Skyline et son concept peuvent être expliqués en détail à l'aide d'exemples pratiques. Par exemple, la réservation de tourisme en ligne permet aux clients de sélectionner soigneusement leurs services touristiques en fonction de leurs besoins et de leur situation financière ; de multiples choix et options sont disponibles pour des services de réservation adaptés et des services à faible coût.

Dans cette section, on discute en détail l'approche proposée pour la réservation de voyages à l'aide de requêtes Skyline dans un environnement IoT comme un exemple applicable dans notre étude.

4.1. Formulation du problème

Soit B un ensemble de services de réservation classés en m services, B= {B_1, B_2, B_3, ..., B_m}. Chaque service a une relation dans la base de données. B_1= {h_1, h_2, h_3, ..., h_n} ϵB représente la relation des hôtels, B_2={a_1, a_2, a_3, ..., a_k},ϵB représente la relation des avions, et B_3={r_1, r_2, r_3,..., r_d},ϵB représente la relation des restaurants, etc. Chaque type de service de réservation est classé en services permanents, qui sont efficaces tout au long de la période de voyage, ou en

services temporaires, qui sont efficaces à un moment précis pendant le voyage. Les services permanents sont annotés à l'aide du signe " + " (par exemple, B+), tandis que les services temporaires sont annotés à l'aide de la marque " − " (par exemple, B−).

Soit C un ensemble de conditions d'utilisateur, où les caractéristiques communes des services de voyage représentent des attributs de jointure J entre les relations; soit d désigne un attribut de notation dynamique entre les relations. Soit K un ensemble d'attributs numériques dans le schéma de chaque relation, où K= ($K_1 \cup K_2 \cup K_3 .... \cup K_m$).

Par exemple, le nombre de services est m=3 (R_1=hotels, R_2=planes, R_3=restaurant), C= {Travel_money, Location, start_date, date_retoure}; dans ce cas J= {Location, date_travel}, D= {Travel_money}, K_1= {price, nb_star}, K_2={price, class}, et K_3={price, quality}.

4.1.1. Espace de recherche de la requête Skyline de réservation

Pour définir l'espace de recherche du LBS, on a une requête <K, J, d> et soit B une table. On peut définir l'espace de recherche de booking Skyline BS en réalisant les conditions suivantes :

a) $BS = B_1 \rhd\lhd B_2 \rhd\lhd B_3 ... B_m$, $B_i \in BS$;

b) $\forall att_i \in (k \cup d) \rightarrow \exists B_i (B_i \in BS) \wedge (att_i \in B_i)$;

c) $\forall B_i (B_i \in BS) \rightarrow \exists att(att_i \in B_i) \wedge (att_i = price_i)$;

d) Dans ce cas, d est le prix total des services de réservation. Par conséquent, pour le meilleur espace de recherche, on définit d'abord le temps de trajet à l'aide de la variable t, où t = date_retoure − start_date; on utilise la variable t avec les services permanents B+:

$\forall B_i+ (B_i+ \in BS) \rightarrow \exists price_{i+}(price_i+ \in B_i+) \wedge (price_{i+} = price_i \times t)$

Pour les services temporaires B -, on utilise la définition suivante :

$\forall B_i\text{-}(B_i\text{-} \in BS) \rightarrow \exists price_i\text{-}(price_i\text{-} \in B_i\text{-}) \wedge (price_i\text{-} = price_i)$

Ensuite, on ne peut pas déterminer BS et les prix totaux sont inférieurs ou égaux à l'attribut dynamique :

$$\sum_{i=1}^{m} price_i^+ + \sum_{i=1}^{m} price_i^- \leq d$$

e) BS' qui a deux propriétés (décrites ci-dessus) et moins les tables autres que BS sont introuvables.

$$\nexists \sum_{j=1}^{m} price_j^+ + \sum_{j=1}^{m} price_j^- \leq \sum_{i=1}^{m} price_i^+ + \sum_{i=1}^{m} price_i^- , j \neq i$$

f) L'espace de recherche de Skyline de réservation distribué β pour une requête <K, J, d> est défini comme suit :

$$\beta = BS_1 \cup BS_2 \cup BS_3 . \cup BS_n, BS_i \in \beta$$

4.1.2. Skyline domination de réservation locale

Pour une requête de Skyline de réservation donnée <K, J, d> et son espace de recherche BS, supposons $t_1 = (v_0, v_1, v_2,..., v_d)$ et $t_2 = (v'_0, v'_1, v'_2,..., v'_d)$ sont deux tuples de BS, t_1 domine t_2 si $\forall att_i (att_i \in K) \wedge (att_i = price_i) \rightarrow |d - \sum_{i=1}^{m} v_i| < |d - \sum_{i=1}^{m} v'_i| \wedge \exists att_j (att_j \in K) : v'_j \leq v_j$.

Pour plus de clarté, on utilise $\mathbf{t_i} >_{\mathbf{dom}} \mathbf{t_j}$ pour indiquer que le tuple ti réservation Skyline domine t_j et $\sum_{i=1}^{m} \text{price}_i \prec_d \sum_{j=1}^{m} \text{price}_j$ pour indiquer que t_i booking Skyline domine t_j dans la somme des services de prix pour d.

4.1.3. Domination Skyline de la réservation globale

Le jeu de résultats global G pour une requête Skyline de réservation Q = <K, J, d> dans son espace de recherche β doit avoir les propriétés suivantes :

a) $\forall t_i \in G \rightarrow \exists t_j (t_j \in \beta) \wedge t_j \succ_{dom} t_i$

b) Supposons que $t_i \in G$ et t_i = (v0, v1,..., vm); si $att_i \in K$, vi doit satisfaire aux conditions précédent.

4.2. Requête Skyline de réservation

Dans cette section, on propose une approche de sélection des services de réservation touristique utilisant des bases de données distribuées dans un environnement IoT.

Notre algorithme de calcul de Skyline de réservation permet de sélectionner les meilleurs services touristiques en fonction des exigences des utilisateurs en termes de coût du voyage, de localisation et de temps de trajet. Pour la jointure (tables) des services dans passerelles, on définit l'emplacement comme un attribut de jointure entre différentes tables, ce qui conduit au premier filtrat dans les tables.

Tables join est une extension de la jointure Skyline originale. La requête Skyline dynamique est adaptée pour minimiser autant que possible l'espace de recherche pour améliorer l'efficacité de la sélection des services de réservation en fonction des conditions de l'utilisateur. Le Skyline de réservation est un ensemble de tous les services qui ne sont dominés dynamiquement par aucun autre service par rapport à la distance d'un service de requête donné. En particulier, on se concentre sur le Skyline des services de réservation pour la sélection de services basée sur les utilisateurs, où ceux réservés via Skyline dominaient les services sans Skyline de réservation. Ceci est attribué au fait que le Skyline des services de réservation permettent de meilleurs résultats pour les utilisateurs que les services sans Skyline de réservation.

Les étapes d'exécution de la procédure Skyline proposée sont spécifiques dans les instructions de l'Algorithme 1. Le tableau V.1 donne la définition des différentes abréviations. Comme indiqué dans la section 4.1, les services de réservation comprennent des processus de recherche et de sélection. Il se compose de quatre étapes.

Table V. 1. Différentes abréviations utilisées.

Abréviations	**Définition**
Si	Local server i
Q	User query
GBS	Global booking Skyline
GBSO	Ordering booking services of GBS
LBS(Si)	Local booking Skyline of Si
BSi	Dataset of the booking server Si
Bi	Table of the travel booking service i
V	Indicates whether service i not temporary during travel
J	Join attribute
Ki	Numerical attributes of Bi
Gi	Group tuples of Bi by J
D	Scoring attribute
LG	Temp table for union selected tuples of different tables

4.2.1. Capture de la demande de l'utilisateur

Une fois qu'un consommateur se connecte à l'interface utilisateur, il saisit ses besoins via une interface Web. Ensuite, le gestionnaire d'interface transmet la requête au serveur, qui l'envoie aux passerelles.

Algorithm 1 Best booking service selection at local servers.

Input: Local Servers S, Q;

Output: GBS and GDSR;

1: **Initialize:**

2: Q: capturing the user request;

3: **for** each Si **do**

4: LBS(Si): computing the local booking Skyline (Si, Q);

5: **end for**

6: GBS: computing the global booking Skyline from LBS;

7: GBSR: ordering booking services of GBS;

4.2.2. Calcul du LBS

Une fois que les passerelles ont reçu la demande de l'utilisateur, chaque serveur local des différentes passerelles calcule le LBS. L'Algorithme 2 illustre le processus de LBS comme suit.

Les exigences de l'utilisateur sont divisées en trois parties: la première partie pour joindre les relations, la deuxième partie pour déterminer l'attribut de notation entre les relations et la troisième partie représente les attributs numériques de chaque relation. De plus, l'utilisateur définit l'attribut de notation, qui est le point principal des algorithmes de Skyline de réservation.

Algorithme 2 Local booking Skyline

Input: BS, v, K, J, d

Output: result of the local booking Skyline

1: **Initialize:**
2: **for** Bi ∈ BS **do**
3: group tuples of Bi by J
4: **if** v **then**
5: **for** tj of Gi **do**
6: pricej = pricej. t
7: LG = LG join Gi
8: **end for**
9: **for** each tuples ti of LG **do**
10: **if** sum (pricek) ≤ d **then**
11: **for** ki∈ Ki **do**
12: **if** ∈ k'j ∈ Ki and k'j ≠ pricei: k'j < ki **then**
13: record ti into LBS
14: **end for**
15: **end for**
16: **end for**
17: return LBS

Après l'arrivée de la demande de l'utilisateur, le groupe d'algorithmes LBS commence l'identification des tuples à l'aide de l'attribut join. Il s'agit de la première opération de filtrage des services de réservation. Ensuite, On vérifie le type de service ; s'il n'est pas temporaire pendant le voyage, on multiplie l'attribut de prix du tableau B_i à une période de voyage t.

Le deuxième processus de filtrage de réservation des données commence à partir de la table temporaire LG. Pour chaque tuple ti de LG, on calcule le coût total du voyage et on le compare avec l'attribut de notation de l'utilisateur d ; s'il est d'accord avec la condition, on applique la requête Skyline aux attributs numériques. Si le tuple est vérifié toutes les conditions, on l'ajoute dans le résultat du calcul LBS.

4.2.3. Calcul du GBS

Après avoir conclu le résultat des serveurs locaux LBS, le résultat est transmis au serveur global, qui fusionne les résultats de tous les services de réservation et calcule le GBS. L'algorithme 3 illustre le processus de détermination du GBS comme suit :

Algorithm 3 Global booking Skyline

input: S, J, K, d

output: result of the global booking Skyline

```
1: Initialize:
2: for Si ∈ S do
3:     LBSi = LocalBookingSkyline (Si, J, K, d)
4:     GBS = GBS ∪ LBSi
5: end for
6: for each tuples ti of GBS do
7:     for each tuples tj of GBS do
8:         if tj dominated by ti then
9:             remove tj from GBS
10:        else
11:            remove ti from GBS
12:    end for
13: end for
14: return GBS
```

Un algorithme GBS s'exécute juste après la collecte des résultats des serveurs locaux à partir de passerelles distribuées. Cet algorithme parvient à combiner tous les résultats dans une table GBS temporaire. Ensuite, la requête Skyline est appliquée aux tuples GBS, où supprimer les tuples dominés par un autre tuple.

4.2.4. Classement de services de réservation

Immédiatement après le calcul du GBS, le résultat final est intégré dans l'algorithme de classement des résultats de réservation. Ensuite, le résultat devient accessible pour être affiché à l'utilisateur.

5. Implémentation et évaluation des résultats

Dans cette section, on présente l'implémentation de notre proposition. De plus, on discute les résultats de l'analyse des différents cas utilisés. Enfin, on fait une comparaison entre les performances du notre modèle de Skyline proposé et celles de l'algorithme itératif [Sun, 2008].

5.1. Implémentation

Nos algorithmes proposés qui ont été implémentés en Java Eclipse Hélios en tant que prototype de la proposition avec un réseau privé était composé de trois nœuds de calcul. Les données sont enregistrées dans la base de données MySQL. Le premier formulaire permet aux utilisateurs de soumettre leurs préférences pour plusieurs services de réservation et de voyage (hôtels, avions, restaurants, etc.) selon l'attribut de notation dynamique. Dans ce cas, pour obtenir les meilleurs résultats avec un temps de recherche minimum, le total de paiement utilisé par le client est prédéfini. L'environnement matériel utilisé pour les expériences consiste en un ordinateur personnel doté d'un processeur Intel Core i5 à 3,2 GHz avec 4 Go de RAM.

5.2. Collecte de données

L'IOT est crucial pour l'exécution de la requête proposée; par conséquent, il est considéré comme un environnement d'intégration pour les informations liées à tous les services de voyage. En fait, il n'y a pas des data sets existants avec les données de divers services touristiques ; par conséquent, on génère une data set synthétique de 100 000 tuples sur trois serveurs locaux. Les expériences ont été répétées dix fois, et on a pris la moyenne pour analyser les résultats.

5.3. Analyse des performances

Pour l'analyse des performances, la requête proposée est examinée en fonction de ses multiples paramètres. Le facteur principal sur la base duquel toute requête est évaluée est le temps de traitement. Par conséquent, l'impact de la modification du nombre de services de réservation et de la taille des ensembles de données sur le temps de traitement est analysé. Enfin, l'impact du nombre de services sur le nombre de réservations en fonction des préférences des clients est étudié.

5.3.1. Effet du nombre de services sur le délai de traitement

Lors de l'évaluation de l'effet du nombre de services sur le temps de traitement, on a mesuré les LBS et les GBS séparément, le nombre de services utilisé passe de 2 à 6 services et la taille des ensembles de données est fixée à 50 000 tuples lors toutes les expériences dans cette évaluation.

Les résultats d'évaluation des performances qu'on a obtenus ont montré que l'augmentation du nombre de services a augmenté le temps de traitement pour LBS et GBS. Cependant, l'augmentation du temps de traitement de LBS a été plus élevée que le cas de GBS en raison de l'augmentation des opérations conjointes entre différents services (tableaux) au sein de LBS elle-même (comme sur la figure V.6).

Figure V. 6. Effet du nombre de services sur le temps de traitement des Skylines de réservation locales et globales.

5.3.2. Effet de la taille des data sets sur le temps de traitement

Dans ce cas, on a examiné l'impact de la taille des data sets sur le temps de traitement des LBS et GBS. Dans chaque expérience on augmente la taille des data sets par génération de nouvelles données, en début de 1000 tuples à 100 000 tuples.

La figure V.7 illustre l'évolution du temps de traitement requis avec une augmentation de la taille des data sets en fonction des besoins de l'utilisateur. Les résultats de l'analyse indiquent que

lorsque la taille des data sets était inférieure à 1000 tuples, la différence dans les temps de traitement des LBS et des GBS diminue considérablement.

Mais, immédiatement que la taille des data sets a dépassé 1000 tuples, le temps de traitement a augmenté jusqu'à atteindre la taille des data sets à 50000 tuples.

Au-delà de 50 000 tuples, la différence de temps de traitement entre LBS et GBS est significativement importante. Depuis le temps de traitement est devenu constant dans le cas de GBS et a continué d'augmenter dans le cas de LBS, en raison de l'augmentation des processus de recherche et de sélection au niveau des serveurs locaux.

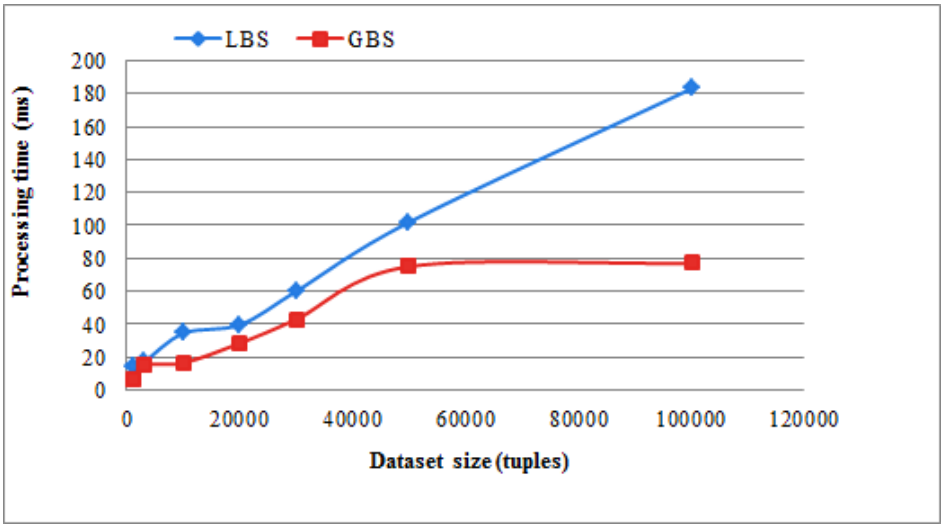


Figure V. 7. Effet de la taille des data sets sur le temps de traitement des Skylines de réservation locales et globales.

5.3.3. Effet du nombre de services sur le nombre de réservation sélectionné

La figure V.8 montre l'impact du nombre de services touristiques (tableaux) sur le nombre de résultats de sélection des services de réservation. Au cours de ces expérimentations, l'exigence de l'utilisateur principal est maintenue constante dans tous les cas lorsque le nombre de services est modifié. Les résultats obtenus révèlent clairement que l'augmentation du nombre de services a considérablement augmenté le nombre de résultats de réservation.

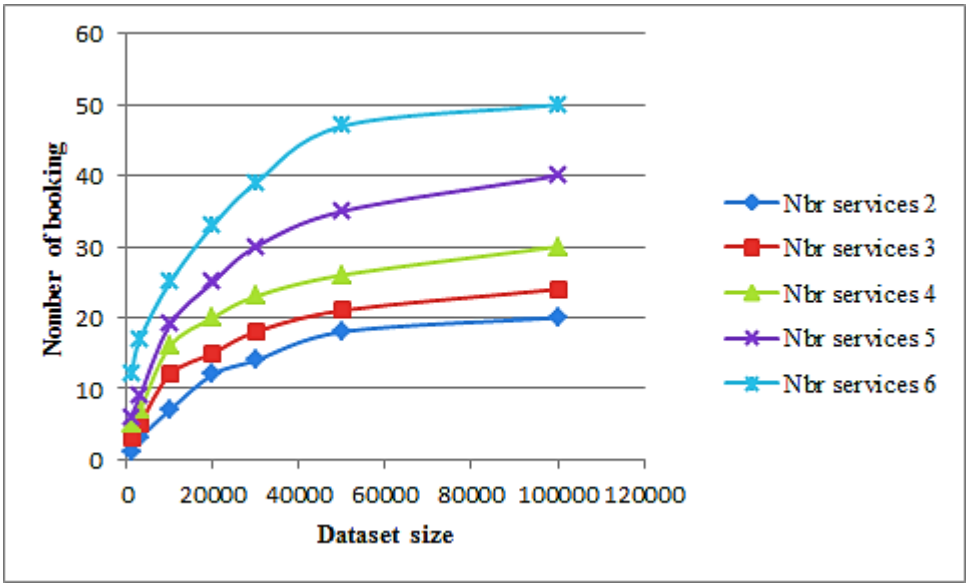


Figure V. 8. Effet du nombre de réservations sélectionnées pour un nombre croissant de services et de la taille de data sets.

5.4. Comparaison et résultats

Nous avons réalisé deux études comparatives de notre approche : une comparaison quantitative et une comparaison qualitative. Le but de la première comparaison est de montrer que la jointure de dynamique Skyline utilisées ont conduit à des meilleures performances par rapport à d'autres Skyline existantes. La seconde comparaison (comparaison qualitative) permet de montrer l'originalité et les performances de notre approche par rapport aux autres approches similaires.

5.4.1. Comparaison quantitative

Les étapes de l'algorithme itératif peuvent être décrites comme suit:

- Calculer les points de Skyline locaux pour chaque table,
- Calculer le résultat de la jointure partielle de Skyline,
- Compiler les résultats de la jointure partielle de Skyline.

En revanche, le modèle de Skyline proposé a commencé par calculer le Skyline dynamique de jointure local, compiler ses résultats et calculer immédiatement le Skyline dynamique de jointure global résultant. Pour comparer les performances du modèle Skyline proposé avec le

modèle de SaLSa en termes de temps de traitement, la taille des data sets utilisés est de 500 000 et le nombre de services de réservation est 6 services.

La figure V.9 montre les résultats, qui indiquent clairement que le modèle Skyline proposé permet d'obtenir un meilleur temps de traitement que l'algorithme itératif. Parce que le calcul répété entraîne une consommation d'énergie élevée avec un temps de traitement plus long, c'est pourquoi le modèle proposé surpasse l'algorithme itératif.

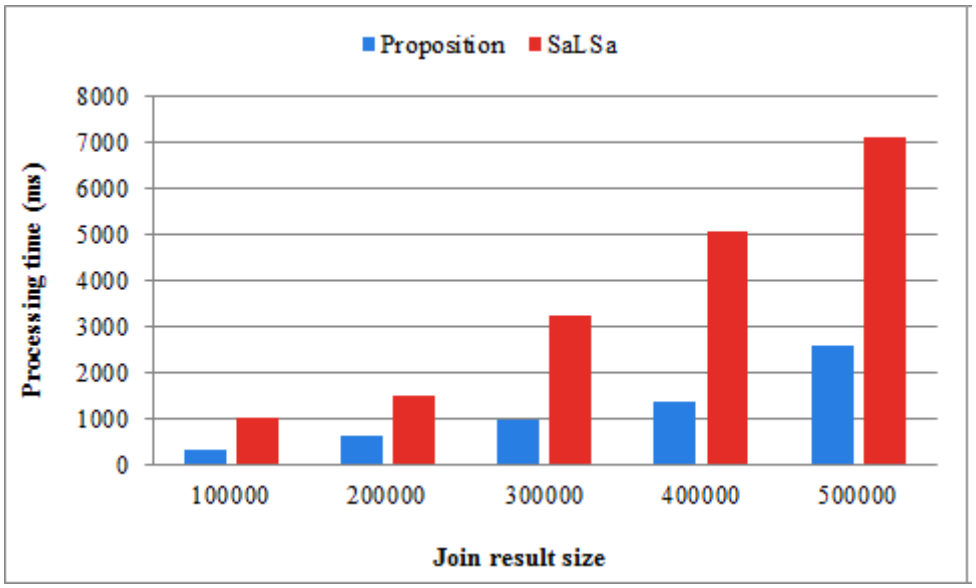


Figure V. 9. Comparaison des performances du modèle proposé avec l'algorithme itératif.

En effet, on pourra conclure que l'utilisation de Skyline locale et de Skyline globale proposé permet d'améliorer les performances du système de sélection de services de réservation. Ainsi, la bonne Skyline utilisées dans le processus de recherche et de sélection des meilleurs éléments est l'une des raisons pour laquelle l'approche de décision multi critère est mieux que les autres approches.

5.4.2. Comparaison qualitative

Pour montrer l'originalité et la bonne performance de notre travail par rapport les travaux similaires, on a mené d'une étude comparative qui va porter sur trois critères de comparaison : le type de requête Skyline, le type de données et l'architecture IOT dans le contexte de recherche et sélection des meilleurs éléments.

Dans cette étude comparative, on a également sélectionné un ensemble des travaux similaires de domaine de décision multi critère à base de requête Skyline dans le triennal 2001-2020. On présente dans la table suivante le résultat de comparaison qualitative.

Table V. 2. Comparaison qualitative de notre travail avec d'autres travaux.

Critères / Works	Multi criteria decision		
	Skyline query type	**Data type**	**IOT architecture**
Notre travail	Join dynamic Skyline query	Distributed complex data	+
Borzsony (2001)	Operator Skyline	Relational database	-
Zaman (2016)	Spatial Skyline query	Maps data (area)	-
Kalyvas (2017)	Temporal Skyline query	Temporal Databases	-
Zou (2010)	Dynamic Skyline query	Graph properties	-
Vlachou (2011)	Skyline join algorithm	Relational database	-
Sun (2008)	Join Skyline query	Distributed data	-
Raghavan (2010)	Join Skyline query	Distributed data	-
Jin (2007)	Join Skyline query	Multi-relational databases	-
Kertiou (2018)	Dynamic Skyline query	Distributed data	+
Zhang (2016)	Skyline join algorithm	Multiple relations	-
Amiruzzaman (2020)	Multi-dimensional Skyline query	Single data set	-

Sur la base des résultats de comparaison qualitative présenté dans la table ci-dessus, on a remarqué qu'uniquement notre travail qui permet de traiter le problème d'hétérogénéité de complexe données qui distribue dans des bases de données différentes (relationnelles, orientées-objet ...) par l'utilisation les options des systèmes IoT qui sont naturellement distribués où résoudre le problème d'hétérogénéité, tandis que d'autres travaux ne traitent que les bases de données homogènes (comme sur le tableau V.2).

De plus, les études précédentes sur Skyline se limitaient à relation unique ou multi-relations dans des environnements distribués ou non distribués. Mais, notre présent travail se concentre sur l'exploitation de la nature distribuée de l'IoT pour faciliter l'utilisation de la jointure dynamique

Skyline locale au niveau des passerelles, puis l'utilisation de jointure de dynamique Skyline globale au niveau du serveur de système et on a prouvé l'efficacité de notre proposition par un exemple typique (réservation des services de voyage). L'objectif principal est de rechercher et de sélectionner les meilleurs éléments disponibles en tenant compte des demandes des utilisateurs dans le contexte de décisions multicritères.

6. Synthèse des résultats des expérimentations

Pour montrer la fiabilité et l'originalité de notre approche par rapport aux autres approches existantes, on a mené un ensemble des expérimentations pour le but d'évaluer les performances du système en termes de qualité de données intégrées, et une expérimentation pour faire une étude comparative.

Sur la base des résultats qu'on a obtenus dans la première expérimentation, on a conclu que notre système proposé à la capacité d'intégrer toutes les données hétérogènes dans l'environnement distribue. Ainsi, notre système est très évolutif dans le sens où il pourra être facilement enrichi par une nouvelle base de données et peut être utilisé dans de nombreux domaines spécialement le domaine de business intelligence.

Par ailleurs, dans la deuxième expérimentation, on a mené deux études comparatives. Une première étude comparative dite quantitative qui permet de comparer les Skyline utilisées dans notre approche avec d'autres Skylines existantes. Selon les résultats de comparaison obtenus, les Skyline proposé ont été bien choisi en termes de valeur de temps d'exécution de requête. En outre, utiliser les options de l'IOT comme une environnement distribue joue un rôle très important pour la agrégation des éléments appliquer en même domaine qui fonctionne séparément dans la réalité ainsi que pour résoudre l'hétérogénéité entre les data sets.

Une deuxième étude comparative dite qualitative qui vise à comparer notre travail avec d'autres travaux similaires afin de montrer l'originalité et de dériver ses avantages par rapport les autres travaux. Les résultats ont montré l'efficacité de notre approche pour le traitement du problème d'hétérogénéité et complexité des données dans les bases de données distribues et hétérogènes.

L'exécution de requête Skyline a besoin des grandes bases de données, pour cela on a généré trois grandes bases de données et on les distribuer sur trois serveurs (serveurs locaux) dans

un réseau privé. L'interrogation de l'utilisateur est au niveau de serveur de système où diffusée la demande dans le réseau. L'approche proposée consiste à appliquer un processus de sélection des meilleurs éléments dans un espace de recherche. Ce processus est défini par quatre phases: diffusée la demande, sélection locales au niveau de serveurs de data sets, agrégation des premiers résultats et sélection de meilleur réponse. De plus, notre approche est valide pour tous les domaines de business intelligence.

Ces résultats d'expérimentations sont très satisfaisants et ils ont montré l'efficacité de notre approche pour atteindre notre objectif concernant le traitement du problème d'hétérogénéité de complexe données pour l'exploration des sources de données distribues et hétérogènes.

7. Conclusion

Ce chapitre a été consacré à la présentation de deuxième contribution décrite au cours de cette thèse. Cette contribution donne une nouvelle approche à base de requête Skyline pour la sélection des meilleurs éléments dans un environnement de recherche distribue et hétérogène par utilisé les options de l'IOT. On a proposé un modèle de recherche et de sélection divisé au deux opérations principales ; la première sélection les meilleurs éléments au niveau des serveurs locaux et le deuxième permet de sélection le meilleur élément au niveau de serveur de système.

Le résultat du processus de sélection et de recherche est donc un ensemble des meilleurs éléments en tenant compte des demandes des utilisateurs. Ce processus est donc avantageux dans le contexte de décision multi critère.

Dans le reste de ce chapitre, on a mené une étude expérimentale en comparant notre processus de sélection des meilleurs éléments à base de requête jointure de dynamique Skyline avec une Skyline itérative. Nos résultats sont très encourageants en termes de qualité des éléments trouvés et le temps écoulé pour sélectionner les éléments appliqué (services de réservation).

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Les travaux présentés dans cette thèse se situent dans le contexte du traitement du problème de calcul et de stockage distribués de structures de données complexes, utiles et dynamiques pour la fouille de données qui sont souvent stockées dans des nombreuses sources hétérogènes, conçues indépendamment les unes des autres. L'accès à ces données entraîne des difficultés à accéder aux besoins d'utilisateurs qui peuvent rendre difficile la restitution des réponses pertinentes. Il est nécessaire d'offrir un accès unifié permettant d'une part, d'explorer aisément ces sources de données et d'autre part de traiter les problèmes de calcul et de stockage distribués de structures de données complexes au niveau des requêtes et des sources de données.

Dans ce cadre, on a proposé deux approches distinctes : la première approche d'analyse de très grands jeux de données, dans un environnement distribué, en minimisant le gaspillage de ressources des réseaux et en maximisant l'interaction avec un utilisateur et la deuxième approche de sélectionner les meilleurs éléments pour les utilisateurs par des requêtes plus complexes multicritères. Par la suite, on a présenté dans un sommaire les deux contributions puis on a proposé les principales perspectives et orientations pour les travaux futurs.

1. Sommaire des contributions

On a rappelé que ce travail doctoral s'inscrit dans le cadre de faciliter la sélection des meilleurs éléments des sources de données tout en traitant le problème de calcul et de stockage distribués de structures de données complexes au niveau de la requête d'utilisateur et les sources de données hétérogènes et distribuées. On a présenté un état de l'art qui compose les trois principaux aspects : aspect de bases de données distribuées où on l'a expliquée la structure et l'intégration des bases de données distribuées, un aspect explique l'extraction des Motifs fréquents, et un dernier aspect pour les requêtes Skyline qui aide à la décision multicritère.

Dans le premier aspect, on a la structure d'une base de données distribuée et l'architecture des SGBD distribués en plus de l'intégration des bases de données. De plus, on a également abordé le traitement de la requête distribuée et les placements d'optimisations. De ce fait, on

distingue deux placements d'optimisations, optimisation globale qui consiste à trouver le meilleur ordre des opérateurs dans la requête, y compris les opérateurs de communication, qui minimisent une fonction de coût. Et optimisation locale où la sortie de la couche d'optimisation de requête est une requête algébrique optimisée avec des opérateurs de communication incluse sur des fragments. Ces informations nous ont aidés à proposer une nouvelle architecture pour les bases de données distribuées nous permet d'appliquer des améliorations de requête distribuée au niveau global et au niveau local.

Dans le deuxième aspect, on a déterminé les tâches et les outils de fouille de données et on distingue le datamining distribué où on a présenté les algorithmes d'extraction des items sets fréquents et des items sets fermés fréquents et on choisit pour l'analyse le comportement d'utilisateur dans le système proposé.

Dans le troisième aspect, les requêtes Skyline joue un rôle crucial dans les problèmes de décisions multicritères. On a utilisé le Skyline comme un outil d'optimisation pour sélectionner les meilleurs éléments demandés par l'utilisateur, ou on a proposé la requête join dynamique Skyline en architecture du système hétérogène, distribué et complexe.

On présente dans ce qui suit nos principales contributions pour le traitement du problème de calcul et de stockage distribués de structures de données complexes, utiles et dynamiques pour la fouille de données qui sont souvent stockées dans des nombreuses sources hétérogènes, conçues indépendamment les unes des autres et comment sélectionner les meilleures parties de ces données pour la demande de l'utilisateur.

1.1. Traitement d'hétérogénéité de données dans un environnement distribué

L'approche proposée vise à manipuler et à analyser des données rapides avec minimisation le gaspillage de ressources des réseaux et en maximisant l'interaction avec un utilisateur où on a proposé un système multi agent qui compose cinq agents, tel que chaque agent gère un composant du système ce qui assure la division des tâches du système entre les agents. Dans le cadre d'analyse des données la proposition de la base de résultats permet de minimiser la quantité des données dans le moteur de recherche des règles d'associations à la méthode de Pasquier qui divise les générateurs des règles aux deux bases : base générique pour les règles d'association exactes et base informative pour les règles d'association approximatives. L'avantage de cette proposition est d'accélérer le processus d'analyse et le rend plus précis.

Le processus d'appariement proposé prend en entrée à la fois la requête initiale soumise par l'utilisateur, et donne en sortie l'ensemble des meilleurs éléments répondant à cette requête avec le rapport d'analyse. Ce processus se déroule en trois phases successives : collecte et transformation de données au niveau de station de travail, processus de réservation des clients au niveau de gestionnaire des tâches et génération des rapports au niveau de base de résultats. Ces phases sont basées sur l'utilisation de systèmes des multi agents. De plus, les résultats obtenus dans chaque étape ont été transmis à la phase suivante, mais le résultat de deuxième étape passe par l'utilisateur pour confirmer son choix et devient alors une entrée dans la dernière étape où les choix des utilisateurs sont analysés.

1.2. Approche de décision multi critère basé sur la requête Skyline

La deuxième contribution présentée dans cette thèse s'inscrit dans le cadre du traitement du problème sélection des meilleurs éléments pour la décision multicritère dans un environnement hétérogène et distribué par l'utilisation des options de l'IOT. L'approche proposée concerne la requête Skyline qui représente une requête plus complexe multicritère. On a pris la réservation des services de voyages comme un cas d'utilisation pour faciliter l'explication de requête Skyline.

De plus, l'approche de recherche et de sélection proposée est divisée aux deux opérations principales : la première sélection les meilleurs éléments au niveau des serveurs locaux et la deuxième opération permet de sélection le meilleur élément au niveau du serveur du système.

De plus, le modèle proposé pour la recherche et la sélection des meilleurs éléments dans un environnement complexe est applicable pour différents types des systèmes (facteurs de risque pour la santé, le marketing, la recherche d'informations pour le grand public, etc.). Sur le plan pratique, on a développé le système de recherche et de sélection par l'utilisation des options de l'IOT basé sur l'architecture en trois couches (serveur global, serveurs locaux et sources de données). De plus, on a mené deux grandes expérimentations :

- Evaluation des performances qui s'appuie sur le calcul de précision et sur l'accélération de traitement des données d'une part et d'autre part le traitement de requête pour évaluer les performances du système en matière de qualité de réponses retournées. Les résultats montrent que notre système proposé à la capacité d'intégrer toutes les données hétérogènes dans l'environnement distribue.

- Comparaison qui se base sur deux études comparatives : une comparaison quantitative des mesures de similarité utilisées par rapport des autres mesures et une comparaison qualitative qui vise à montrer l'originalité et l'efficacité de notre approche par rapport aux travaux similaires. Sur la base des résultats qu'on a obtenus, on peut dire que le Skyline proposé a été bien choisi en matière de valeur de temps d'exécution de requête. En outre, l'utilisation des options de l'IOT comme un environnement distribué joue un rôle très important pour l'agrégation des éléments appliquée au même domaine qui fonctionne séparément dans la réalité ainsi que pour résoudre l'hétérogénéité entre les data sets.

Finalement, notre approche est très évolutive dans le sens où elle pourrait être facilement enrichie par une nouvelle base de données et peut-être utilisée dans de nombreux domaines spécialement le domaine de business intelligent.

2. Perspectives

Outre les contributions présentées dans cette thèse, ce travail ouvre la voie vers diverses perspectives permettant des améliorations dans les deux approches proposées :

Dans l'approche de traitement d'hétérogénéité de données dans un environnement distribué, parmi les perspectives les plus prometteuses sont de prévoir une restructuration de cube de données sur les entrepôts de données pour rendre les recherches d'informations et les processus de sélection plus précis et plus rapides. De plus, comme une autre perspective importante, on souhaite étudier les performances de nos propositions pour un autre système de business intelligence par exemple le marketing est découvert des tendances.

Dans l'approche de décision multicritère basée sur la requête Skyline, on prévoit de générer des grandes bases de données dans une grille pour confirmer la performance de cette proposition. Aussi, on travaillera à proposer une méthode sémantique pour explorer au mieux les bases de données.

BIBLIOGRAPHIE

[Agrawal, 96] Agrawal, R., & Shafer, J. C. “Parallel mining of association rules”, IEEE Transactions on knowledge and Data Engineering, 8(6), 962-969. (1996).

[Agard, 05] Agard, B., & Kusiak, A. “Exploration des bases de données industrielles à l’aide du datamining–Perspectives”, In 9ème colloque national AIP PRIMECA, (2005, April).

[Agrawal, 94] Agrawal, R., & Srikant, R. “Fast algorithms for mining association rules”, In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499). September 1994.

[Alramouni, 06] Alramouni, S., & Lee, J. Y. (2006). DARCI: Distributed Association Rule Mining Utilizing Closed Itemsets. CSIT 2006 computer science and information technology.

[Amiruzzaman, 20] Amiruzzaman, M., & Jamonnak, S. “Multi-dimensional Skyline query to find best shopping mall for customers”, In 2020 6th Conference on Data Science and Machine Learning Applications (CDMA) (pp. 71-76). IEEE, (2020, March).

[Bartolini, 06] Bartolini, I., Ciaccia, P., & Patella, M. “SaLSa: Computing the Skyline without scanning the whole sky”, In Proceedings of the 15th ACM international conference on Information and knowledge management (pp. 405-414), (2006, November).

[Bemile, 14] Bemile, R., Achampong, A., & Danquah, E. “Online hotel reservation system”, International Journal of Innovative Science, Engineering & Technology, 1(9), 583-588, (2014).

[Berry, 04] Berry, M. J., & Linoff, G. S. “Data mining techniques: for marketing, sales, and customer relationship management”, John Wiley & Sons, 615 pages, 2 ème edition 2004.

[Bhagat, 18] Bhagat, S. S., Bagul, A. D., Patil, P. N., & Dahale, S. A. “Perceptive car parking booking system with IOT technology”, International Research Journal of Engineering and Technology, 5(2), 1123-1125, (2018).

[Borzsony, 01] Borzsony, S., Kossmann, D., & Stocker, K. “ The skyline operator”, In Proceedings 17th international conference on data engineering (pp. 421-430). IEEE. (2001, April).

[Bouberka,22] Bouberka, S., & Rabah, B. “L’impact du système d’information dans l’optimisation du processus décisionnel Cas: La Direction Opérationnelle d’Algérie Télécom de Tizi-Ouzou. ”, Diss. Université Mouloud Mammeri, 2022.

[Chaabane, 13] Chaabane, L. “Fusion et fouille de données guidées par les Connaissances: application à l’analyse d’image ”, Doctorat, Universite Mohamedkhider–biskra. (2013).

[Chevance,01] Chevance, R. J. “Bases de données réparties et fédérées”, (2001).

[Cheung, 96] Cheung, D. W., Ng, V. T., Fu, A. W., & Fu, Y. “ Efficient mining of association rules in distributed databases”, IEEE transactions on Knowledge and Data Engineering, 8(6), 911-922. (1996).

[Cheung, 96] Cheung, D. W., Han, J., Ng, V. T., Fu, A. W., & Fu, Y. “ A fast distributed algorithm for mining association rules”, In Fourth International Conference on Parallel and Distributed Information Systems (pp. 31-42). IEEE. (1996, December).

[Chomicki, 03] Chomicki, J., Godfrey, P., Gryz, J., & Liang, D. “Skyline with presorting”, In ICDE (Vol. 3, pp. 717-719), (2003, March).

[Chomicki, 05] Chomicki, J., Godfrey, P., Gryz, J., & Liang, D. “Skyline with presorting: Theory and optimizations”, In Intelligent Information Processing and Web Mining (pp. 595-604), Springer, Berlin, Heidelberg, (2005).

[Date, 71] Date, C. J., & Hopewell, P. “File definition and logical data independence”, In Proceedings of the 1971 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control, pp. 117-138, (1971, November).

[Devogele, 95] Devogele, T., Parent, C., & Spaccapietra, S. “On spatial database integration”. International Journal of Geographical Information Science, 12(4), 335-352, (1998).

[Djeffal, 14] Djeffal, A. Cours Fouille de données avancée. (2014).

[DU, 89] Du, W., & Elmagarmid, A. “Quasi serializability: a correctness criterion for global concurrency control in InterBase”, In Proceedings of the 15th International Conf. on Very Large Data Bases. (1989, August).

[Fatiha, 11] Fatiha, B. “Data Mining Distribue”, Doctoral dissertation, Universite mohamed boudiaf'des sciences et de la technologie d'oran. (2011).

[Fayyad, 96] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. “From data mining to knowledge discovery in databases”, AI magazine, 17(3), 37-37, (1996).

[Fayyad, 01] Fayyad, U. “Knowledge discovery in databases: An overview”, Relational data mining, 28-47. (2001).

[GARCIA-MOLINA, 82] Garcia-Molina, H., & Wiederhold, G. “Read-only transactions in a distributed database”, ACM Transactions on Database Systems (TODS), 7(2), 209-234. (1982).

[Gawande, 12] Gawande A. D., Bhuyar P. R., and Deshmukh A. B., “Horizontal Fragmentation Technique in Distributed Database”, International Journal of Scientific and Research Publications, 2(5), 2012.

[Godfrey, 05] Godfrey, P., Shipley, R., & Gryz, J. “Maximal vector computation in large data sets” ,In VLDB (Vol. 5, pp. 229-240), (2005, August).

[Grama, 03] Grama, A., Kumar, V., Gupta, A., & Karypis, G. “Introduction to parallel computing”, Pearson Education. (2003).

[Gray, 97] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., ... & Pirahesh, H. “ Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals”, Data mining and knowledge discovery, 1(1), 29-53, (1997).

[Grech, 09] Grech, J. Designing and Implementing a Distributed Database for a Small Multi-Outlet Business. (2009).

[guezouli, 22] http://www.larbiguezouli.com/Fichiers/Enseignement/MasterIAM/bddist/Bases_de_donnees_distribuees.pdf.10/04/2022.

[Gutiérrez, 21] Gutiérrez, C., & Sequeda, J. F. “Knowledge graphs”, Communications of the ACM, 64(3), 96-104. (2021).

[Han, 00] Han, J., Pei, J., & Yin, Y. “Mining frequent patterns without candidate generation”, ACM sigmod record, 29(2), 1-12. (2000).

[Jin, 06] Jin, W., Ester, M., Hu, Z., & Han, J. “The multi-relational Skyline operator”, In 2007 IEEE 23rd International Conference on Data Engineering (pp. 1276-1280). IEEE. (2006, April).

[Jin, 04] Jin, W., Han, J., & Ester, M. “Mining thick Skyline over large databases”, In European Conference on Principles of Data Mining and Knowledge Discovery (pp. 255-266). Springer, Berlin, Heidelberg. (2004, September).

[Jin, 06] Jin, W., Ester, M., Hu, Z., & Han, J. “The multi-relational Skyline operator”, In 2007 IEEE 23rd International Conference on Data Engineering (pp. 1276-1280). IEEE, (April, 2006).

[Kahn, 15] M. G. Kahn, J. S. Brown, A. T. Chun, B. N. Davidson, D. Meeker, P. B. Ryan, ... & M. N. Zozus, “Transparent reporting of data quality in distributed data networks”. Egems, 3(1), 2015.

[Kalyvas, 17] Kalyvas, C., Tzouramanis, T. “A survey of skyline query processing”, arXiv preprint arXiv: 1704.01788, (2017).

[Kaundal, 14] Kaundal, G., Kaur, S., & Vashisht, S., “Review on Fragmentation in Distributed Database Environment”. IOSR Journal of Engineering, 4(3), 28-32, (2014).

[Kertiou, 18] Kertiou, I., Benharzallah, S., Kahloul, L., Beggas, M., Euler, R., Laouid, A., & Bounceur, A. “A dynamic Skyline technique for a context-aware selection of the best sensors in an IoT architecture”, Ad Hoc Networks, 81, 183-196, (2018).

[Kossmann, 02] Kossmann, D., Ramsak, F., & Rost, S. “Shooting stars in the sky: An online algorithm for skyline queries”, In VLDB'02: Proceedings of the 28th International Conference on Very Large Databases (pp. 275-286). Morgan Kaufmann, (2002, January).

[Lefevre, 13] Lefevre, L. “Contributions à la flexibilité et à l'efficacité énergétique des systèmes distribués à grande échelle (Doctoral dissertation, Ecole normale supérieure de lyon-ENS LYON), (2013).

[Li, 08] Li, H., Wang, Y., Zhang, D., Zhang, M., & Chang, E. Y. “Pfp: parallel fp-growth for query recommendation”, In Proceedings of the 2008 ACM conference on Recommender systems (pp. 107-114). (2008, October).

[Liu, 07] Liu, C., Zheng, Z., Cai, K. Y., & Zhang, S. “ Distributed frequent closed itemsets mining”, In 2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System (pp. 43-50). IEEE. (2007, December).

[Maamra, 18] Maamra, O. E., & Kholladi, M. K. “Intelligent Reservation Systems Based on MAS & Data Mining Method”, In International Conference on Advanced Intelligent Systems for Sustainable Development (pp. 1-12), Springer, Cham. (2018, July).

[Maamra, 22] Maamra, O. E., Kholladi, M. K., Kazar, O., & Harous, S. "Smart-Approach Based Internet of Things and Skyline Query for Multicriteria Decisions for Travel Services", Ingénierie des Systèmes d'Information, 27(1), (2022).

[McTavish, 10] McTavish, C., & Sankaranarayanan, S. "Intelligent agent based hotel search & booking syst", m. In 2010 IEEE International Conference on Electro/Information Technology (pp. 1-6). IEEE, (2010, May).

[Merzoug, 18] Merzoug, S. "Une approche basée agent pour l'optimisation d'allocation des ressources dans le green cloud", (Doctoral dissertation, University of Eloued جامعة الوادي) ,(2018).

[Mokeddem, 17] Mokeddem, D. " Datamining Distribué: Contribution à l'Amélioration des Performances ", Doctoral dissertation, Université Mohamed Boudiaf des Sciences et de la Technologie-Mohamed Boudiaf d'Oran. (2017)

[Navathe, 95] Navathe, S., Karlapalem, K., & Ra, M. "A mixed fragmentation methodology for initial distributed database design", Journal of Computer and Software Engineering, 3(4), 395-426. (1995).

[Niswonger, 09] Niswonger .B, Haas. L. M, & Miller. R. J., "Transforming Heterogeneous Data with Database Middleware beyond Integration". (2009).

[Ogirima, 14] Ogirima, S. A. O., Awode, T. R., & Adeosun, O. O. "Online computerized hotel management system", J. Comput. Biosci. Eng., ISSN, 2348-7321, (2014).

[Oloyede, 14] Oloyede, M. O., Alaya, S. M., & Adewole, K. S. "Development of an online bus ticket reservation system for a transportation service in Nigeria", Development, 5(12), 40-2, (2014).

[Orlunwo placida, 21] Orlunwo Placida, O., & Prince Oghenekaro, A. (2021). Distributed Database Management System (DBMS) Architectures And Distributed Data Independence. January 2021.

[ÖZSU, 96] Özsu, M. T., & Valduriez, P. "Distributed and parallel database systems", ACM Computing Surveys (CSUR), 28(1), 125-128. (1996).

[ÖZSU, 99] Özsu, M. T., & Valduriez, P. "Principles of distributed database systems" (Vol. 2), Englewood Cliffs: Prentice Hall. (1999).

[Özsu, 11] Özsu. M. T and Patrick.V. "Principles of Distributed Database Systems", Third Edition, Springer, 2011.

[Papadias, 03] Papadias, D., Tao, Y., Fu, G., & Seeger, B. “An optimal and progressive algorithm for Skyline queries”, In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (pp. 467-478), (2003, June).

[Papadias, 05] Papadias, D., Tao, Y., Fu, G., & Seeger, B. “Progressive Skyline computation in database systems”, ACM Transactions on Database Systems (TODS), 30(1), 41-82, (2005).

[Park, 02] Park, B. H., & Kargupta, H. “Distributed data mining: Algorithms, systems, and applications”, (2002).

[Pasquier, 00**]** Pasquier, N. “Extraction de Bases pour les Règles d'Association à partir des Itemsets Fermés Fréquents”, In Inforsid'2000 Congress (pp. 56-77), (2000, May).

[Pasquier_2, 00**]** Pasquier, N. “Data mining: algorithmes d'extraction et de réduction des règles d'association dans les bases de données”, (Doctoral dissertation, Université Blaise Pascal-Clermont-Ferrand II), (2000).

[Raghavan, 10] Raghavan, V., & Rundensteiner, E. A. “ProgXe: progressive result generation framework for multi-criteria decision support queries”, In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (pp. 1135-1138), (2010, June).

[Raïssi, 10] Raïssi, C., Pei, J. et Kister, T. “ Computing closed skycubes ”, PVLDB, 3(1):838–847. 56, (2010).

[Sangeetha,12] Sangeetha, K. V., & Rajeshwari, P. “Data mining and warehousing”, Journal of Computer Applications, 5(1), 35-43, (2012).

[Site, 22] Site : “Global Distribution System (GDS) : quels avantages pour les hôtels ? ”, sur la page, https://www.revfine.com/fr/systeme-de-distribution-global/ consulté le 18/08/2022.

[Site_1, 20] Site : https://pdfslide.net/documents/chapter-14-distributed-systems-computer-science-computer.html consulté le 10/02/2020

[Sharifzadeh, 06] Sharifzadeh, M., & Shahabi, C. “The spatial Skyline queries”, In Proceedings of the 32nd international conference on Very large data bases (pp. 751-762), (2006, September).

[Sharifzadeh, 09] Sharifzadeh, M., Shahabi, C., & Kazemi, L. “Processing spatial Skyline queries in both vector spaces and spatial network databases”, ACM Transactions on Database Systems (TODS), 34(3), 1-45, (2009).

[Stephan,01] Stephan, B., Donald, K., & Konrad, S. “The Skyline operator”, In Proc. ICDE (Vol. 1), (April, 2001).

[Sun, 08] Sun, D., Wu, S., Li, J., & Tung, A. K. “Skyline-join in distributed databases”, In 2008 IEEE 24th International Conference on Data Engineering Workshop (pp. 176-181). IEEE, (April, 2008).

[Tan, 01] Tan, K. L., Eng, P. K., & Ooi, B. C. “Efficient progressive skyline computation”, In VLDB (Vol. 1, pp. 301-310). (2001, September).

[Tanasescu, 04] Tanasescu, A., Boussaid, O., & Bentayeb, F. “Towards Complex Data Warehousing: A new approach for integrating and modeling complex data”, In 5th International Conference on Modeling, Computation and Optimization in Information Systems and Management Sciences, (2004, July).

[Ugarte Rojas, 14] Ugarte Rojas, W. “Extraction de motifs sous contraintes souples”,Doctoral dissertation, Caen, (2014).

[Vangenot, 03] Vangenot, C. “Datawarehouse. Technical report, Ecole polytechnique fédérale de Lausanne”. (2003).

[Vlachou, 06] Vlachou, A., Doulkeridis, C., Kotidis, Y., & Vazirgiannis, M. “Skypeer: Efficient subspace Skyline computation over distributed data”, In 2007 IEEE 23rd International Conference on Data Engineering (pp. 416-425). IEEE, (April, 2006).

[Wiederhold, 92] Wiederhold, G. “Mediators in the architecture of future information systems”, Computer, 25(3), 38-49. (1992).

[Wikipedia, 22] Wikipedia : "Système de réservation informatique", sur la page, https://fr.wikipedia.org/wiki/Système_de_réservation_informatique , consultée le 22 Décembre 2022.

[Wrembel, 06] Wrembel, R. (Ed.). “Data Warehouses and OLAP: Concepts, Architectures and Solutions: Concepts, Architectures and Solutions”, Igi Global. (2006).

[Xia, 08] Xia, T., Zhang, D., & Tao, Y. “On skylining with flexible dominance relation”, In 2008 IEEE 24th International Conference on Data Engineering (pp. 1397-1399). IEEE, (2008, April).

[Yuan, 05] Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J. X., & Zhang, Q. “ Efficient computation of the Skyline cube”, In Proceedings of the 31st international conference on Very large data bases (pp. 241-252), (2005, August).

[Zaman, 16] Zaman, A., & Morimoto, Y. “Area Skyline query for selecting good locations in a map”, Journal of Information Processing, 24(6), 946-955, (2016).

[Zighed, 02] Zighed, D. A., & Rakotomalala, R. “Extraction de connaissances à partir de données (ECD). Techniques de l'Ingénieur”, H, 3, 744. (2002).

[Zhang, 16] Zhang, J., Lin, Z., Li, B., Wang, W., & Meng, D. “Efficient Skyline query over multiple relations”, Procedia Computer Science, 80, 2211-2215, (2016).

[Zou, 10] Zou, L., Chen, L., Özsu, M. T., & Zhao, D. “Dynamic Skyline queries in large graphs”, In International Conference on Database Systems for Advanced Applications (pp. 62-78). Springer, Berlin, Heidelberg, (2010, April).