Université Amar Telidji Laghouat

Faculté des Sciences
Département De Mathématiques et d'informatique
Filière : Informatique
Option :Système d'information et de décision

Contribution des systèmes multi-agents dans la recherche de l'information dans les bases de données distribuées

Réaliser par : Boucherit Malika et Boulefa Hasna

2014/2015

27 mai 2015

Table des matières

Table des figures

Acronymes

SMA Système Multi-Agent.

SGBD Système de Gestion de Base de Données.

MAS-CommonKADS Multi-Agent System Knowledge Analysis and Development System.

MaSE Multiagent System Engineering.

UML unified modeling language.

O-MaSE (Organization based Multiagent System Engineering.

JADE Java Agent Development Framework.

FIPA Foundation for Intelligent Physical Agents.

BDD Base de Données Distribuée.

SGBDD Système de gestion des bases de données distribuée.

SGBDR système de gestion de base de données relationnelles.

ACL Agent Communication Language.

RMA Remote Agent Management.

AMS Agent Management System.

DF Directory Facilitator.

JDBC Java Database Connectivity.

IPMS Inter-Platform Mobility Service.

AMM Agent Mobility Manager.

AMM Agent Mobility Manager.

Dédicace

Je tiens d'abord à remercier le bon Dieu, le tout puissant de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout.

Je dédie ce modeste travail à ma mère et mon père pour leurs soutiens et leurs affections.

A ma sœur Imene et a mes adorables petites nièces Sofia et Sarah.

A mon frère Omar.

A toute ma Famille.

A mon binôme Hasna et à toute la famille Boulefa.

A mes amies d'enfance Meriem, Aicha, Keltoum, Sana, Isra, Manel...Pour leur soutien morale.

BOUCHERIT Malika

Dédicace

Je dédie ce mémoire à :

Mes parents :

Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude. Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi.

Mes frères Abdenour et Idris, mes sœurs Mina et Nesrine vous m'avez soutenue et encouragé durant toutes ces années ; vous étiez toujours présents quand j'avais besoin de vous, je n'aurais pu achever ce travail sans votre générosité et votre affection ; Que le tout puissant soit à vos côtés et vous protège.

Mes professeurs de l'UATL qui doivent voir dans ce travail la fierté d'un savoir bien acquis.

BOULEFA Hasna

Remerciement

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui on voudrait témoigner toute notre reconnaissance.
On voudrait tout d'abord adresser toute notre gratitude à la directrice de ce mémoire, Kerrouche Badra, pour sa patience, sa disponibilité et ses conseils, qui ont contribué à alimenter notre réflexion.
Nous remercions Messieurs les membres du jury de nous faire l'honneur de juger notre modeste travail.
Nous remercions tous nos professeurs du département d'Informatique.
Remerciement les plus sincères à toutes les personnes qui nous ont aidé de Près ou de loin à accomplir notre travail.

Résumé

Avec l'accroissement du besoin en information, la recherche devient incontournable. Une demande d'information peut être répartie sur plusieurs sites d'où la nécessité d'une reconstitution de la réponse par une entité gérante. D'autre part, les échanges des données et la répartition des requêtes d'une application distribuée nécessitent l'interaction entre différentes entités à travers le réseau. Dans ce mémoire, nous proposons une approche qui utilise les agents. Ces derniers apparaissent dans ce contexte comme une solution prometteuse facilitant la mise en œuvre d'applications réparties. Nous décrivons notre architecture à base d'agents pour la recherche d'information dans les bases de données distribuée et nous présentons une évaluation de temps d'exécution des requêtes SQL du modèle agents par comparaison au SGBD et ceci à partir d'une application simple et d'une base de données distribuée entre deux sites. Cette évaluation a été effectuée dans l'environnement Java en utilisant la plateforme JADE à travers différents scénarios d'exécution des requêtes.
MOTS-CLÉS : Système multi-agents, Base de données Distribuée, SGBD distribuée.

Abstract

Première partie

Introduction Générale

Chapitre 1

Introduction

1.1 Contexte et objectifs de la thèse

L'intérêt pour la recherche d'information dans des sources distribuées a conduit à considérer non seulement la distribution des données, mais aussi la possibilité d'intégrer les agents que l'on peut qualifier d'autonomes. Dans ces systèmes, le nombre de requêtes à traiter est trop important pour qu'il soit possible de solliciter toutes les sources de données. Outre que cela créerait une charge réseau trop importante, les sources elles-mêmes ne sont certainement pas dimensionnées pour recevoir et traiter toutes les requêtes. Cette complexité a amené les chercheurs à utiliser le paradigme agent qui a montré sa capacité à gérer ce type de problèmes grâce à ses caractéristiques telles que l'interaction durant la résolution de problèmes,l'adaptation aux changements de l'environnement, l'autonomie des entités, l'élaboration, la simulation et/ou la compréhension de systèmes coopératifs ou compétitifs, distribuées, et ouverts pouvant intégrer à la fois les agents humains et/ou artificiels. Les agents sont utilisés de plus en plus fréquemment dans le domaine de la recherche de l'information grâce aux capacités et services exhibés tels que :

- Rechercher, acquérir, analyser, intégrer les informations provenant des différentes sources.
- S'adapter à l'évolution et au changement de l'environnement.

Les objectifs de notre travail seront :

- Rechercher de l'information des sites distribuées,
- Analyser les requêtes syntaxiquement,
- Décomposer les requêtes en sous-requêtes,
- réduire le coût de transfert,

– Router les requêtes à l'aide d'un agent mobile qui transportera la requête vers les différents sites ,et renvoyer la réponse vers le sites demandeurs.

1.2 Problématique

Dans un système distribué constitué d'un grand nombre d'utilisateurs et de sources de données, un utilisateur cherche une information particulière sans nécessairement savoir si elle existe ou non. L'utilisateur émet une requête et le système doit chercher la source appropriée à la demande si toutefois elle existe. Et compte tenu de nombreux sites à accéder, le temps de réponse des requêtes peut devenir très élevé. Il est bien évident que les performances d'un système de gestion de bases de données réparties dépendent essentiellement de la capacité de l'optimisation des requêtes pour produire des stratégies efficaces de traitement des requêtes. Le but du traitement des requêtes distribuées est d'exécuter ces requêtes le plus efficacement possible afin de minimiser le temps de réponse aux utilisateurs, et de minimiser le coût de communication total associé à une requête, cette amélioration se fait par le traitement parallèle, le partage des données, et l'augmentation de la capacité de gestion des données.

1.3 Organisation du mémoire

Ce mémoire est structuré en quatre(04)chapitres.
- Le chapitre 1 présente l'introduction general.
- Le chapitre 2 présente l'état de l'art. Nous y définissons les systèmes de gestion des bases de données distribuées et les systèmes multi-agents.
- Le chapitre 3 présente la conception de notre approche, et explique le déroulement de cette étape en passant par les differents diagrammes de la méthodologie MASE et en détaillant les structures des agents.
- Le chapitre 4 présente l'implementation de notre approche.
- et le dernier chapitre 5 présente la conclusion et les perspectives.

Deuxième partie

Etat de l'art

Chapitre 2

Etat de l'art

2.1 Système multi-agent

2.1.1 Introduction

Un effort particulier a été porté ces dernières années sur l'intelligence artificielle distribuées et les systèmes multi-agents, l'objectif de ces derniers est de modéliser des systèmes plus complexes, hétérogènes et évolutifs. Zambonelli et Van Dyke Parunak [2] soulignent que les systèmes logiciels récents ont un très grand degré de complexité. Les agents et les systèmes multi-agents fournissent plusieurs solutions à ces problèmes.

Un système multi-agent est défini comme un macro-système constitué d'agents autonomes qui interagissent dans un environnement commun pour réaliser une activité collective cohérente.Ainsi, de nombreux travaux sur les architectures d'agents ont tenté d'identifier les différentes fonctions et composantes (perception, communication, délibération,...) d'un agent et la façon dont elles sont organisées pour fournir le comportement global d'un agent.le but de la section suivante est de donner une vue globale sur la notion de systèmes multi-agents et ses composantes.

2.1.2 Généralités

2.1.2.1 Qu'est ce qu'un agent ?

Pour définir un agent plusieurs définitions existent, nous optons pour celle de Ferber " Un agent est une entité logicielle ou matérielle, adaptative, rationnelle, autonome, qui est capable de percevoir et d'agir sur elle-même

et sur son environnement et en coopération (communiquer, négocier) avec d'autres agents afin d'atteindre ses objectifs "

2.1.2.2 Propriétés et types d'agents

La liste des différentes caractéristiques d'un agent présentées dans ce paragraphe n'est pas exhaustive. Dans certains cas, les chercheurs attribuent aux agents d'autres caractéristiques.

- Un agent est une entité **autonome** : il agit sans l'intervention des humains ou des autres agents, et il contrôle ses actions en fonction de son état interne et de son environnement.

- Un agent est une entité **proactive** : l'agent a en effet sa propre activité et son propre but. Contrairement aux objets, L'agent n'agit pas simplement en réponses aux messages reçus des autres agents, mais son activité est dirigée par son but.

- Un agent est une entité **adaptative** : L'agent est capable de réguler ses aptitudes en fonction de l'agent avec lequel il interagit et/ou de l'environnement dans lequel il évolue. Autrement dit, un agent adaptatif a la capacité de modifier les propriétés de ses différentes taches afin de satisfaire les demandes internes et externes.

2.1.2.3 Types d'agents

Selon leurs modes de fonctionnement et leurs représentations de leurs environnements, les agents peuvent être classés en trois catégories essentielles qui sont : les agents réactifs, les agents cognitifs et les agents hybrides.

Agents réactifs :
L'approche des agents réactifs est fondée sur l'idée qu'il n'est pas nécessaire que chaque agent d'un système multi-agents soit individuellement « intelligent » pour parvenir à un comportement global intelligent. Cette approche propose la coopération d'agents de faible granularité mais beaucoup plus nombreux. L'agent réactif ne dispose que d'un protocole et d'un langage de communication réduits. Ses capacités répondent uniquement à la loi stimulus/actions : l'occurrence de chaque événement déclenche l'exécution d'une action prédéfinie.

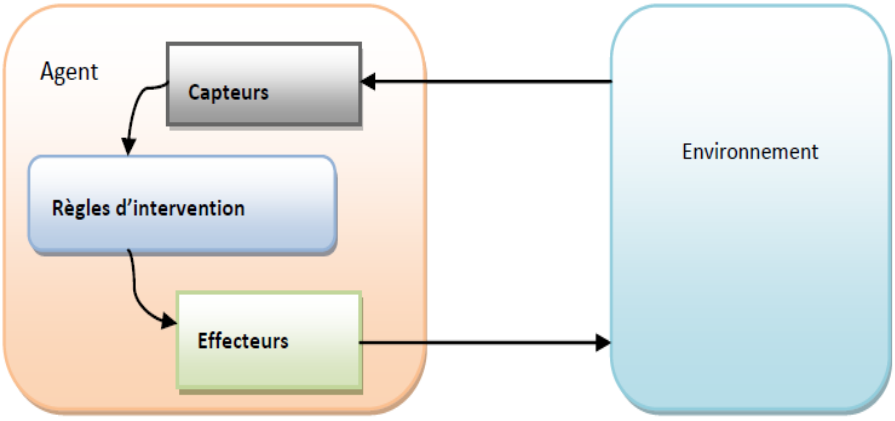


FIGURE 2.1 – Architecture interne de l'agent réactif[1]

Agents cognitifs :
Un système d'agents cognitifs est constitué d'un petit nombre d'agents de forte granularité ; chaque agent est apparenté à un système expert plus ou moins évolué. Les actions de ces agents seront ainsi « réfléchies » en ce sens qu'elles sont basées sur les connaissances de l'agent sur lui- même, sur les autres et sur son environnement et les objectifs qui le guident.

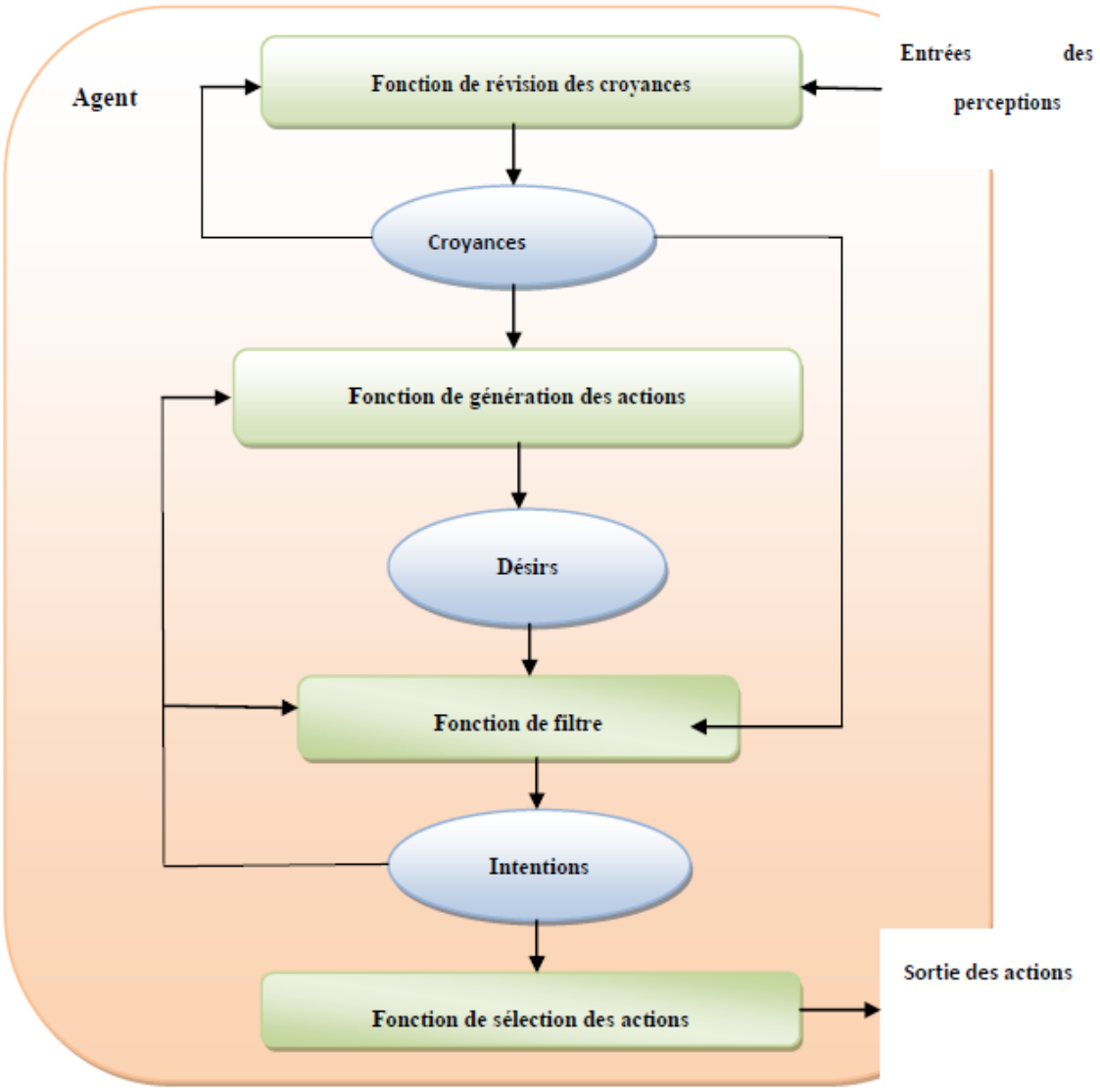


FIGURE 2.2 – Architecture interne de l'agent Cognitif[1]

Agents hybrides :
Dés le début des années 90, on savait que les systèmes réactifs pouvaient bien convenir pour certains types de problèmes et moins bien pour d'autres. De même, pour les agents cognitifs. De là est venu les agents hybrides qu'ils conjuguent la rapidité de réponse des agents réactifs Avec le raisonnement des agents cognitifs .

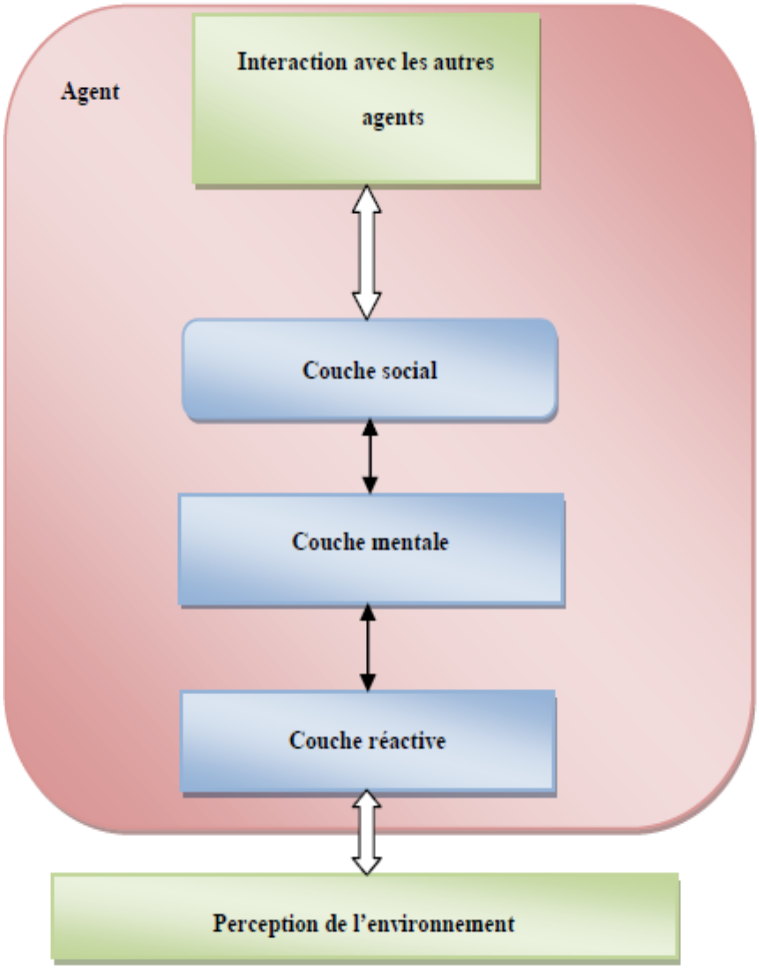


FIGURE 2.3 – Architecture interne de l'agent hybride[1]

Agents mobiles :
Un agent mobile est un agent capable de se déplacer d'un site à un autre. Ce type d'agent s'oppose aux agents statiques (ou stationnaires), qui s'exécutent seulement dans le système où ils ont commencé leur exécution. Par contre, un agent mobile n'est pas lié au système dans lequel il débute son exécution. Il peut transporter son état et son code d'un environnement vers un autre où il poursuit son exécution. Il s'agit donc de déplacer les traitements vers les données plutôt que les données vers les traitements. Un agent mobile peut avoir la même architecture qu'un agent réactif ou cognitif, bien qu'il y ait une nécessité d'avoir un module supplémentaire gérant la mobilité de l'agent.

2.1.3 Définition des Systèmes Multi-Agents

La définition la plus simple d'un système multi-agents est la suivante : Un est un ensemble d'agents interagissant dans un environnement commun. Il est défini d'une façon plus détaillée par Ferber dans [3] comme étant un système composé de :
– un environnement.
– un ensemble d'objets passifs pouvant être perçus, crées, modifiés ou détruits par des agents.
– un ensemble d'agents actifs.
– un ensemble de relation qui relient les objets entre eux.
– un ensemble d'opérations ou de compétences offrant la possibilité aux agents de percevoir, produire, consommer, transformer et manipuler des lois de l'environnement.
– un ensemble de lois qui sont des opérateurs chargés de représenter l'application des action des agents sur le monde et la réaction de ce monde à ces actions et qu'on appellera des lois universelles.
S 'il y a moins de trois agents, on parle plutôt d 'interaction homme/machine, ou machine/machine que de systèmes multi-agents.»

Les Systèmes multi-agents ont des applications dans le domaine de l'intelligence artificielle ou ils permettent de réduire la complexité de la résolution d'un problème en divisant le savoir nécessaire en sous-ensembles, en associant un agent intelligent indépendant à chacun de ces agents.

Ainsi, les agents d'un SMA, n'ayant pas une visibilité globale sur leur environnement, ils ne peuvent avoir qu'un champ d'actions limité sur l'ensemble des objets de cet environnement. De ce fait pour résoudre un problème global, ces agents sont amenés à coopérer et à communiquer pour échanger des informations et pour mieux coordonner leurs actions individuelles et locales.

2.1.4 Problématique des SMA

L'étude d'un système multi-agent est faite sur un ensemble d'axes, nous introduisons la vision donnée par Yves Demazeau appelée la décomposition

Voyelles AEIO (Agent, Environnement, Interactions, Organisation)

2.1.4.1 Agent

Le concept d'agent a été défini plus haut.

2.1.4.2 Environnement

L'environnement est le milieu dans lequel les agents sont introduits. Son rôle est de permettre la communication entre agents, il supporte les actions des agents en définissant les règles et en renforçant ces actions, il est observable par les agents, et prend en charge l'activité propre des objets et des ressources présentes. Lorsque les agents sont réactifs, l'environnement détient une importance capitale car il est le médiateur de leurs interactions. En effet, comme ces agents ne peuvent communiquer directement entre eux, ils s'influencent mutuellement soit par leurs positions s'ils sont situés, soit par l'intermédiaire d'objets qu'ils perçoivent et modifient.

D'aprés Wooldridge [4] Un environnement peut être :

- **Accessible** si un agent peut, à l'aide des primitives de perception, déterminer l'état de l'environnement et ainsi procéder, par exemple, à une action. Si l'environnement est inaccessible alors il faut que l'agent soit doté de moyens de mémorisation afin d'enregistrer les modifications qui sont intervenues.
- **Déterministe**, ou non, selon que l'état futur de l'environnement ne soit, ou non, fixé que par son état courant et les actions de l'agent.
- **Episodique** si le prochain état de l'environnement ne dépend pas des actions réalisées par les agents.
- **Statique** si l'état de l'environnement est stable (ne change pas) pendant que l'agent prend des décisions. Dans le cas contraire, il sera qualifié de dynamique.
- **Discret** si le nombre des actions faisables et des états de l'environnement est fini.

2.1.4.3 Interaction

Une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques [3] Les interactions s'expriment ainsi à partir d'une série d'actions dont les conséquences exercent en retour une influence sur le comportement futur des agents. Les agents interagissent le long d'une suite d'événements pendant lesquels les agents sont d'une certaine manière en contact les uns avec les autres, que ce contact soit

direct ou qu'il s'effectue par l'intermédiaire d'un autre agent ou de l'environnement.
On distingue différents types d'interaction que les agents peuvent adopter :

- **Coordination** : [5]définit la coordination comme "la propriété d'un système composé d'au moins deux agents, exécutant des actions dans un environnement partagé". Cette notion de ressources partagées dans l'environnement implique la nécessité de la coordination. Les agents devraient coordonner leurs actions individuelles avec les autres pour aboutir à l'objectif global du groupe. Cette coordination permet alors :
  – d'éviter les situations de conflits par la négociation pour les agents antagonistes (ayant des buts et des objectifs contradictoires),
  – d'améliorer l'efficacité et l'utilité de chaque agent par la coopération pour les agents non antagonistes.

- **Négociation** : C'est une méthode de coordination qui permet à plusieurs agents, d'atteindre suite à un processus de communication et d'échange d'informations un accord mutuel pour entreprendre une action donnée d'une certaine manière. Elle induit, par cette communication, des relaxations de buts initiaux, des concessions mutuelles, des mensonges ou des menace [6] . Elle est donc considérée comme une méthode de résolution de conflits et de recherche de consensus.

- **Coopération** : D'après [5], la coopération est la coordination parmi des agents non antagonistes, qui cherchent à se satisfaire mutuellement sans se gêner. Cette coopération, initiée par un échange d'information, est souvent associée à la notion de collaboration. la collaboration est une forme d'interaction qui étudie la manière de répartir le travail, et par conséquent l'allocation de tâches, entre plusieurs agents. En effet l'allocation de tâches peut se faire dès la conception du système multi-agents, en définissant une organisation de résolution de problèmes qui est non adaptable. Mais dans d'autres cas, on peut avoir une allocation de tâches qui se fait d'une manière dynamique suite à un processus de coopération entre agents, de manière à ce que la planification des actions individuelles à entreprendre par chaque agent, se fasse aussi d'une manière dynamique.

2.1.4.4 Organisation Muti-Agents

L'organisation d'un système multi-agents, définit l'architecture globale du système. Elle précise aussi pour chaque agent son rôle et ses fonctions par rapport au groupe, et les règles d'interaction à adopter dans son environnement. Ces organisations d'agents logiciels s'inspirent des organisations

des sociétés humaines et animales, et ce pour la définition des rôles et des interactions.

2.1.5 Roles des systèmes Multi-Agents

– Résoudre un problème de manière distribuée :systèmes multi-experts.
Les actions des agents sont des transformations d 'objets liées à la description d 'un problème.
Agents plutôt rationnels
– Simulation de phénomènes complexes.
Les agents simulent des actions physiques, biologiques ou sociales qui produisent des modifications du monde représenté. Ex : simulation de la pêche dans le delta du Niger, des épidémies, écosystèmes (proies / prédateurs).
Agents plutôt réactifs.

– Gérer et maintenir un environnement de travail.
Les actions physiques ou sociales réalisées par les agents sont des actions réelles, elles évoluent dans le temps et modifient le monde : robots footballeurs, agents négociant un rendez-vous au profil de l 'utilisateur.
–Agents plutôt cognitifs et sociaux.

2.1.6 Caractéristiques des systèmes Multi-Agents

Un système Multi Agents posséde géenéralement les caractéristiques suivantes :

1. Il n'y a pas de controle global du système.
2. Les données sont décentralisées.
3. Le calcul est asynchrone.
4. Chaque agent a des informations ou des capacités de résolution limitées de problème, ainsi chaque agent a un point de vue partiel.

2.1.7 Quand utiliser un Système Multi-Agents ?

– Compléxité inhérente de l'application
Le problème est trop complexe pour etre résolu par un seul système du fait de limitations logicielles ou materielles.

– Distribution inhérente de l'application
Existence de différents domaines de connaissances

distribution de données, du controle, des connaissances, des ressources.

– Contraintes d'exécution
Volonté d'avoir des résolutions concurrentes, simultanées, asynchrones.
Satisfaction de contraintes de fiabilité, de contraintes phisyques..

– Besoin d'évolutivité Adaptation au modifications et/ou à l'environnement.

– Besoin d'ouverture Le système doit pouvoir s'adapter dynamiquement au retrait/ajout de nouveaux composants .
Développement incrémental.

2.1.8 Apport des SMA dans la recherche de l'information dans des systèmes distribués

Dans un système distribué un utilisateur qui cherche une information se sent, de ce fait, "noyé" dans une masse d'informations pourtant disponible. Les agents permettent à des systèmes d'information d'accélérer et de rendre plus efficace la recherche sur des sources hétérogènes et distribuées pour répondre à la requête d'un utilisateur

Parmi les applications des agents dans les systèmes distribués sont
-La découverte et l'analyse de sources de données à distance, et
-L'intégration de l'information. Nous nous proposons dans la suite d'étudier les différents cas d'utilisation des agents par rapport à la recherche et l'intégration de l'information dans un environnement d'information distribué.

Agents Sources d'Information (Adaptateurs ou "Wrappers") Ce sont les agents qui permettent de récupérer des données dans des sources d'informations hétérogènes et distribuées sur le réseau. Il peuvent communiquer avec une source de données à distance pour extraire des informations. Il sont dits "wrappers" [7] [8] ou adaptateurs, puisqu'ils permettent d'adapter le contenu et le format des informations récupérées sur les sources distantes, de leur format d'origine, au format requis par l'application SMA qui les utilisent. Ce sont des traducteurs qui permettent à un système d'information orienté agents de communiquer avec des sources de données hétérogènes ; ils sont capables de traduire les données récupérées en langages SMA et vice versa.

En effet ces agents automatisent le processus d'accès à distance aux différentes sources de données. Ces sources d'informations peuvent être des bases de données distantes, des sites web sur internet, des applications sur des serveurs distants, ou même des service web. Pour accéder à ces sources de données, chaque agent doit maîtriser le protocole de communication de la source de données dont il est responsable. Ainsi on peut avoir : – Des agents de bases de données : il sont capables de communiquer avec des SGBD à distance. -Agents Internet : ces agents sont capables de récupérer des informations sur des pages web ou de remplir des formulaires web d'une manière automatisée.

Agents mobiles Un agent mobile est un agent qui part d'un serveur de départ avec un ensemble prédéfini de tâches à exécuter sur différents serveurs du réseau. Son itinéraire dans le réseau peut être prédéfini dès son départ, ou complété sur chaque serveur hôte. Ces agents sont souvent associés à la recherche et la collecte de l'information sur plusieurs sources d'informations distribuées sur le réseau [9], si ces tâches prédéfinies sont un ensemble de requêtes à exécuter. Ces sources d'information peuvent aussi bien être des bases de données distribuées [**?**] que des pages web sur Internet [10].

Agents Médiateurs Les agents Médiateurs sont aussi appelés des agents de médiation de sources de données. En effet ils fournissent à ces sources des mécanismes d'interopérabilité. Ces mécanismes comprennent en plus du protocole de communication, des ontologies décrivant les données contenues dans ces sources
Wiederhold [11] a introduit le concept de médiateur, ou manager de requêtes, entre bases de données hétérogènes et distribuées. Le médiateur qui reçoit une requête de l'utilisateur, est capable de la décomposer en sous-requêtes qu'il envoie ensuite, aux bases de données correspondantes généralement hétérogènes et distribuées. Il assure ainsi la médiation entre l'utilisateur ou l'application qui envoie la requête globale et les différentes sources de données. Ceci suppose, que le médiateur connaisse, non seulement le domaine du contenu (domaine de connaissances) de ces sources mais aussi leurs adresses sur le réseau.
Ces agents sont des facilitateurs, qui facilitent l'accès à plusieurs sources d'information. Il sont aussi utilisés pour la médiation, entre un utilisateur ou un agent interface et plusieurs agents "wrappers" sources d'informations. Les agents de médiation sont aussi appelés, selon [12], des agents d'exécution de

tâches : "Taskagent" , puisqu'ils vont coordonner la répartition et l'exécution des requêtes sur les agents sources d'informations. Ils sont aussi appelés "Agents courtier" ou "broker agents", puisqu'ils assurent le courtage d'information entre clients et fournisseurs. Enfin par le courtage, la médiation, la découverte, le filtrage et l'intégration, ces agents assurent la fonction de recherche d'information dans sa globalité.

Agents Annuaires Il est difficile pour un agent médiateur de disposer d'une connaissance complète sur tous les agents sources d'information. Dans ce cas, il est nécessaire de disposer d'un agent annuaire (Directory Agent) qui disposerait d'une vision globale sur les domaines de connaissances et les adresses de tous les agents sources de données de l'environnement. Cet agent permet d'orienter et d'aiguiller les activités et les sous requêtes des agents médiateurs, et ce selon les requêtes que ces derniers reçoivent [13]. Néanmoins, les agents médiateurs gardent leurs compétences d'interrogation, de collecte et d'intégration d'informations.

Concernant les travaux sur les architectures ou organisations des systèmes, une solution consistant à faire circuler les requêtes de proche en proche, d'une source à l'autre, est proposée dans [17]. Cette solution est particulièrement intéressante car elle permet à un groupe de participants de gérer eux-mêmes le routage sans dépendre d'un tiers. Cependant, elle nous semble peu adaptée à notre problématique car chaque fournisseur serait alors sollicité au moins une fois pour chaque requête, ce qui représente une charge trop importante.
Deux architectures majeures utilisant des agents intermédiaires ont été proposées. Le projet INFOSLEUTH [18] propose une architecture pour déployer des applications agents qui focalisent sur la collecte et l'analyse d'informations à partir de réseaux dynamiques de sources d'informations. Le projet RETSINA [19] a commencé par s'intéresser aux agents intermédiaires et [20] n'a proposé une infrastructure finalisée que bien plus tard.
Ces deux projets considérent un champ trés vaste de problémes allant du langage de communication entre les agents, à la gestion de l'interopérabilité sémantique, en passant par les problémes de déclaration de capacité et de services nécessaires pour répondre à n'importe quel type de demande. Les solutions sont donc relativement complexes et les applications dans lesquelles elles sont utilisées sont de taille modeste en nombre de sources d'information.

2.1.8.1 La recherche de l'information dans le web [WWW]

Récemment, il ya un intérêt croissant pour l'utilisation des agents intelligents qui aident les utilisateurs dans le Web. En raison des caractéristiques : flexibles et dynamiques des agents intelligents, ils sont largement utilisés en tant qu'interface système entre l'utilisateur et le WWW pour différentes applications. Par exemple, [14] ont développé un agent qui aide l'utilisateur à la recherche littérature et scientifique, [15] un agent qui trouve des pages Web pour l'utilisateur. D'autres tentatives telles que [16] ont utilisé une approche multi-agents pour aider les utilisateurs avec un intérêt commun pour partager des pages Web, dans [15] ils ont proposé un agent intermédiaire entre un utilisateur et une variété de ressources d'information pour faciliter la recherche.

De nombreux travaux ont été élaborés afin d'introduire la technologie d'agents mobiles et les concepts liés à cette dernière pour la recherche d'information dans des environnements dynamiques. Le concept d'agent mobile apparaît dans ce contexte comme une solution facilitant la mise en oeuvre d'applications réparties.
Parmi ceux-ci on distingue :

- Le système DBMS-Aglet [17] implante une solution à base d'agents mobiles en Java pour l'interrogation de bases de données hétérogènes via le Web. Un agent mobile transporte la requête sur le site serveur où il acquiert dynamiquement le pilote JDBC qui convient, il pose ensuite sa requête et retourne sur le site client avec les résultats.
- M3 "MultiMedia Database Mobile agents" [18] est un système de recherche de données multimédia par le contenu qui repose sur les agents mobiles, Java et CORBA. L'agent mobile peut mémoriser les informations recueillies sur un site, les utiliser sur les sites visités ensuite, les faire évoluer pendant le parcours. Les problèmes de sécurité sont pris en compte via des mécanismes de sessions indépendantes, les mécanismes de sécurité de CORBA, et des restrictions de droits.
- Enfin d'autres travaux comme AGATHE [19], ARCADIA [20], JAVANE [21], NETSA [22], ISAME [23]proposent une autre alternative pour la recherche d'information : des modèles d'agents mobiles et multi-agents.

Tous ces systèmes sont des systèmes d'information coopératifs, qui en coopérant assurent une recherche et une intégration d'informations.

2.1.9 Outils de développements

2.1.9.1 Méthodologie de conception

La méthodologie de conception des agents est toujours en cours de recherche, il n'existe pas encore une méthode standard car chaque groupe de recherche propose différentes méthodes qui convient à leurs application.
On va présenter quatre méthodes typiques qui sont nées dans les derniers années :
· MAS CommonKADS
· MaSE
· Gaia
· OMaSE

MAS-CommonKADS MAS-CommonKADS(Multiagent System – Knowledge Analysis and Development System) est étendu de la méthodologie de génie connaissance en prenant les techniques orientées objet et celles de la méthodologie de génie protocole. Elle contient sept modèles :

· *Modélisation des agents* :détermine les caractéristiques de l'agent : Les capacités de raison, les habiletés, les détecteurs et les effecteurs, les services, les buts, etc.
· *Modélisation des tâches* : décrit les tâches (les buts) devant être exécutés par les agents, et la décomposition des tâches, en utilisant les spécimens textuels et les diagrammes.
· *Modèle d'expertise* : détermine les connaissances dont les agent ont besoin pour obtenir leurs objectifs.
· *Modèle d'organisation* : détermine l'organisation dans laquelle le système multi-agent sera introduit et l'organisation de la société d'agents.
· *Modèle de coordination* : détermine les conversations entre les agents : les interactions, les protocoles et les capacités nécessaires.
· *Modèle de communication* : détaille les interactions entre l'agent humain et le logiciel, et les facteurs humains nécessaires au développement de ces interfaces utilisateur.
· *Modèle de conception* : recueille les modèles précédents et est subdivisé en trois sous-modèles :

– Conception de réseau , il conçoit les fondations du réseau d'agent.
– Conception d'agent, conçoit l'architecture interne de l'agent.
– Conception de platforme, sélectionne la platforme de développement pour chaque architecture d'agent.

MaSE MaSE (Multiagent System Engineering) : Elle considère un agent comme un type d'objet, soit ayant de l'intelligence, soit non. L'objectif de MaSE est d'aider le concepteur à analyser, concevoir et implémenter un SMA à partir d'un cahier des charges initial. MaSE comporte sept étapes réparties en deux phases. Chacune des étapes a pour résultat, un ou plusieurs diagrammes.
La phase d'analyse comprend trois étapes :
l'identification des objectifs, dont le résultat est le diagramme hiérarchisé des objectifs ;
l'identification des cas d'utilisation, dont le résultat est un ensemble de cas de d'utilisation exprimés par un ou plusieurs diagrammes de séquence (à la UML) ;
l'affinage des rôles, dont le résultat est le modèle de rôles de MaSE et un ensemble de diagrammes de tâches concurrentes.
La phase de conception comporte quatre étapes :
la création des classes d'agent, dont le résultat est le diagramme de classes d'agent ; *la construction des interactions entre agents*, dont le résultat est un ensemble de diagrammes d'interactions ; *l'assemblage des classes d'agents*, dont le résultat est l'architecture interne des agents ; *la conception du système*, dont le résultat est le diagramme de déploiement (à la UML).

Gaia La méthodologie Gaia permet de parcourir systématiquement le chemin qui commence par l'énoncé des demandes du problème et mène à une conception assez détaillée pour être implémentée tout de suite.
· *Modèle d'organisation* :divise le système en plusieurs sous-systèmes. selon l'identification des sous-systèmes qui existent déjà dans le système.
· *Modèle d'environnement* : considère l'environnement en terme des ressources calculées abstraites. Il est considéré comme une liste des ressources.
· *Modèle préliminaire de rôle* : définition préliminaire des rôles et des protocoles de l'organisation.
· *Modèle préliminaire d'interaction* : capte les indépendances et les relations entres les rôles dans le système. · *Modèle de règle organisationnelle* : la règle organisationnelle est considérée comme le responsabilité de l'organisation.

O-MaSE (Organization – based Multiagent System Engineering) : Cette méthode considère un système multi-agent comme une organisation des agents dans laquelle les agents sont membres, chaque agent joue un rôle spécifique selon ses capacités pour obtenir ses objectifs. Donc, le but de cette méthode est de construire une société organisationnelle.
Les étapes de la méthode :

· *Modèle des buts* : arbre hiérarchie des buts.
· *Modèle préliminaire d'organisation* : définit les interactions avec des acteurs externes. L'organisation peut être divisée par plusieurs sous-organisations selon les buts d'organisation.
· *Modèle des rôles* : définit les rôles dans l'organisation, les services qu'ils fournissent et les capacités dont ils demandent pour les exécuter.
· *Modèle d'ontologie* : définit les entités dans le domaine d'application de l'organisation. Chaque entité est décrite par ses propres attributs et relations.
· *Modèle des agents* : définit les agents et leurs relations dans l'organisation.
· *Modèle de protocole* : définit les protocoles entre les agents. Chaque protocole est décrit par un diagramme de séquence qui indique les participants, les messages échangés, l'ordre d'échanger des messages.
· *Modèle d'état d'agent* : définit le comportement de chaque agent en utilisant le diagramme des états finis.

2.1.9.2 Plateforme agent

Il existe un nombre important d'environnements de développement des applications orientées agents : il y a aussi bien des produits commerciaux que des logiciels dans le domaine public.
Parmi les plates-formes fournies comme logiciels libres, il y quelques plates-formes plus connues pour avoir été utilisées dans le développement de plusieurs applications : JADE, MACE, ZEUS, et MADKIT pour les agents cognitifs, et SWARM pour les agents réactifs. Il faut noter que cette liste n'est pas unique, et qu'il y a aussi d'autres plates-formes qui ont été utilisées avec beaucoup de succès pour bâtir diverses applications.

MACE est le premier environnement de conception et d'expérimentation de différentes architectures d'agents dans divers domaines d'application. Dans MACE, un agent est un objet actif qui communique par envoi de messages. Les agents existent dans un environnement qui regroupe tous les autres agents et toutes les autres entités du système. Un agent peut effectuer trois types d'actions : changer son état interne, envoyer des messages aux autres agents et envoyer des requêtes au noyau MACE pour contrôler les événements internes. Chaque agent est doté d'un moteur qui représente la partie active de l'agent. Ce moteur détermine l'activité de l'agent et la façon dont les messages sont interprétés. MACE a été utilisé pour développer des simulations d'applications distribuées.

MADKIT La pateforme MadKit dévelopée à l'universtié de Montepellier II, est basée sur la notion d'agent, de groupe et de rôle. MadKit fournit une Api permettant la construction d'agent en spécialisant une classe d'agent abstraite. Chaque agent peut tenir différents rôles au sein de différents groupes. Les agens sont lancés par le noyau de MadKit, qui propose notamment les services de gestion des groupes et de communication. Il est ainsi possible d'échanger des messages directement à un agent ou à l'ensemble d'un groupe. Cette plateforme est surtout intéressante pour l'approche organisationnelle qu'elle met en avant lors de l'analyse et de la conception d'un SMA.

ZEUS [24] est une plate-forme multi-agents conçue et réalisée par British Telecom (Agent Research Programme of BT Intelligent Research Laboratory) pour développer des applications collaboratives. ZEUS est écrit dans le langage Java et il est fondé sur les travaux de la FIPA. L'architecture des agents ZEUS est similaire à la majorité des agents collaboratifs. Elle regroupe principalement les composantes suivantes :

- une boîte aux lettres et un gestionnaire de messages qui analyse les messages de la boîte aux lettres et les transmet aux composantes appropriées ;
- un moteur de coordination ;
- un planificateur qui planifie les tâches de l'agent en fonction des décisions du moteur de coordination, des ressources disponibles et des spécifications des tâches ;
- plusieurs bases de données représentant les plans connus par l'agent, les ressources et l'ontologie utilisée ;
- un contrôleur d'exécution qui gère l'horloge locale de l'agent et les tâches actives.

JADE (Java Agent Development Framework - Bellifemine, Poggi, Rimassa, 1999) est une plate-forme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA (FIPA, 1997). JADE comprend deux composantes de base : une plate-forme agents compatible FIPA et un paquet logiciel pour le développement des agents Java.

SWARM (Minar e.a., 1996) est une plate-forme multi-agents avec agents réactifs. L'inspiration du modèle d'agent utilisé vient de la vie artificielle. SWARM est l'outil privilégié de la communauté américaine et des chercheurs en vie artificielle. L'environnement offre un ensemble de bibliothèques qui permettent l'implémentation des systèmes multi-agents avec un grand nombre d'agents simples qui interagissent dans le même environnement. De

nombreuses applications ont été développées à partir de SWARM qui existe aujourd'hui implémenté en plusieurs langages (Java, Objective-C).

2.1.9.3 La norme FIPA pour les systèmes multi-agents

FIPA est une organisation dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes [25], établissant les règles normatives qui permettent à une société d'agents d'inter opérer. Les documents FIPA décrivent le modèle de référence d'une plateforme multiagents ou il identifient les rôles de quelques agents clés nécessaires pour la gestion de la plateforme, et spécifient le contenu du langage de gestion des agent et l'ontologie du langage.Trois rôles (agent) principaux sont identifés dans une plateforme d'agent [26] :
1. Le Système de gestion d'Agent (AMS) Agent qui exerce le contrôle de supervision sur l'accès et l'usage de la plateforme ; il est responsable d'authentifier les agents résidents et de contrôler la les enregistrements.
2. Le Canal De communication ACC) Agent qui fournit le chemin pour les interactions de base entre les agents dans et en d'hors de la plateforme ; c'est la méthode de communication implicite qui offre un service fiable et précis pour le routage des messages ; il (l'agent) doit aussi être compatible avec le protocole IIOP pour assurer l'interopérabilité entre les différentes plateformes.
3. Le faciliteur d'Annuaire (DF) Agent qui fournit un service de pages jaunes à la plateforme. FIPA spécifie aussi le Langage de Communication d'agents ACL. La communication entre agents ne se fait que par envoi de messages seulement.

FIPA-ACL est le langage standard des messages et impose le codage, la sémantique et la pragmatique des messages. La norme n'impose pas de mécanisme spécifique pour le transport interne de messages.Plutôt, puisque les agents pourraient s'exécuter sur des plateformes différents et utilisent des technologies différentes 'interconnexion, FIPA spécifie que les messages transportés entre les différentes plateforme devrait être codé sous forme textuelle.

2.2 Base de données distribuée

2.2.1 Introduction

Le récent avancement des systèmes informatiques rend les systèmes des bases de données distribuées plus attractifs, ils sont venus à être de plus en plus sujet de discussion. Ces systèmes permettent aux utilisateurs d'utiliser des bases de données distribuées géographiquement en tant qu'une base de données logique unique. L'Amélioration de la fiabilité du système, la réduction du coût de communication et la facilité de l'expansion de la capacité du système sont attendus par la réalisation de ces systèmes. Cependant, l'efficacité de ces systèmes dépend en grande partie du volume du trafic entre les sites de base de données, qui peut souvent être un grand facteur de goulot d'étranglement de performance. Si une représentation goulot survient, le temps du traitement des requêtes sur les sites connexes seront augmentés. Une telle situation provoque une perte des avantages majeurs mentionnés ci-dessus. En conséquence, le facteur majeur des systèmes de base de données distribuée consiste à limiter le volume du trafic entre les sites autant que possible.

2.2.2 Notions de système distribués

Selon Tanenbaum [27], un système réparti est un ensemble d'ordinateurs (ou processus) indépendants qui apparaît à un utilisateur comme un seul système cohérent. Les ordinateurs peuvent garder leur autonomie et être regroupés dans un même lieu ou dispersés sans que cela ne soit visible de l'extérieur par un utilisateur. Du fait que l'ensemble des ordinateurs forment un système en entier, la défaillance d'un ordinateur peut avoir un impact négatif le fonctionnement du système et introduire des incohérences. En prenant en compte cet aspect, un système distribué peut être défini comme "un système qui vous empêche de travailler si une machine dont vous n'avez jamais entendu parler tombe en panne, (Leslie Lamport). S'il existe moult définitions dont nous ignorons le nombre, on peut dire que les principaux objectifs des systèmes répartis sont de faire coopérer plusieurs ressources dans l'optique de partager des tâches, de faire des traitements parallèles, etc . Ainsi, un système distribué peut être vu comme une application qui coordonne les tâches de plusieurs équipements informatiques. Cette coordination se fait le plus souvent par envoi de messages via un réseau de communication qui peut être un LAN (Local Area Network), WAN (Wide Area Network), Internet, etc.

2.2.3 Qu'est ce qu'une base de données distribuée ?

Une base de données distribuée (BDD) est une collection de sites connectés par un réseau de communication. Chaque site est une base de donnée centralisée qui stocke une portion de la base de données. Chaque donnée est stockée exactement sur un seul site [28]. La gestion d'une base de données distribuée est gérée de manière transparente par un SGBD distribuée. Les transactions peuvent être envoyées sur chaque site puis traduites en transactions locales avant d'être routées sur les sites appropriés (stockant une portion des données manipulées). Les résultats sont intégrés puis renvoyés aux applications clientes.

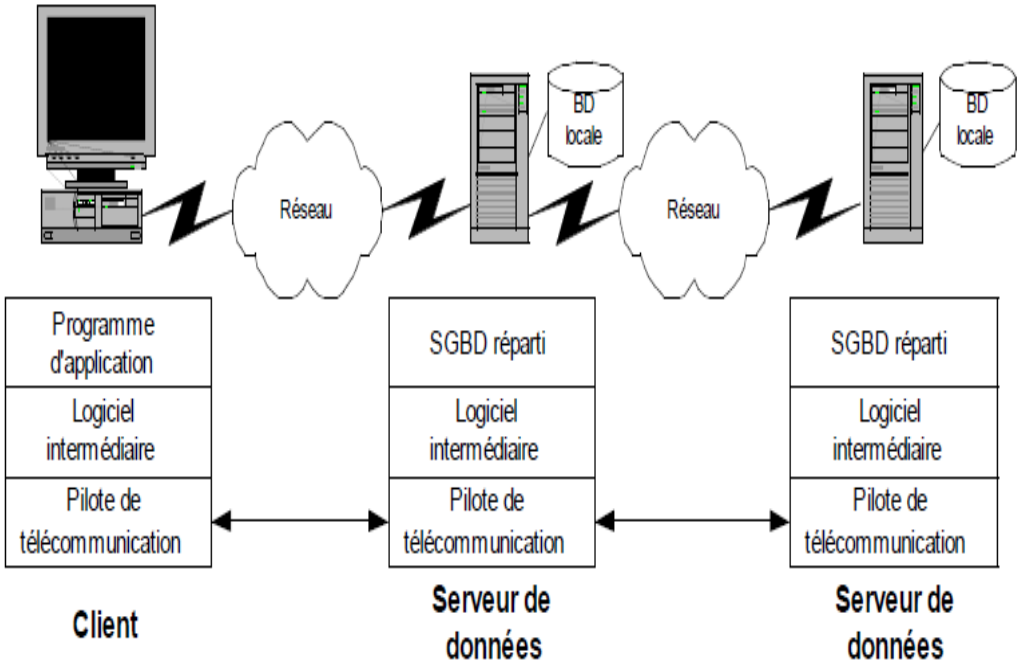


FIGURE 2.4 – Une base de données distribuée

2.2.4 Système de gestion des bases de données distribuée (SGBDD)

Le SGBD (Système de Gestion des Bases de Données) est l'outil principal de gestion d'une base de données. Il permet d'insérer, de modifier et

de rechercher efficacement des données spécifiques dans une grande masse d'informations. C'est une interface entre les utilisateurs et la mémoire de masse. Il facilite ainsi le travail des utilisateurs en leur donnant l'impression que l'information est organisée comme ils le souhaitent. Une base de données distribuée est gérée par plusieurs processeurs, sites ou SGBDs. Un système de bases de données distribuées ne doit donc en aucun cas être confondu avec un système dans lequel les bases de données sont accessibles à distance. Il ne doit non plus être confondu avec une multibase ou une BD fédérée.
Du point de vue organisationnel nous distinguons deux architectures :
1. Architecture Client-Serveur : les serveurs, ont pour rôle de servir les clients. Par servir, on désigne la réalisation d'une tâche demandée par le client.
2. Architecture Pair-à-Pair (Peer-to-Peer, P2P) : par ce terme on désigne un type de communication pour lequel toutes les machines ont une importance équivalente.

2.2.5 Traitement des requêtes

La figure suivante montre le manuel d'achitecture classique pour le traitement des requêtes. Cette architecture a été utilisé, par exemple, dans IBM Starburst projet [Haas et al. 1989]. Elle peut être utilisé pour tout type de système de base de données y compris centralisée, systèmes parallèles ou distribués. Le processeur reçoit une requête SQL comme entrée, traduit et la découpe en un ensemble de sous-requêtes en fonction des fragments nécessaires. Chaque sous-requête est dupliquée, d'une part, autant de fois que le fragment à traiter dispose de répliques et d'autre part, en fonction de la localisation de la réplique puis optimise cette requête en plusieurs phases en un plan de requête exécutable, et exécute le plan pour obtenir les résultats de la requête.
Catalogue :
Dans un système de base de données centralisée le catalogue est principalement utilisée pour stocker le schéma, qui contient des informations concernant les relations, les indices et les vues. La méta-données stocké pour les relations comprend le nom de la relation, les noms d'attributs et les types de données et les contraintes d'intégrité. Diverses statistiques sont également stockées dans le catalogue, comme le nombre de clés distinctes dans un attribut donné et la cardinalité d'une relation particulière. Ces statistiques aident l'optimiseur des requêtes dans l'estimation de la taille des résultats intermédiaire des plans d'exécution, ce qui permet à l'optimiseur de déterminer une estimation des coûts des plans. L'informations sur l'état actuel du système est également mise à disposition dans le catalogue, y compris le nombre de pages de mémoire tampon dans le pool et la taille de la page Système en

octets [29].
Dans un SGBDD le catalogue doit stocker des informations supplémentaires,l' emplacement des relations et de leurs répliques. Le catalogue doit également inclure l'ensemble du système des informations telles que le nombre de sites dans le système ainsi que leurs identificateurs. Une question qui se pose dans un cadre distribué est où placer le catalogue dans le système. Deux alternatives : stocker le catalogue sur un site unique ou le répliquer sur tous les sites.
Stocker le catalogue sur un seul site présente un point de défaillance unique dans le système. Si le site contenant le catalogue échoue, alors l'ensemble du système ne peut pas accéder au catalogue, ce qui peut entraîner l'ensemble du système à un arrêt.
Placer une copie du catalogue sur un site unique provoque la surcharge du site par beaucoup de requêtes provenant d'autres sites, qui peuvent conduire à une dégradation des performances. La création des répliques du catalogue à chaque site élimine le problème unique de défaillance et réduit également la communication de réseau nécessaire pour tous les sites d'accéder au catalogue à distance. Toutefois, dans un système où les mises à jour du catalogue sont fréquentes, beaucoup de temps peut être investi dans la synchronisation des répliques catalogue, qui peut dégrader les performances du système[30].

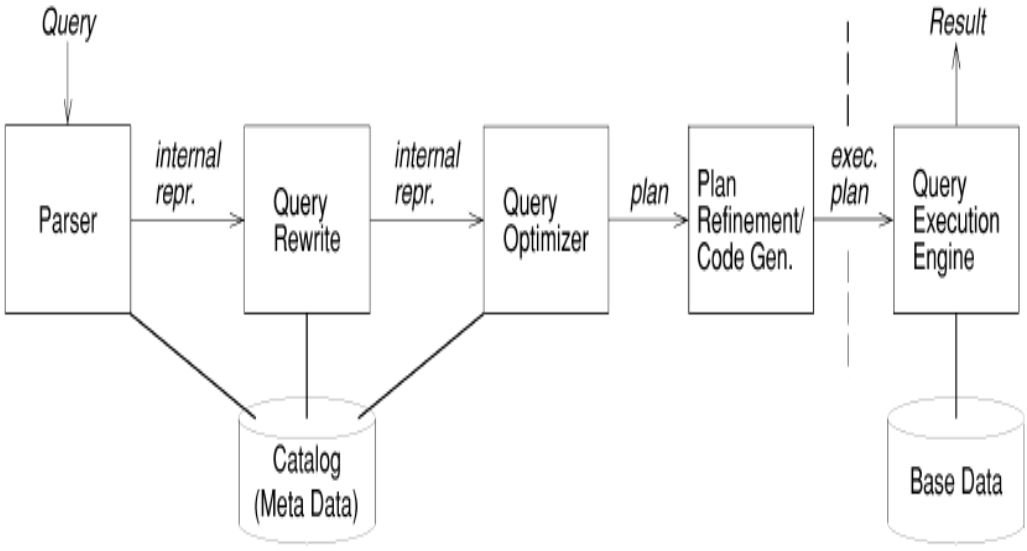


FIGURE 2.5 – Traitement des requêtes [Haas et al. 1989]

2.2.6 Travaux sur les BDs les SGBDs réparties

Routage des Transactions dans les Bases de Données à Large Echelle

La réplication dans les bases de données a été largement étudiée, au cours des trois dernières décennies. Elle vise à améliorer la disponibilité des données et à augmenter la performance d'accès aux données. Un des défis majeurs de la réplication est de maintenir la cohérence mutuelle des répliques, lorsque plusieurs d'entre elles sont mises à jour, simultanément, par des transactions. Des solutions qui relèvent partiellement ce défi pour un nombre restreint de bases de données reliées par un réseau fiable existent. Toutefois, ces solutions ne sont pas applicables à large échelle car elles nécessitent une communication rapide et fiable entre les noeuds traitant les transactions ; ce qui limite le nombre de noeuds (de l'ordre d'une centaine) et le type d'interconnexion (réseau local).
SARR [31] s'intéressa à la gestion des transactions dans une base de données répliquées à large échelle et particulièrement au routage des transactions. ses principaux objectifs peuvent être résumés comme suit :
– réduire le temps de réponse des transactions, en équilibrant la charge des répliques et en tenant compte de la disponibilité des ressources (SGBD, gestionnaire de transactions).
– contrôler la cohérence des accès aux données réparties et répliquées afin de rendre aux applications des résultats conformes à leur exigence ;
– garantir l'autonomie des applications et des SGBD i.e. pouvoir intégrer les solutions proposées avec des applications et SGBD existants en les modifiant le moins possible.

Gestion des données pour les systèmes hautement distribués , Le réseau est la base de données

Navas et Wynblatt [32] se sont intéressés à la méthodologie et la mise en œuvre d'un Système de gestion de données pour des systèmes hautement distribués, qui ont été construit pour résoudre les problèmes d'évolutivité et de fiabilité rencontrés dans une application de la logistique postale développé chez Siemens. Le cœur de l'approche consiste à emprunter d'Internet les protocoles de routage, et leur évolutivité et la robustesse éprouvée, pour construire un indice de base de données de réseau intégré dynamique, et d'augmenter l'optimisation des schémas pour utiliser cet indice.

Proposition d'un cadre générique d'optimisation de requêtes dans les environnements hétérogènes et répartis

L'objectif de cette thèse est de définir un cadre générique d'optimisation de requêtes, qui permet d'intégrer de différentes techniques d'optimisation de

requêtes pour construire efficacement des optimiseurs, dans le contexte d'un système de médiation de sources de données hétérogènes et réparties. Cette thèse a été effectuée dans le cadre d'un contrat CIFRE. Le travail de la thèse a été mis en oeuvre dans le module d'optimiseur d'un produit d'intégration de données commercialisé par l'entreprise XCalia - Progress Software Corporation appelé DVS (Data Virtualization Server). L'optimiseur de DVS nécessite une implémentation souple permettant d'intégrer facilement différentes stratégies de recherche pour différents types de requête. L'évolution de l'optimiseur pour l'intégration des éléments d'optimisation de requêtes (règles de transformation, algèbre, etc.) ne doit pas demander que toute l'implémentation soit refaite. De plus, dans [33], un processus de modélisation, optimisation et évaluation de requêtes a été proposé, afin d'évaluer les requêtes hétérogènes semi-structurées répondant aux exigences de la médiation et du traitement de requêtes. Un modèle de présentation, permettant de modéliser toutes les requêtes XQuery non-typées, appelé Tree Graph View (TGV) est proposé. Ce modèle TGV est utilisé comme plan d'exécution dans le processus d'évaluation de XQuery et peut être transformé pour des raisons d'optimisation. Ce contexte de travail nécessite toutefois un travail étendu sur l'optimisation de requête, qui fait une partie de l'objectif de cette thèse.[34]

Algorithme Génétique pour l'optimisation des requêtes des bases de données distribuées

L'Optimisation des requêtes de base de données distribué relationnelle est un problème d'optimisaltion combinatoire. Cette recherche rend compte d'une enquête initiale sur la possibilité d'un algorithme génétique (AG) pour l'optimisation des requêtes distribuées. Un algorithme génétique est développé et sa performance comparer à des techniques alternatives d'optimisation stochastiques : recherche aléatoire, MultiStart et recuit simulé. Le problème de réduction de toutes les tables dans une requête est utilisé pour comparer les techniqules. Pour ce problème, l'évaluation de la la fonction de remise en forme est une opération coûteuse. Le projet de AG utilise un modèle de données arborescente avec les opérateurs de croisement et de mutation qui évitent la nécessité de entièrement re-évaluer la fonction de remise en forme de nouvelles solutions. L'optimisation de requête est une tâche qui doit être exercée dans un temps réel. Une technique est nécessaire qui fonctionne bien au début d'une recherche, mais évite le problème de convergence prématurée. L'AG proposé utilise une phase de recherche locale qui fournit les performances requises en temps réel. les expériences montrent que le projet de AG peut faire mieux que le techniques alternatives, testées. Le potentiel d'un AG d'offrir une réduction précieuse du coût pour le traitement des requêtes distribuées est démontrée[35].

2.2.6.1 Conclusion

Dans ce chapitre nous avons proposé l'approche multi-agents comme une approche récente et utile pour étudier les systèmes complexes ou distribués, et essayé de présenter brièvement les composantes d'un SMA.
La plupart des travaux tentent de développer un système multi-agent qui agit en tant que médiateur pour aider l'utilisateur à localiser, récupérer et intégrer l'information. Les agents collectivement :

1. Fournissent aux utilisateurs des information avec l'illusion d'un système d'information unique.
2. Recherchent des informations de manière proactive provenant de différentes sources distribuées et en évitant la répétition pour satisfaire leur intérêt.
3. Fournissent des informations qui sont pertinentes à l'intérêt de l'utilisateur.
4. Surveillent et mettre à jour les changements des ressources d'information périodiquement.
5. Fournissent une réponse dans un délai spécifié.

Pour le coté développement, on a présenté quelques méthodologies, nous avons choisie MaSE pour l'analyse et la conception. Cette méthode est simple, détaillée (utilise les diagrammes d'UML pour décrire le fonctionnement des agents dans le systèmes) et donne la possibilité de passer d'une étape à une autre efficacement. Pour l'implémentation nous avons opté pour la plateforme JADE (définie dans la partie 2.1.9.2)et le SGBD SQL server Express 2008 et enfin nous avons décidé d'utiliser l'architecture peer 2 peer (définie dans la partie 2.2.4) car chaque pair pourra communiquer directement avec l'ensemble de ses voisins.

Troisième partie

Conception et implémentation

Chapitre 3

Conception de l'approche

3.1 Introduction

Dans le chapitre précédent on a vu une étude détaillée sur les notions des systèmes multi-agents (SMA) et quelques notions des bases de données distribuées (BDD).

Dans ce chapitre, nous présentons la modélisation de notre travail qui portera sur la recherche dans les BDD par une approche à base d'agent. Pour le réaliser on va appliquer la méthodologie MaSE définie dans le chapitre précèdent.

3.2 Les étapes de MaSE

1. **Identier les objectifs**
   La première étape est d'identifier les objectifs en transformant les spécifications initiales du système en objectifs structurés. Le point de départ pour identifier les objectifs est le contexte initial du système et les spécifications et l'analyse des besoins. Ces besoins expriment les services que le système doit fournir et quel comportement le système doit avoir, selon ses données et son état actuel.
   L'identification se fait en deux étapes :
   (a) **Numéroter les objectifs :**

   Dans cette étape il faut faire sortir l'essentiel des objectifs à partir des besoins, extraire des Scénarios à partir des spécifications initiales et décrire l'objectif de chaque scénario.
   Nous pouvons identifier les objectifs suivants :

i. Rechercher dans une BDD
ii. Analyse syntaxique de la requête
iii. Consulter le catalogue
iv. Décomposition de la requête
v. Réecriture de la requête en arbre algébrique
vi. Optimisation global (Coût de transfert)
vii. Routage de la requête
viii. Recherche local
ix. Jointure
x. Affichage du résultat

- **Rechercher dans une BDD** c'est l'objectif principal de l'approche, les autres sont les sousobjectifs qui aideront à réaliser celuici.
- **Consulter le catalogue** cet objectif nous permettra d'extraire l'emplacement et les statistiques des tables citées dans la requête depuis le catalogue.
- **Décomposition de la requête** après la consultation du catalogue, la décomposition de la requête en des sousrequête pourra se faire car on aura la localisation des tables.
- **Optimisation global** cet objectif nous permettra de prendre le coût minimal de transfert des données, on aura plusieurs scénarios et on utilisera le plus optimal. Chaque sites recevra le plan à suivre pour renvoyer le résultat.
- **Routage de la requête** c'est le transport des sousrequêtes vers leurs destination suivant le plan choisit.
- **Recherche local** c'est l'exécution de la sousrequête qui se fera dans chaque sites ayant reçu la requête.
- **Jointure** c'est la jointure des résultats envoyés depuis les autres sites suivant le plan.

(b) **Structurer les objectifs :**

Organiser les objectifs d'une façon hiérarchique dans un graphe acyclique ou les nœuds représentent les objectifs et les arcs définissent la relation entre un objectif et ses sous-objectifs, ce graphe n'est pas un arbre puisque un objectif peut être un sous-objectif de plus d'un objectif père.
Les objectifs sont organisés selon leurs importances et les relations entre

eux, les sous-objectifs sont généralement plus faciles à gérer et à comprendre. La première étape pour construire un diagramme hiérarchique des objectifs est d'identifier l'objectif global du système qui sera placé tout en haut dans le diagramme, dans notre cas l'objectif global du diagramme est la recherche.

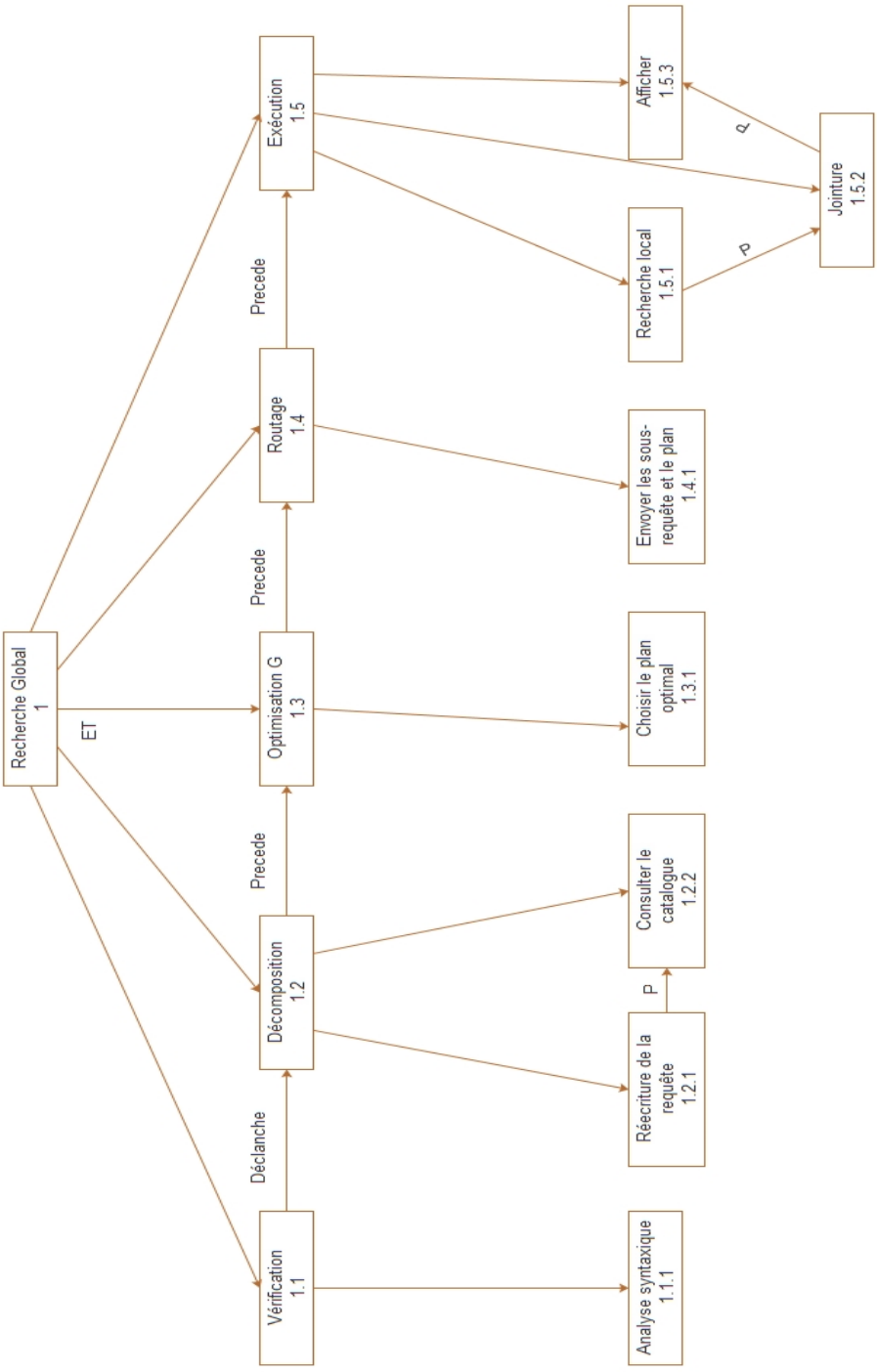


FIGURE 3.1 – Structure hiérarchique des objectifs.

2. **Appliquer les cas d'utilisation**
   La seconde phase de MaSE est l'application des cas d'utilisation qui est une étape cruciale pour traduire les objectifs en des rôles et les tâches associées. Les cas d'utilisation sont tirés des exigences du système et décrivent les évènements qui définissent le comportement du système, comment le système devrait réagir. Pour aider à déterminer les communications dans les SMA, les cas d'utilisation sont convertis en des diagrammes de séquences. Les diagrammes de séquences de MaSE sont similaire au diagramme de séquence d'UML sauf qu'ils sont utilisé pour représenter les évènements entre les rôles et de définir les communications entre les agents qui exécuteront les rôles.

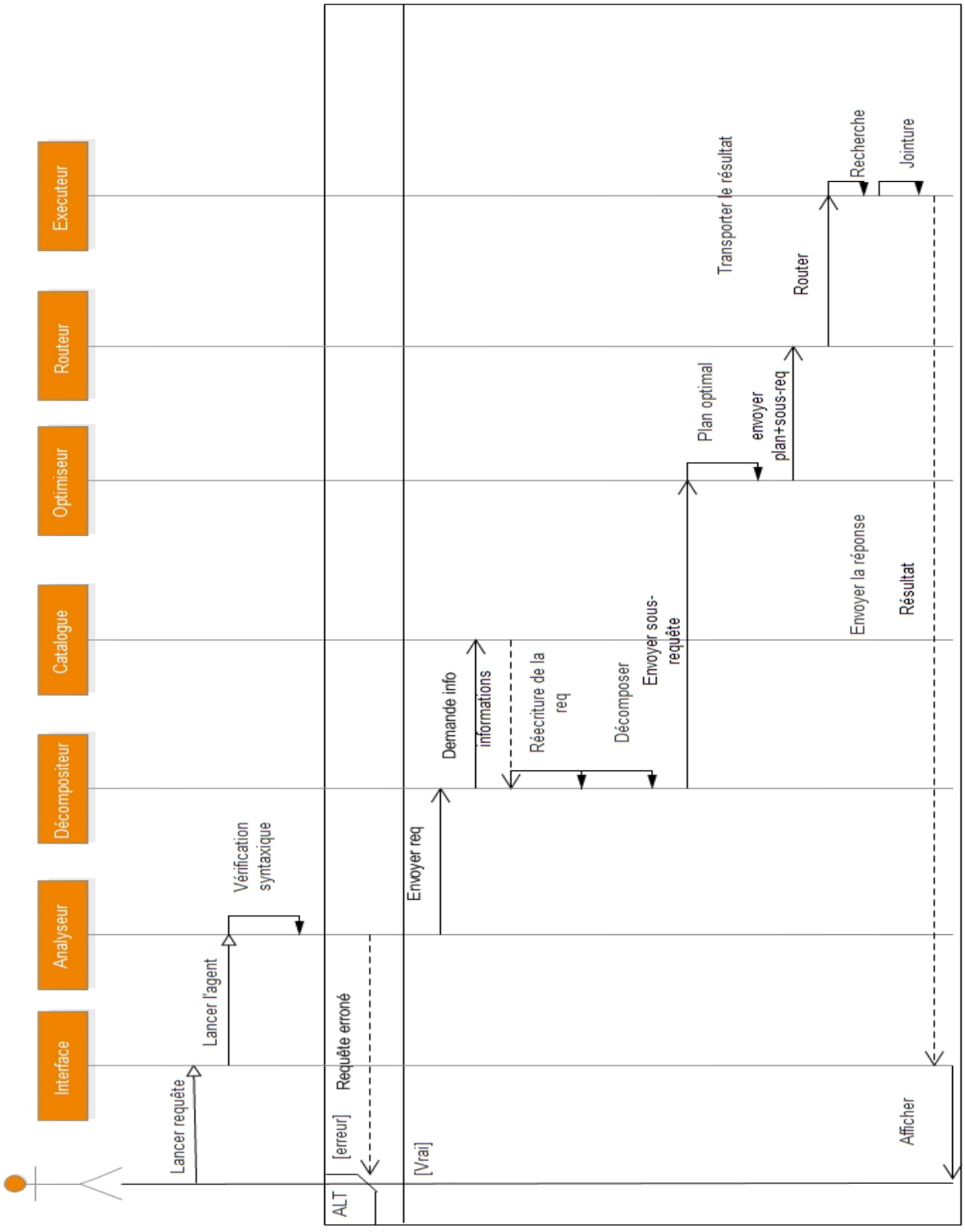


FIGURE 3.2 – Diagramme de séquence.

Dans la **figure** 3.2 on voit un système qui recherche de l'information dans des bases de données distribuées,

- L'utilisateur du système lance une requête SQL,
- le système analysera automatiquement la syntaxe de la requête, si elle contient des erreurs, un message s'affichera,
- si elle a été écrite correctement le système passera à l'étape suivante qui est la réécriture de la requête en un arbre algébrique puis la décomposition en sous-requête par rapport à leurs localisations (la localisation se fait depuis le catalogue SGBD) ,en plus des données de localisation le décompositeur demandera les statistiques de chaque table,
- après la décomposition l'optimiseur à l'aide des statistiques choisira le plan le plus optimal par rapport au transfert,
- le routeur transportera les sous-requêtes vers chaque sites correspondants,
- Sur chaque site l'exécuteur fera la recherche locale puis la jointure des résultats et transmettra le résultat vers l'interface qui l'affichera.

3. **Raffiner les rôles**
   La troisième phase est de transformer les objectifs du diagramme hiérarchique et les diagrammes de séquences en des rôles et leurs tâches associées. Les rôles constituent le fondement des classes d'agents et correspondent aux objectifs du système durant la phase de conception
   Le cas général des transformations des objectifs en rôles, est de les transformer un par un. Il existe des situations où il est plus utile d'avoir un rôle étant responsable de plusieurs objectifs.

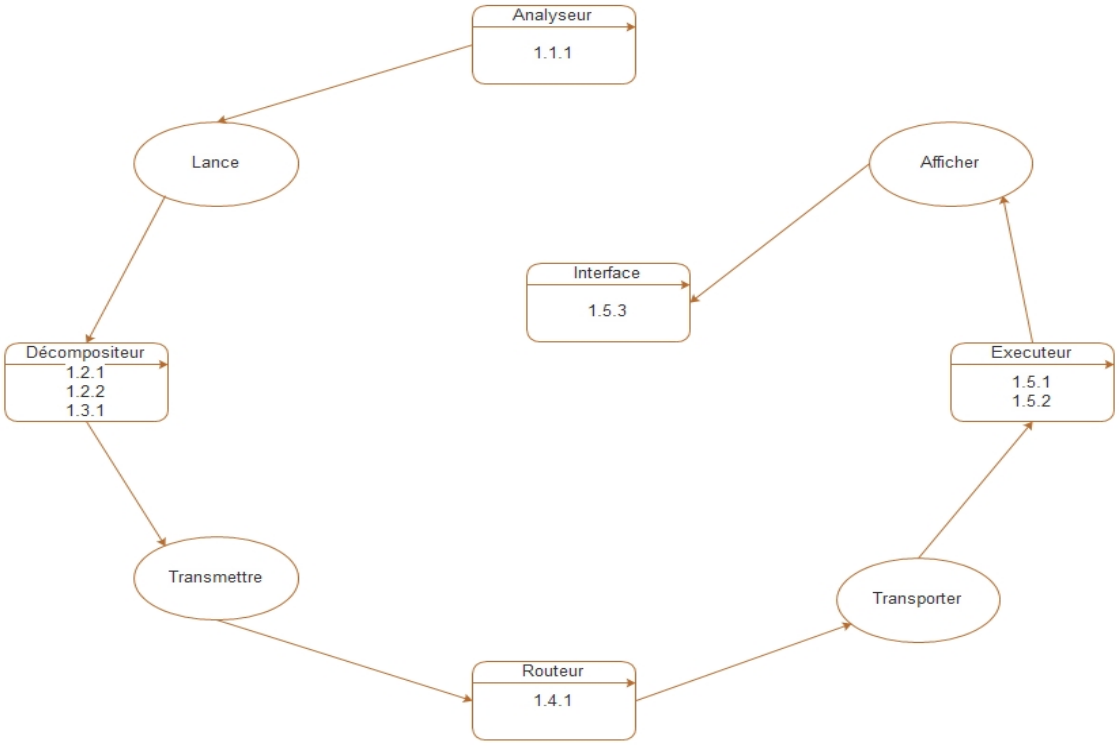


FIGURE 3.3 – Model des rôles.

La figure montre le modèle de rôles obtenu. Chaque feuille dans le diagramme d'objectifs (**Figure** 3.1) doit être associée à un rôle qui peut le réaliser, donc chaque rôle doit atteindre au moins un objectif, généralement un rôle peut réaliser plusieurs objectifs. Il existe cinq(05) rôles :

(a) **Analyseur** : Ce rôle vérifie la syntaxe de la requête si elle est correcte il lance l'agent décompositeur.

(b) **Décompositeur** : Ce rôle réécrit la requête en arbre algébrique, consulte le catalogue pour extraire les informations sur la localisations des tables et les statistique et puis fait le choix du plan le plus optimal par rapport au coût de transfert (ce rôle regroupe deux objectifs la décomposition et l'optimisation).

(c) **Routeur** : Ce rôle transporte le plan et les sous-requêtes vers les autres sites et retourne le résultat vers le site demandeur.

(d) **Exécuteur** : Ce rôle exécute la requête sur le site, fait la recherche local, puis envoi chaque résultat vers un autre site suivant le plan.

(e) **Interface** : ce rôle affiche initialise le système , lance l'agent analyseur et affiche le résultat final.

4. **Le model des tâches conccurente**
   Après l'identification des rôles et tâches, le développeur capture le comportement des rôles en définissant les détails de chaque tâches. Un rôle peut consister en plusieurs tâches qui, lorsqu'elles sont pris ensemble définissent le comportement de ce rôle. Les tâches concurrente sont définies dans un modèle de tâche concurrente et sont spécifiées comme étant des automates d'états, qui consistent en des états et des transitions. Dans notre cas on n'a pas de concurrence.
5. **Création de la classe d'agent**
   Dans cette phase les classes sont identifiées à partir des rôles définis précédemment. Cette étape produit un diagramme de classe d'agents, qui décrit l'organisation générale du système et les conversations entre les agents. Chaque agent est défini par les rôles qu'il va jouer et par les communications auxquelles il va participer.

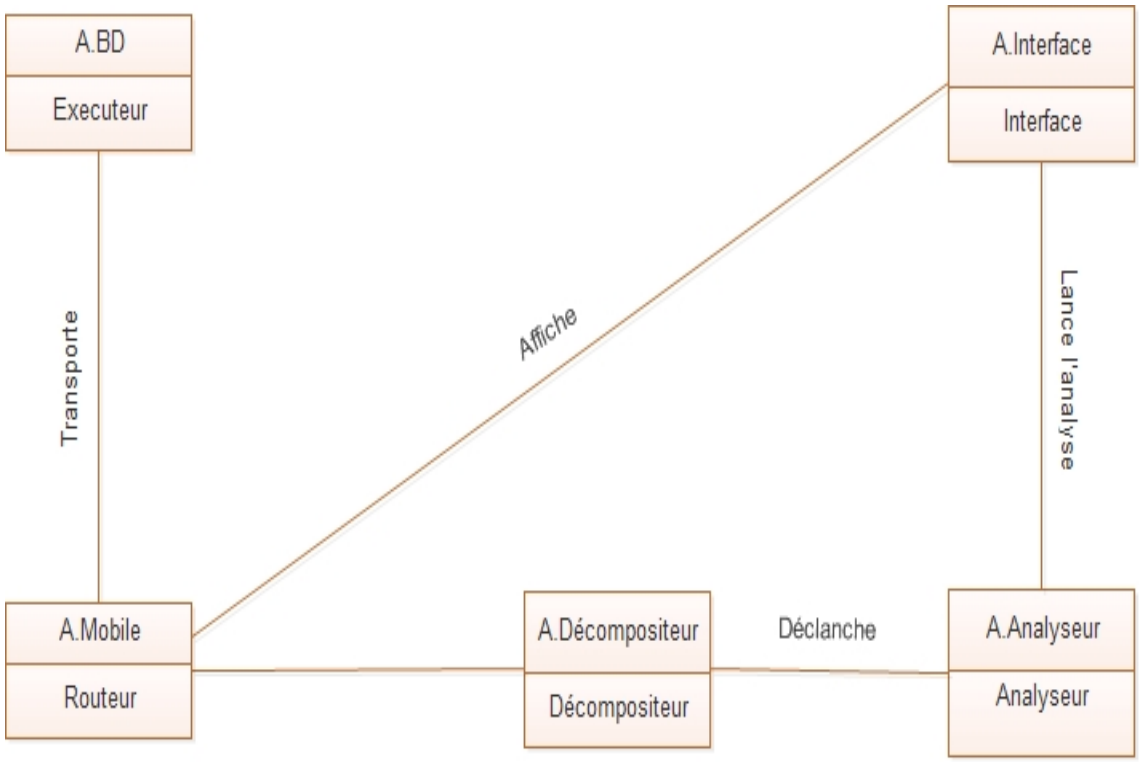


FIGURE 3.4 – Diagramme de classe d'agents .

La **figure** 3.4 montre un modèle général d'agents, nous y trouvons :
- **Agent Interface** qui joue le rôle de l'interface,
- **Agent Analyseur** qui joue le rôles d'analyseur ,
- **Agent Décompositeur** qui joue le rôle du décompositeur des requête,
- **Agent Mobile** qui joue le rôle du routeur,
- et puis l'**Agent BD** qui joue le rôle de l'exécuteur (qui exécute localement la requête).

6. **Construction des conversations**
   L'objectif du modèle de protocole est de définir les détails des protocoles identifiés dans les modèles de rôles et les modèles de classes d'agents, le modèle de protocole défini les protocoles en termes de messages envoyés entre les agents, ou entre les agents et les acteurs externes.

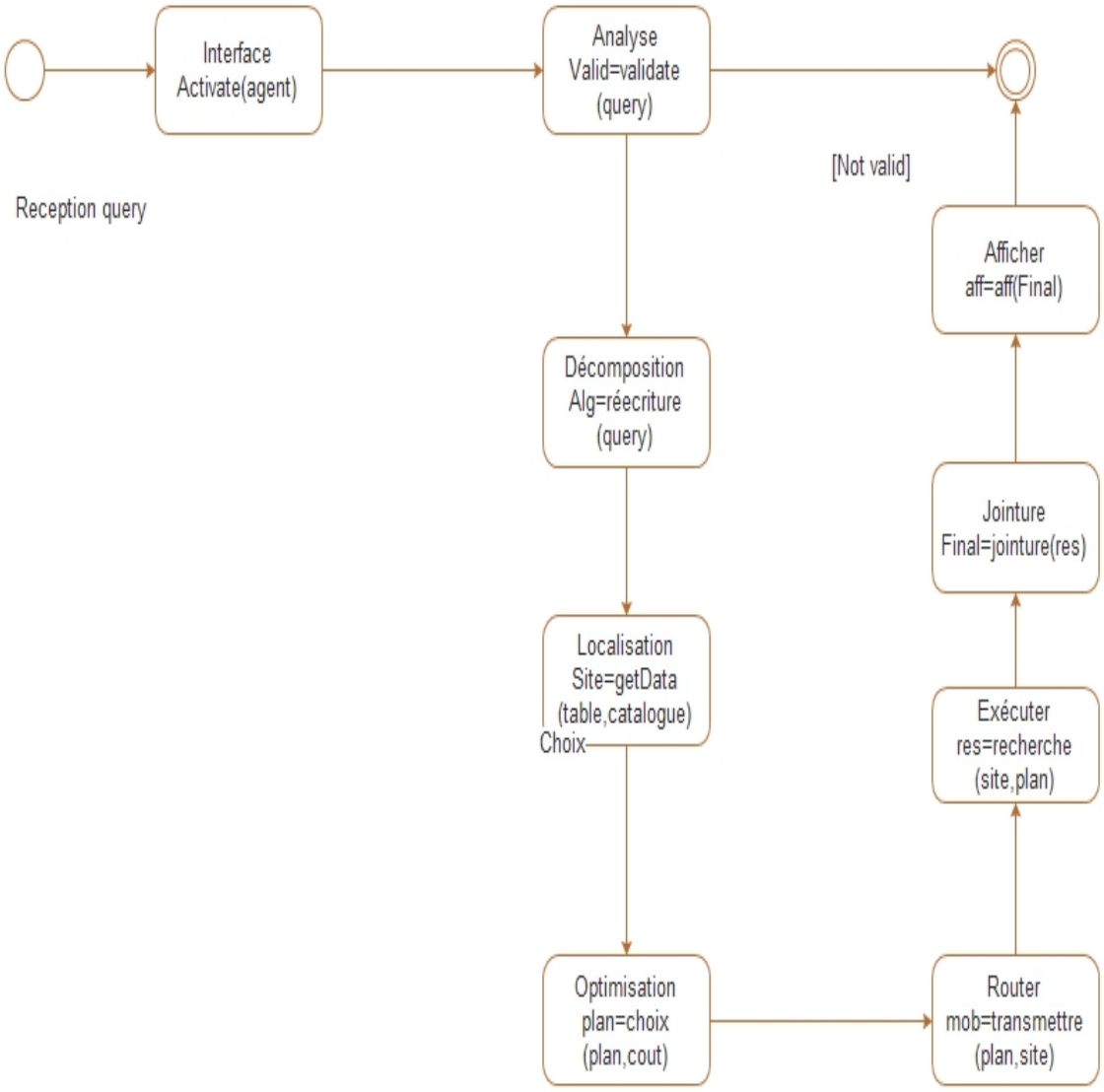


FIGURE 3.5 – Diagramme des protocoles.

Le système(l'interface) reçoit une requête, l'agent va directement lancer l'agent analyse pour analyser la requête syntaxiquement si elle est valide ou non, si l'agent trouve une erreur il va envoyer un message à l'utilisateur sinon il va réécrire la requête en arbre algébrique puis à l'aide du catalogue la décomposera en sous-requête par rapport à la localisation des sites et choisira le plan optimal par rapport au coût de transfert calculer à l'aide des statistiques, après la localisation l'agent

mobile transportera les sous-requêtes vers les sites. L'exécution se fera sur chaque site ayant reçu la requête correspondante, l'agent fera la recherche puis enverra le résultat de chaque vers un autre site suivant le plan pour faire la jointure ; l'agent mobile devra attendre l'exécution des requêtes sur chaque SGBD distribué et puis transporté le résultat vers l'agent interface qui l'affichera.

7. **Assemblage des agents**
   Cette étape défini l'architecture des agents et des composants (qui consiste en un ensemble d'attributs et de méthodes).L'architecture et la définition interne des composants doivent être consistant avec les conversations défini auparavant.

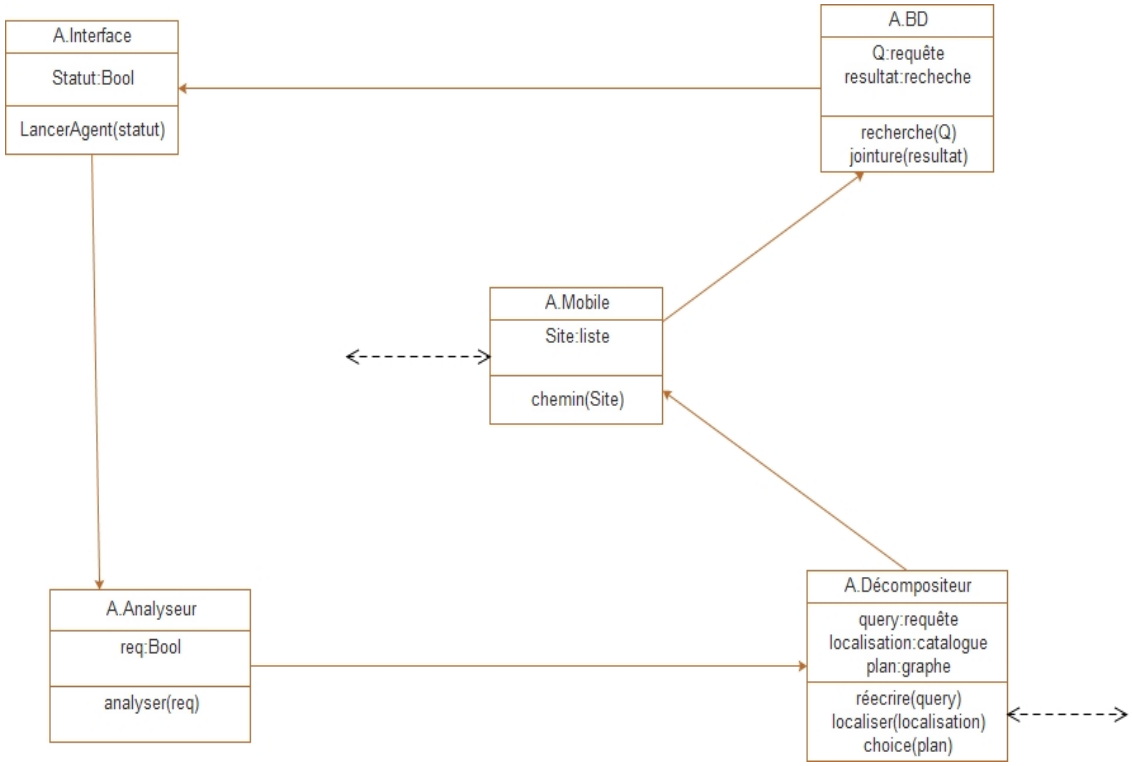


FIGURE 3.6 – Architecture interne des classes.

Les composants sont joints avec les connecteurs d'agents interne ou externe. Les connecteurs interne (flèche continue) définissent la visibilité entre les composants alors que les connecteurs externe (flèche discontinue) définissent les connexions avec les ressources externes tel que les

capteurs, les effecteurs, les bases de données et les data stores. Dans notre cas la flèche discontinu avec l'agent analyseur défini la connexion avec le catalogue et l'autre flèche discontinu liée avec l'agent mobile défini la connexion avec la base de données.

8. **La conception du système**
   La dernière phase de la méthode MaSE prends les classes d'agents et les instancies les agents réels. Elle utilise un diagramme de déploiement pour montrer le nombre, les types, et l'emplacement des agents dans un système.

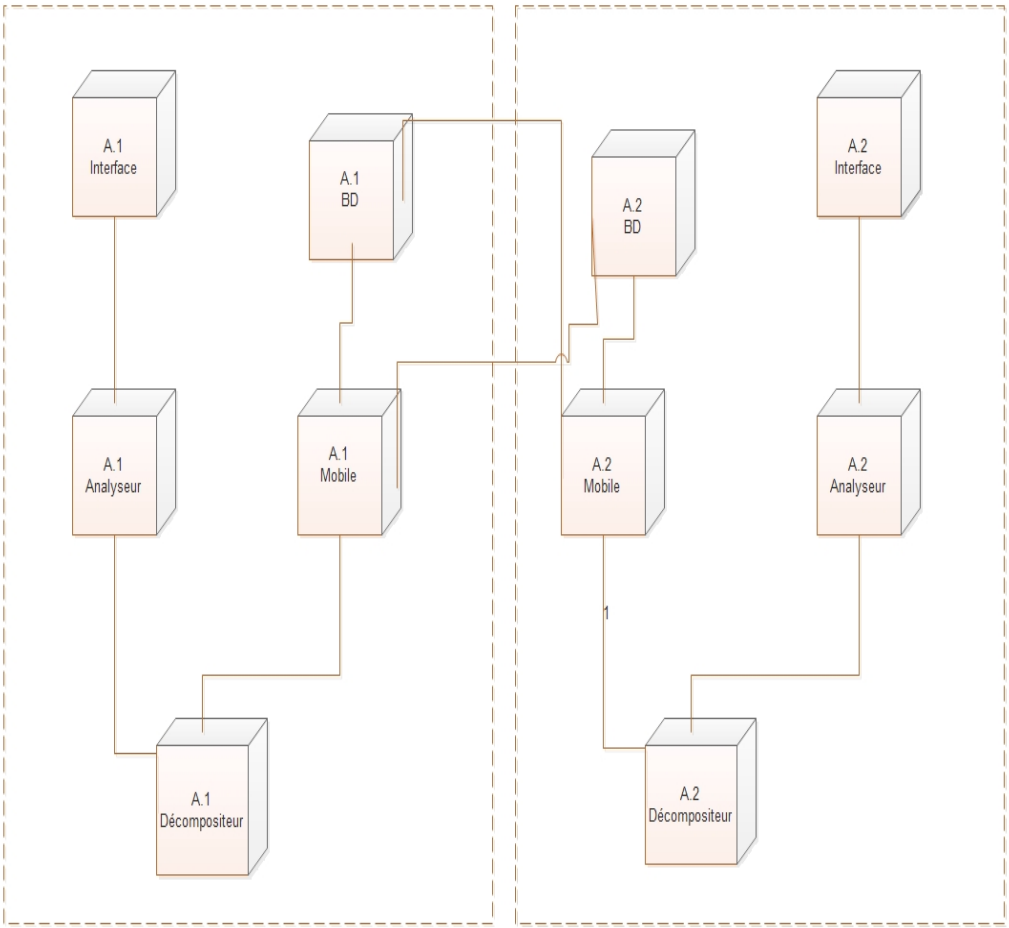


FIGURE 3.7 – Diagramme de déploiement.

Les boites à trois dimensions sont des agents, et les lignes de connexion représentent des conversations entre eux. Les agents sont nommés après leur classe d'agents. Le rectangle en pointillé indique que les agents sont logés sur la même plate-forme.

3.3 Conclusion

Nous avons présenté une démarche de conception à base d'agent en utilisant MaSE. Cette approche est simple et en même temps assez détaillée. La force de cette méthode est que le concepteur peut faire des modifications après la conception de l'organisation du système, et ainsi générer des configurations du système varié.
L'implémentation de ce système sera présentée dans le chapitre suivant, Nous utilisons Java comme langage de programmation et JADE comme plateforme d'implémentation.

Chapitre 4

Implémentation de l'approche

4.1 Introduction

Les langages de programmations des systèmes multi-agents, les plate-formes et les outils de développement sont des éléments importants qui peuvent affecter la diffusion et l'utilisation des technologies de l'agent à travers différents domaines d'applications. En fait, le succès des systèmes multi-agents dépend largement de la disponibilité de la technologie appropriée (càd-les langages de programmation, les bibliothèques et les outils de développement) qui permet la mise en œuvre relativement simple des concepts et des techniques qui forment la base des systèmes multi-agents.

4.2 Les outils utilisés

4.2.1 SQL server

SQL Server est un système de gestion de base de données relationnelles, plus communément appelé SGBDR. Edité par Microsoft, la première version date de 1989 et a fortement évolué pour s'adapter aux besoins les plus exigeants. Différentes versions existent, Standard, Enterprise, Datacenter, chacune avec ses spécificités et limitations. Nous choisissons ici la version gratuite 2008 R2 « Express ».

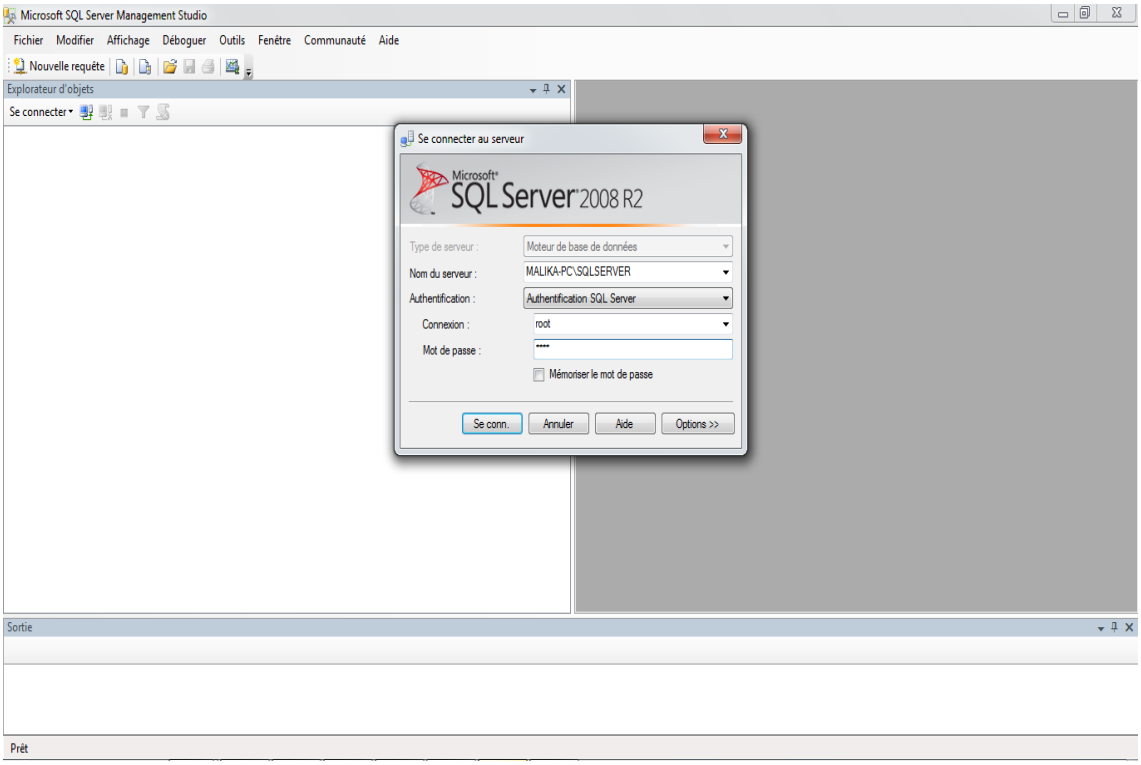


FIGURE 4.1 – SQL Server 2008 R2.

4.2.1.1 La distribution

La distribution en SQL server n'existe pas, mais on peut lier des serveurs pour utiliser les données provenant d'autres sources. Une fois qu'un serveur lié a été créé, il est possible d'exécuter des requêtes distribuées sur ce serveur, et les requêtes peuvent joindre des tables de plusieurs sources de données. Pour lier des serveurs, il faut que les machines soient connéctées à un réseau, pour cela on a configuré un réseau sans fil ad hoc et connecté les deux machines. La liaison se fait en éxecutant la requête suivante sur les machines connéctée :
$EXECsp\_addlinkedserver@server =' servername'$
$EXECsp\_addlinkedsrvlogin'servername',' false', NULL,' username',' password'$
et puis en recherchant les serveurs sur le réseau. Après la liaison, on pourra lancer toutes les requêtes.

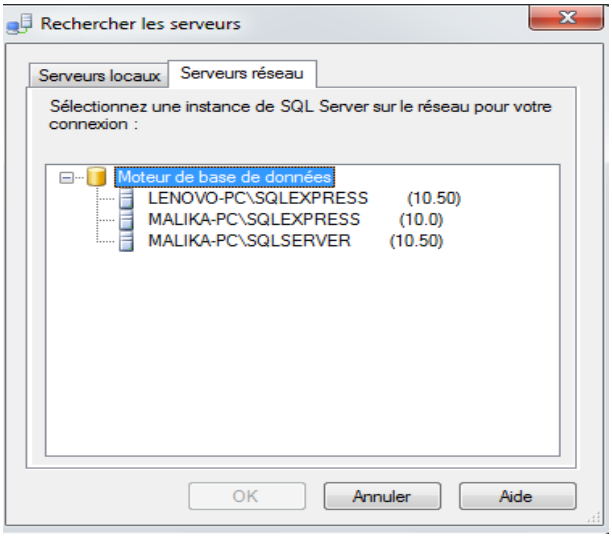


FIGURE 4.2 – La recherche d'un serveur sur le réseau.

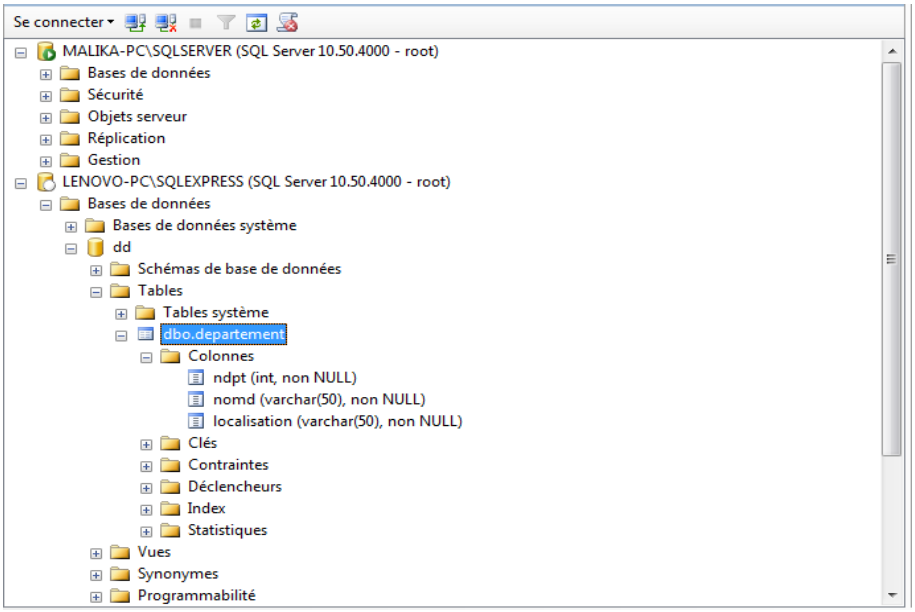


FIGURE 4.3 – Liaison entre deux serveurs.

4.2.2 JADE

JADE est un logiciel-médiateur qui permet une implémentation flexible des Systèmes Multi-Agents communiquant grâce à un transfert efficace des messages ACL (Agent Communication Language), conformes aux spécifications de la FIPA. JADE est écrit en Java, supporte la mobilité, évolue rapidement et fait partie aujourd'hui des rares plateformes multi-agents qui offrent la possibilité d'intégration des services Web.
JADE contient :

- **Un runtime Environment :** l'environnement ou les agents peuvent vivre. Ce runtime environment doit être activé pour pouvoir lancer les agents.
- **Une librairie de classes :** que les développeurs utilisent pour écrire leurs agents
- **Une suite d'outils graphiques :** qui facilitent la gestion et la supervision de la plateforme des agents.

Chaque instance du JADE est appelée conteneur " container ", et peut contenir plusieurs agents. Un ensemble de conteneurs constituent une plateforme. Chaque plateforme doit contenir un conteneur spécial appelé main-container et tous les autres conteneurs s'enregistrent auprès de celui-là dés leur lancement.
Un main-container se distingue des autres " simples " conteneurs par une autre chose ; il contient toujours deux agents spéciaux appelés AMS et DF qui sont lancés automatiquement au lancement du main-container

- **RMA** (Remote Agent Management) JADE offre une interface graphique pour l'administration de la plate-forme par le biais de son agent RMA.
- **AMS** (Agent Management System) qui fournit le service de nommage (pour assurer par exemple que chaque agent possède un identifiant unique dans la plateforme) et qui représente l'autorité de la plateforme (par exemple il est possible de créer/arrêter des agents en envoyant des requêtes à l'AMS).
- **DF** (Directory Facilitator) qui fournit un système de pages jaunes qui permet aux agents de retrouver les agents fournisseurs de services.

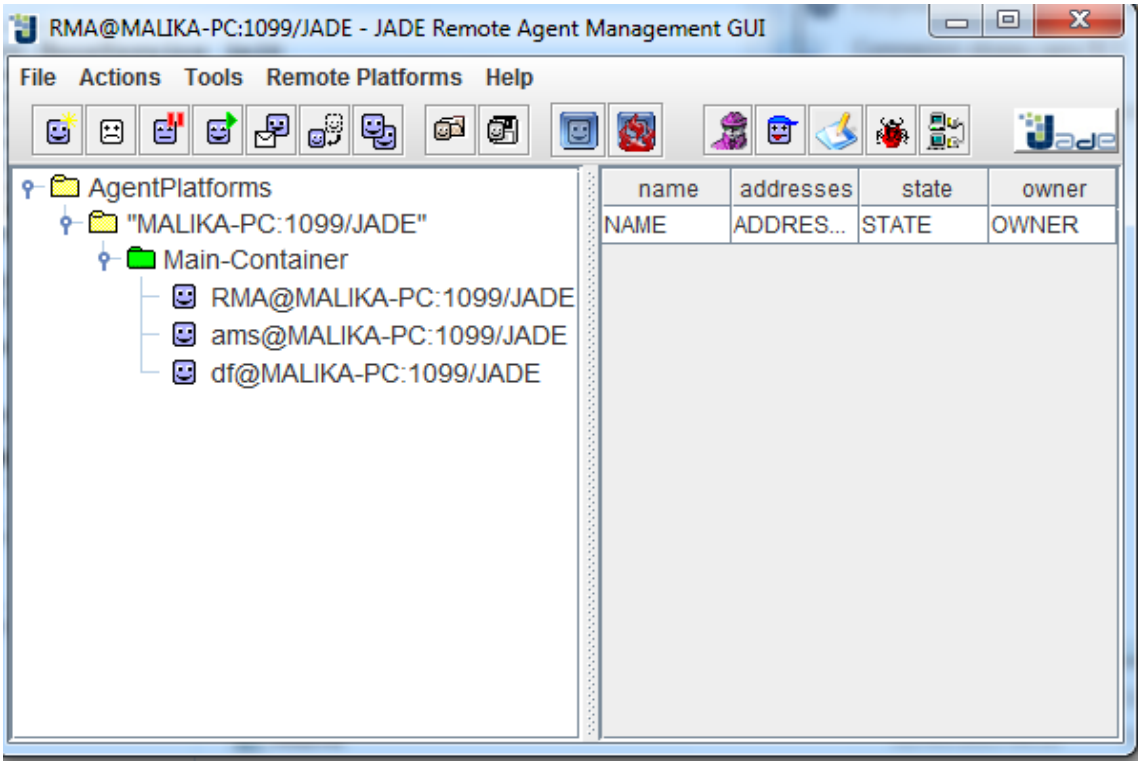


FIGURE 4.4 – La plateforme JADE

4.2.2.1 La configuration

– Pour lancer la plate-forme JADE, la variable d'environnement CLASSPATH doit être définie en incluant tous les fichiers *.jar :
C :\> set JADE_HOME=c :\jade
C :\> set CLASSPATH=%JADE_HOME%\lib \jade.jar ; %JADE_HOME%
\lib \jadeTools.jar ;%JADE_HOME%\lib\http.jar ; %JADE_HOME%\lib\iiop.jar ;
%JADE_HOME% \lib\commons-codec\commons-codec-1.3.jar ;%JADE_HOME%\classes

après cela JADE pourra être lancée à partir de l'invité de commande avec la commande ci-dessous :
`java jade.Boot -gui`
– Le pilote **JDBC** Java Database Connectivity est une API fournie avec Java. Le pilote doit aussi être ajouté dans le CLASSPATH.
– Ant est un outils dont le but est d'accélérer la construction et le déploiement des projets Java. Avant de lancer Ant, le chemin vers le ant/bin

devra être ajouté au PATH, ajouter une autre variable ANT_HOME avec le chemin d'installation d'Ant.

set ANT_HOME=c :\ant
set PATH=%PATH%;%ANT_HOME% \bin

– **IPMS** Inter-Platform Mobility Service créé pour fournir la mobilité de plate-forme en plate-forme pour les agents JADE. IPMS n'est pas intégrée dans la plate-forme mais doit être installée comme un plus. Lors de l'installation le package devra être décompressé dans le fichier JADE ou la commande *ant lib* est utilisée pour créer les fichiers Jar contenant toutes les classes compilées.IPMS devra aussi être inclut dans le CLASSPATH. Pour utiliser le service, il doit être spécifié explicitement dans la commande :

java jade.Boot -services jade.core.mobility.AgentMobilityService ;jade.core.migration.
InterPlatformMobilityService

et la plate-forme s'affichera comme suite avec l'agent mobility manager (amm) qui gère l'ensemble du processus de migration, chaque fois qu'un agent veut migrer entre deux endroits l'AMM correspondant établit un dialogue qui se termine avec le mouvement de l'agent.

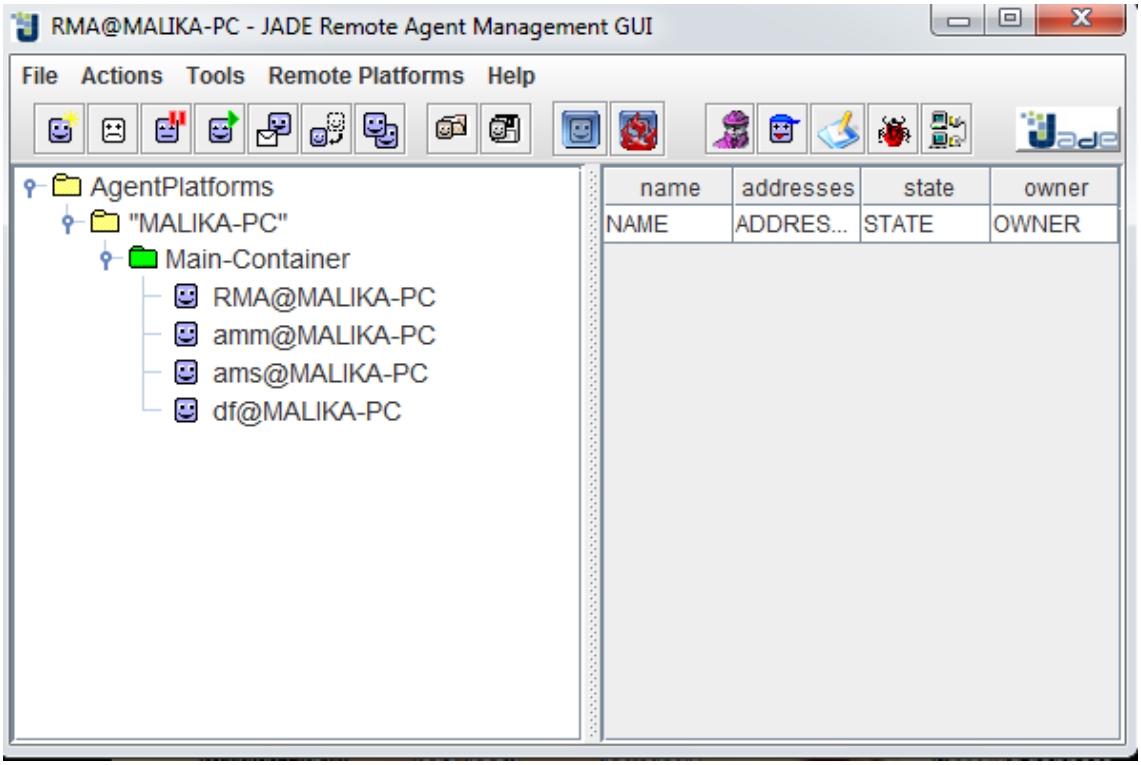


FIGURE 4.5 – Agent mobility manager

4.2.3 Description du système

4.2.3.1 Approche implementée

Nous décrivons dans cette partie l'approche implémentée, on a proposé une structure de pilotage composée d'un agent qu'on a appelé agent interface, qui se lancera dès le démarrage du système, cet agent va afficher l'interface pour l'utilisateur qui va taper la requête SQL et l'envoie, dès son envoie la requête sera analysée syntaxiquement, si la requête est erronée un message d'erreur s'affichera à l'utilisateur pour l'informer, si elle est correcte, l'agent vérifie si les données demandées existent sur le site demandeur ou bien sur un autre site, si elles se trouvent sur le même site elles s'exécutera et l'agent affichera le résultat, sinon il lancera l'agent mobile vers le site qui contient les données pour récupérer la table et l'insérer dans une table temporaire dans le site demandeur puis exécute la requête. Si la requête demande une jointure la même chose arrivera, l'agent interface lancera l'agent mobile pour récupérer la table, l'insérer dans une table temporaire et puis faire la jointure,

la **Figure** 4.6 résume notre approche. La semi-jointure n'est pas utilisée dans cette approche ni la fragmentation et la réplication.

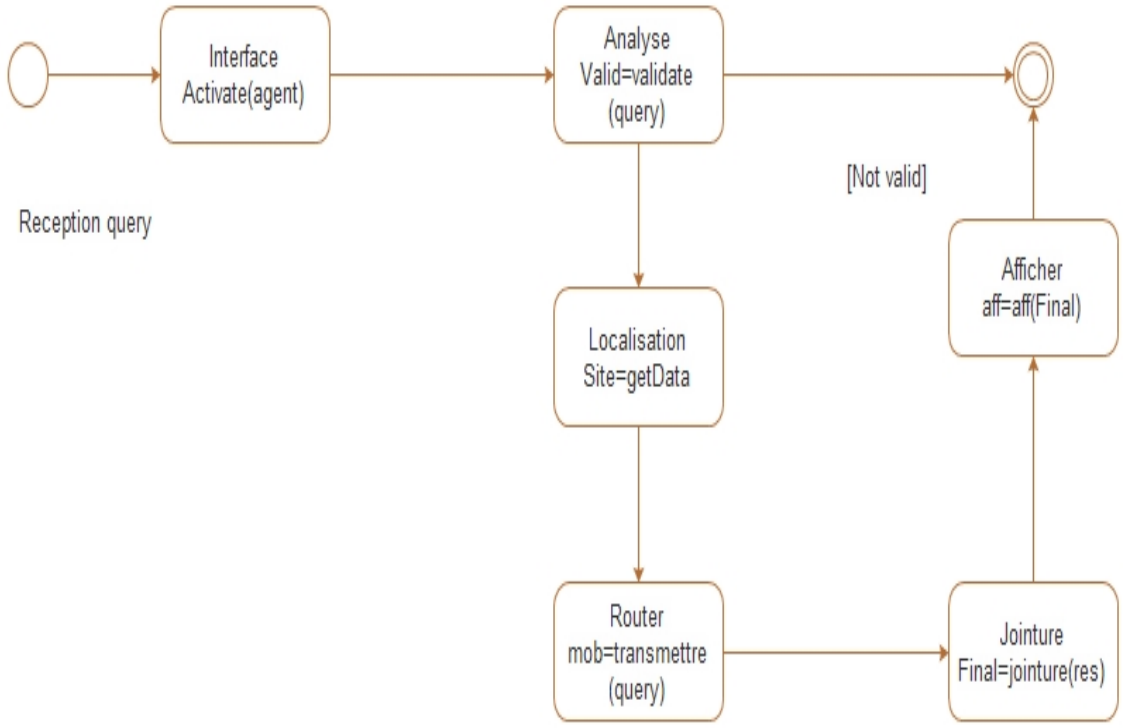


FIGURE 4.6 – Diagramme de protocole finale

4.2.3.2 Exemple d'exécution

Echantillons des tables existantes :
Notre base de données contient deux tables, **employée***(id ,nom,prenom,ddn,adr,ndpt)* qui contient 20.000 enregistrements et la table **departement**(*ndpt,nomd,localisation*) qui contient 5.000 enregistrements , chaque table est située sur un site différent Les données ont été géneré avec une demos du logiciel payant **"Red-Gate SQL data generator"**.

Machine utilisé :
Voici les performances des deux machines :

Machine1 Lenovo		Machine2 Lenovo
Processeur	Inter® Core™ i5-3230M CPU @2.60GHz 2.60GHz	Intel® Core™ Duo CPU T6570 @2.10GHz 2.10GHz
RAM	4.00Go	3.00Go
OS	Windows 7 32bits	Windows 7 32bits

TABLE 4.1 – Performance des machines

L'agent interface qu'on appelera Portail dans notre implémentation se lance avec le lancement du système, il affiche directement l'interface **figure**4.7 qui recevra la requête SQL.

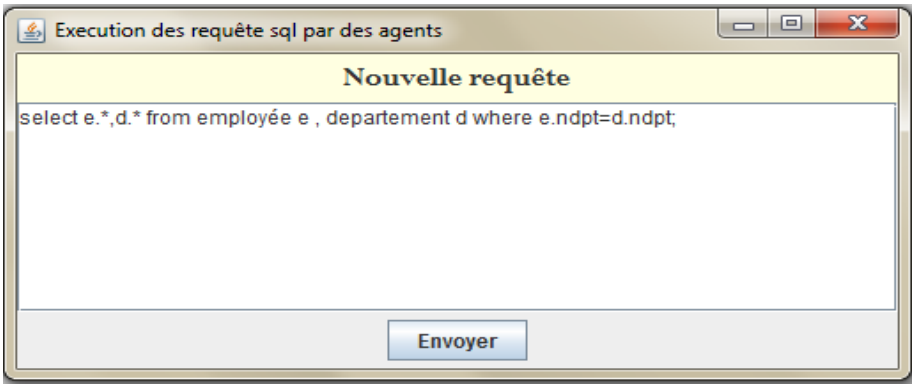


FIGURE 4.7 – L'interface de l'application

L'utilisateur tape la requête et appuis sur le bouton "Envoyer la requête " l'agent Portail lance l'agent analyseur qui va analysé la syntaxe de la requête, si elle est erronée un message s'affichera à l'utilisateur.

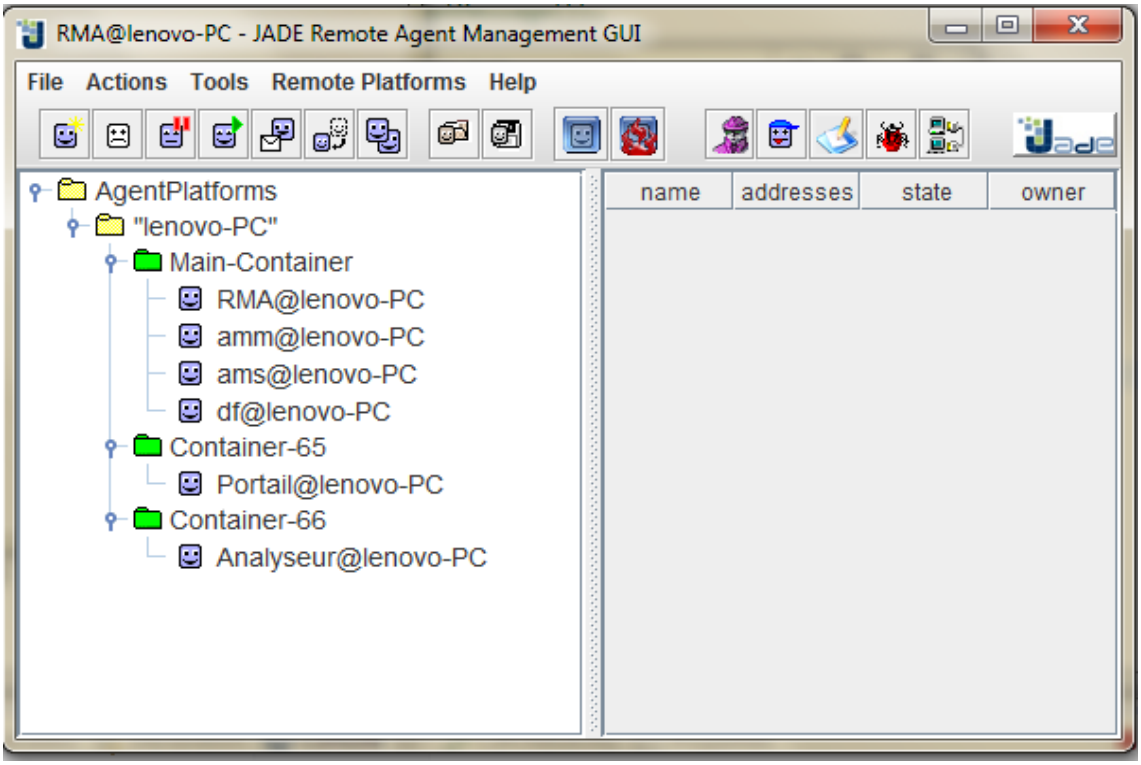


FIGURE 4.8 – Agent analyseur

Après avoir analysée la requête, l'agent vérifie l'emplacement des tables, si elles sont sur le même site, le SGBD exécutera la requête, sinon l'agent lancera l'agent mobile qui transportera une requête demandant toute la table vers le site concernée.

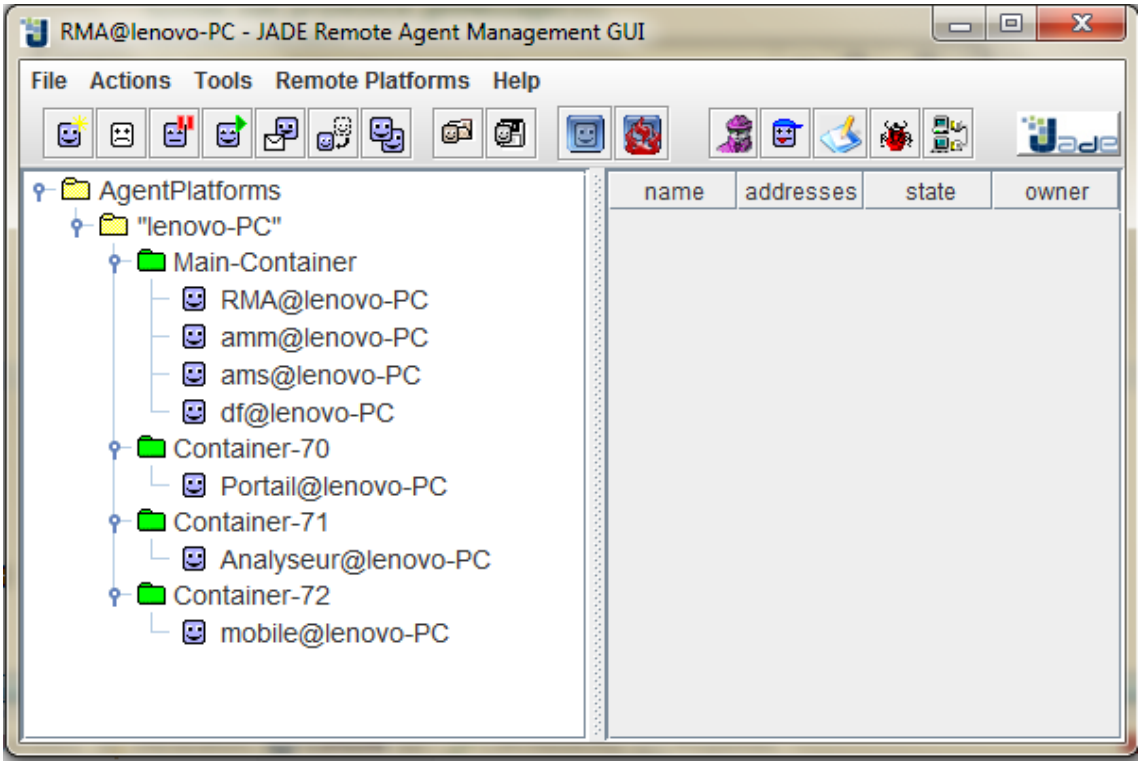


FIGURE 4.9 – Agent Mobile sur le site demandeur

Dès l'arrivée de l'agent mobile **figure**4.10 sur site contenant les données, le SGBD exécutera la requête ramenée et l'enverra vers le site demandeur.

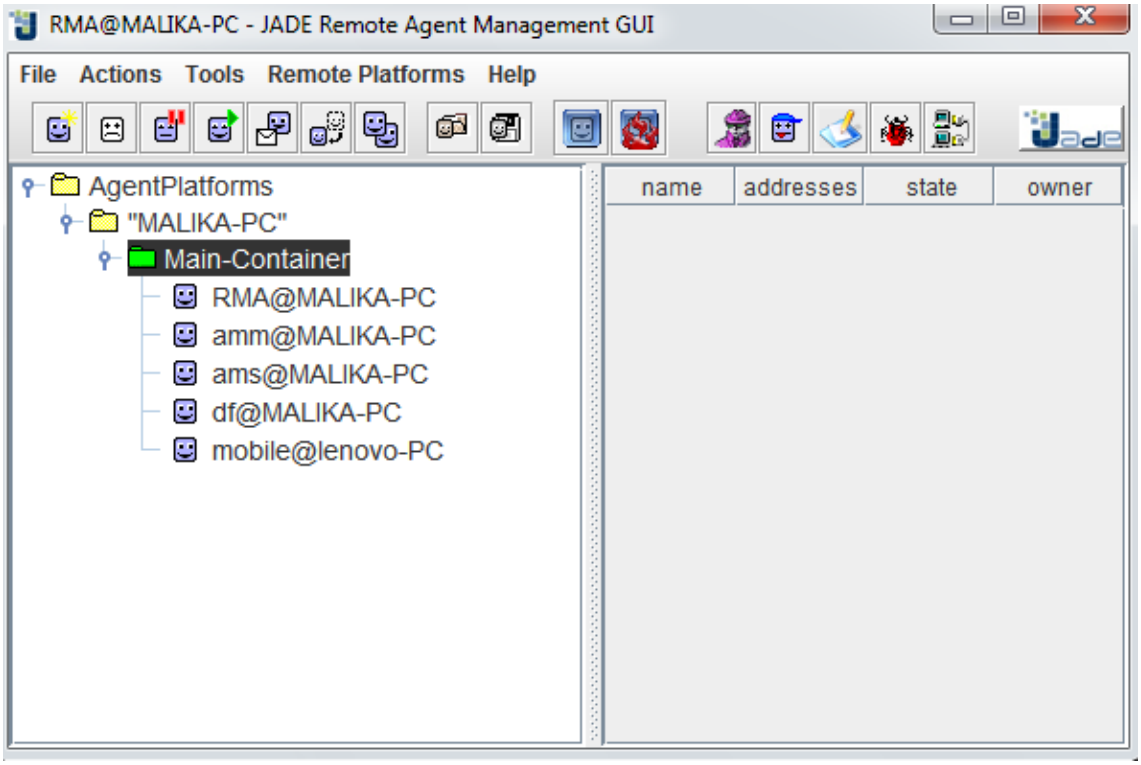


FIGURE 4.10 – L'agent mobile sur le site contenant les données

A l'arrivé de l'agent mobile sur le site demandeur, le SGBD prends le résultat et éxecute la première requête ;

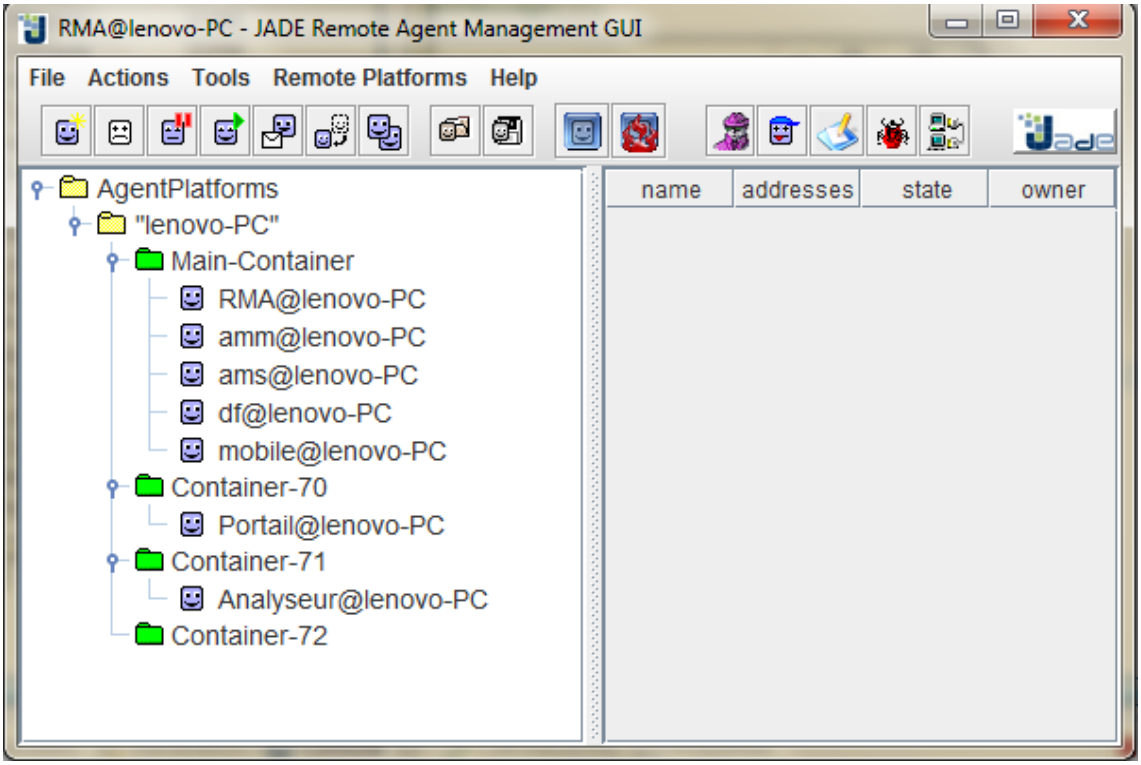
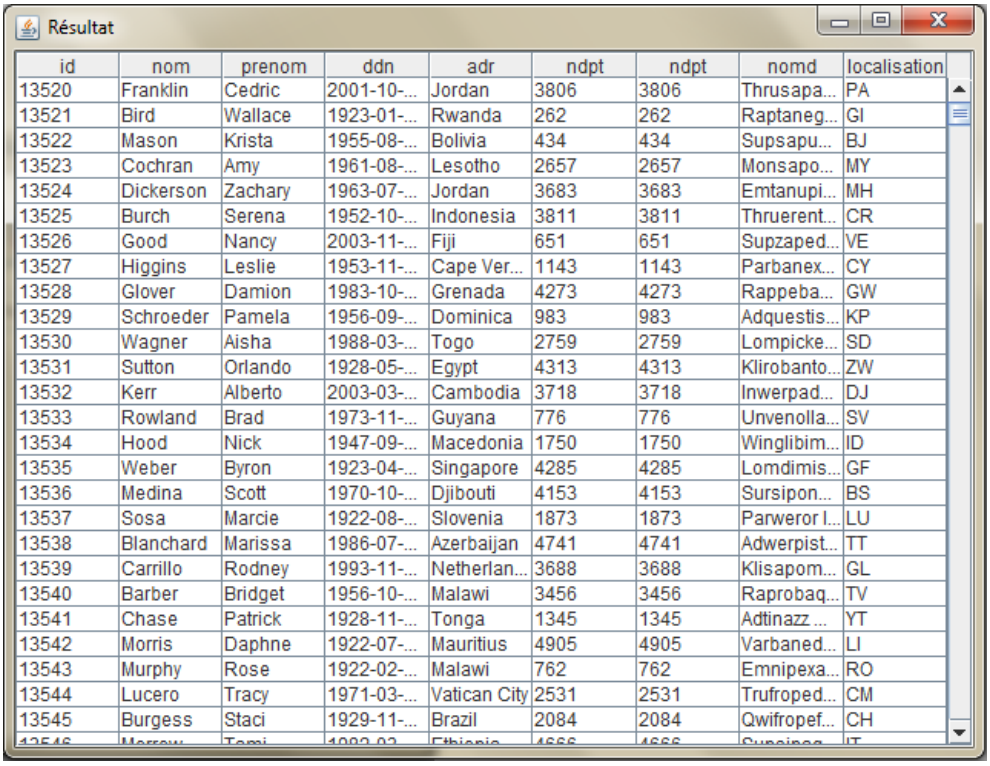


FIGURE 4.11 – L'agent mobile3

Puis l'agent portail affiche le résultat de la requête sur le site demandeur.



id	nom	prenom	ddn	adr	ndpt	ndpt	nomd	localisation
13520	Franklin	Cedric	2001-10-...	Jordan	3806	3806	Thrusapa...	PA
13521	Bird	Wallace	1923-01-...	Rwanda	262	262	Raptaneg...	GI
13522	Mason	Krista	1955-08-...	Bolivia	434	434	Supsapu...	BJ
13523	Cochran	Amy	1961-08-...	Lesotho	2657	2657	Monsapo...	MY
13524	Dickerson	Zachary	1963-07-...	Jordan	3683	3683	Emtanupi...	MH
13525	Burch	Serena	1952-10-...	Indonesia	3811	3811	Thruerent...	CR
13526	Good	Nancy	2003-11-...	Fiji	651	651	Supzaped...	VE
13527	Higgins	Leslie	1953-11-...	Cape Ver...	1143	1143	Parbanex...	CY
13528	Glover	Damion	1983-10-...	Grenada	4273	4273	Rappeba...	GW
13529	Schroeder	Pamela	1956-09-...	Dominica	983	983	Adquestis...	KP
13530	Wagner	Aisha	1988-03-...	Togo	2759	2759	Lompicke...	SD
13531	Sutton	Orlando	1928-05-...	Egypt	4313	4313	Klirobanto...	ZW
13532	Kerr	Alberto	2003-03-...	Cambodia	3718	3718	Inwerpad...	DJ
13533	Rowland	Brad	1973-11-...	Guyana	776	776	Unvenolla...	SV
13534	Hood	Nick	1947-09-...	Macedonia	1750	1750	Winglibim...	ID
13535	Weber	Byron	1923-04-...	Singapore	4285	4285	Lomdimis...	GF
13536	Medina	Scott	1970-10-...	Djibouti	4153	4153	Sursipon...	BS
13537	Sosa	Marcie	1922-08-...	Slovenia	1873	1873	Parweror l...	LU
13538	Blanchard	Marissa	1986-07-...	Azerbaijan	4741	4741	Adwerpist...	TT
13539	Carrillo	Rodney	1993-11-...	Netherlan...	3688	3688	Klisapom...	GL
13540	Barber	Bridget	1956-10-...	Malawi	3456	3456	Raprobaq...	TV
13541	Chase	Patrick	1928-11-...	Tonga	1345	1345	Adtinazz ...	YT
13542	Morris	Daphne	1922-07-...	Mauritius	4905	4905	Varbaned...	LI
13543	Murphy	Rose	1922-02-...	Malawi	762	762	Emnipexa...	RO
13544	Lucero	Tracy	1971-03-...	Vatican City	2531	2531	Trufroped...	CM
13545	Burgess	Staci	1929-11-...	Brazil	2084	2084	Qwifropef...	CH

FIGURE 4.12 – Le resultat

4.2.3.3 Discussion

Afin d'évaluer l'intérêt de la programmation par agents dans l'exécution des requêtes par rapport aux SGBD, nous avons comparé leurs rapidités d'exécution sur deux machines, en utilisant des SGBD homogènes. Les résultats présentés sont des graphes, chaque graphe représente le temps qu'à mit un SGBD par rapport à l'agent à exécuter une requête d'une même table.
Il faut aussi signaler que la charge du réseau est nulle et que les informations circulantes seront uniquement celle de la requête et que la complexité des requêtes n'a pas été prises en comptes.

Le test réalisé a été de calculer le temps que la requête mettait à s'exécuter par un SGBD et un agent qui ramener les données d'un autre site distant, en fonction du nombre d'enregistrement. La requête a été exécutée sur deux machines différentes, le site1 se trouvant sur la Machine1 à lancer la requête vers le site2 distant (la table departement se trouvant sur la Machine2) à l'aide de l'agent mobile qui transporte les données, et a exécuté la requête localement. Voici la courbe obtenue :

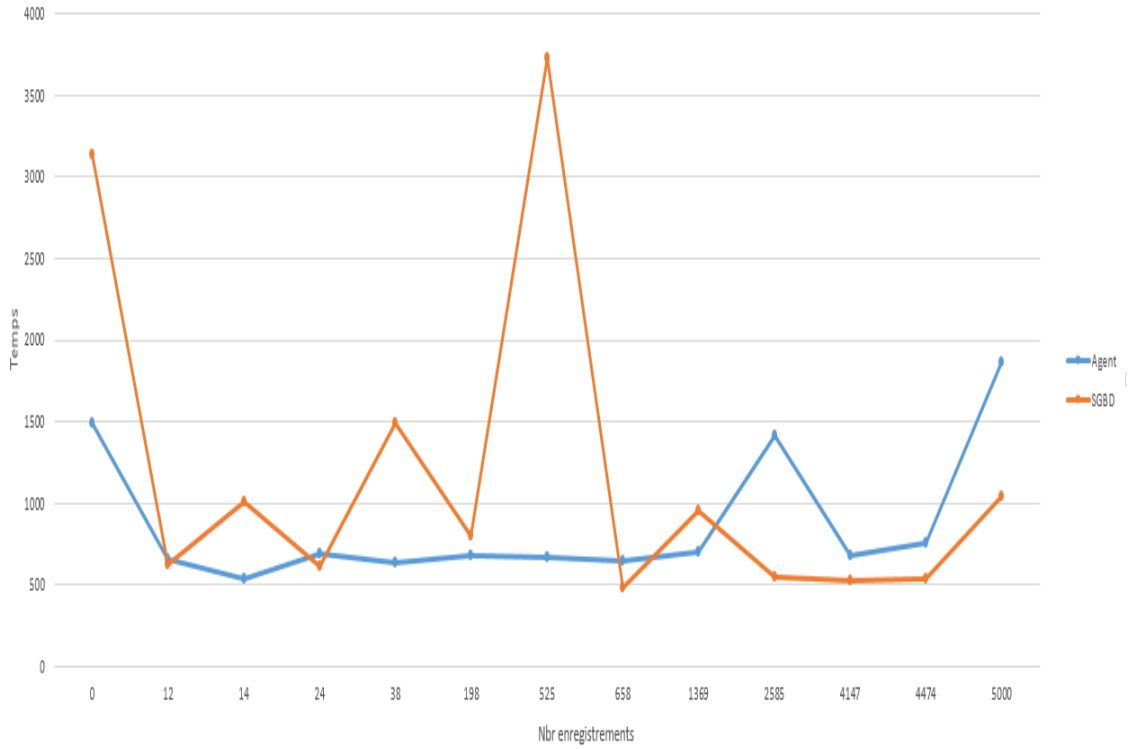


FIGURE 4.13 – Exécution d'une requête de la table departement

La courbe en bleu représentera dans tous les graphes la Machine1 et la courbe orange la Machine2.

Interprétation des résultats

Le résultat est obtenue après le calcule du temps que différente requêtes mettaient pour s'exécuter, en fonction du nombre d'enregistrement ramené. On peut constater que le SGBD mettait plus de temps à exécuter des requêtes que l'agent. On peut donc aussi voir que l'agent est plus performant que le SGBD, on remarque qu'à partir d'un certain nombres enregistrements (a peu près 1400 enregistrements) le temps d'exécution de la requête sur le SGBD diminue et inversement l'agent augmente, on constate que le SGBD devient meilleure, on peut aussi remarquer que certaine requête on prit plus de temps sur sur le site1 (la Machine1) avec le SGBD qu'avec l'agent sur le site2 (la Machine2), ce qui nous laisse déduire que la performance des machines a un rôle dans cette différence de temps.

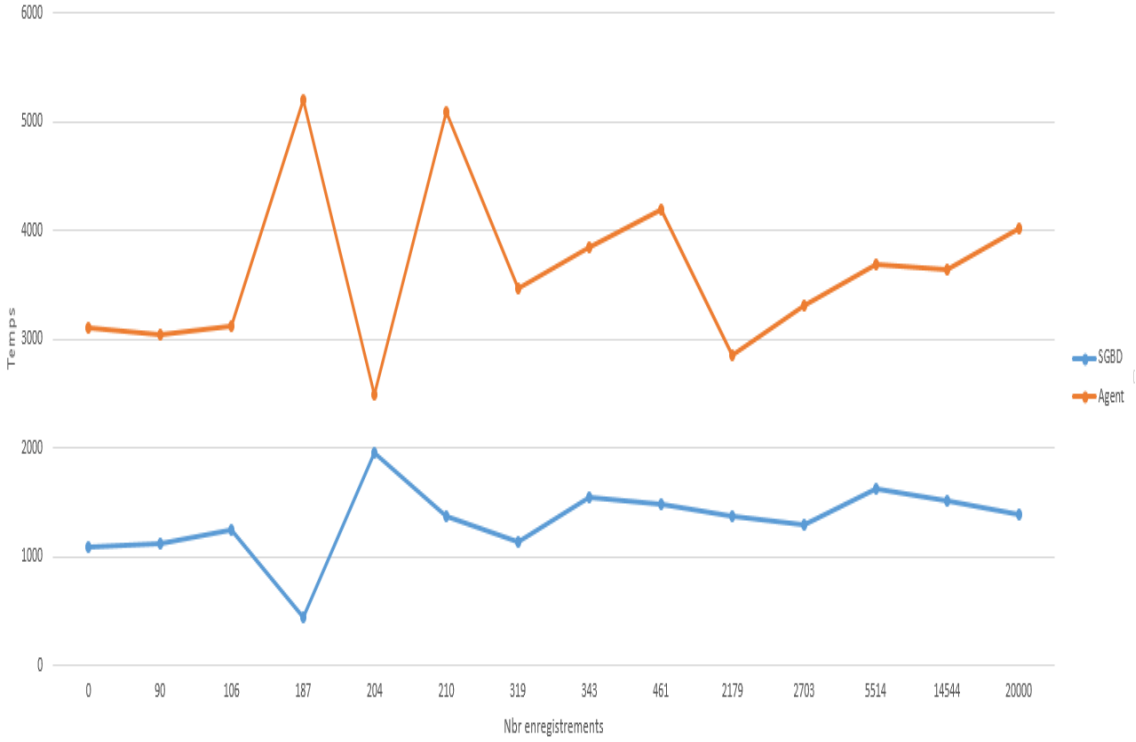


FIGURE 4.14 – Exécution d'une requête de la table employée

Cette courbe représente le temps d'exécution des requêtes par rapport au nombre d'enregistrements sur la table employée qui se trouve sur site2 (la Machine2). On voit que le SGBD est beaucoup plus performant que l'agent ce qui nous laisse en déduire que la performance de la machine joue réellement un rôle très important dans cette étude.

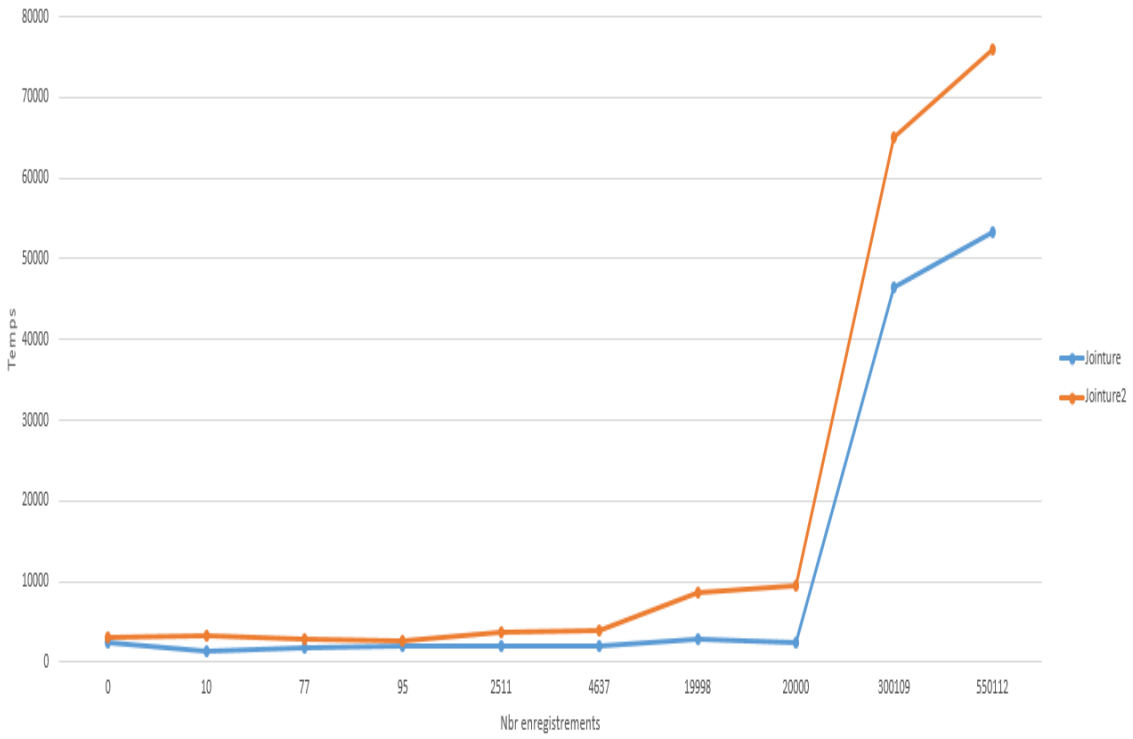


FIGURE 4.15 – Exécution d'une requête de jointure

Cette courbe représente le test de jointure (jointure des deux tables employée et departement) fait sur les deux sites, on voit que les deux courbes sont à peu près identiques sauf que le site1 est toujours meilleurs. On en conclu que la capacité des machines jouent un rôle très important dans l'exécution des requêtes en diminuant le temps.

4.3 Conclusion

Après plusieurs tests sur deux machines, on en a conclu que le temps d'exécution des requêtes par agents est meilleur que par le SGBD dans certaines conditions, où la machine est performante et où le nombre d'enregistrements est limité. Un autre test aurait été important, refaire la même expérience mais en utilisant des requêtes complexe et ajouter d'autres sites , ça nous aurait permis de voir si la complexité à un rôle dans l'exécution des requêtes.

Quatrième partie

Conclusion Générale

Chapitre 5

Conclusion et Perspectives

5.1 Conclusion

Dans ce travail de recherche nous avons présenté une approche orienté agent pour la recherche d'information dans les bases de données distribuées, nous avons proposé une architecture qui permet d'exécuter des requêtes SQL sur des sites distribuées, pour fournir des résultats et de voir si les SMA contribues dans la recherche de l'information. Cette application est composée de plusieurs qui coopère entre eux pour répondre à une requête. Les agents sélectionnent les sites a interrogé et retourne le résultat.

5.2 Perspectives

Nos principales perspectives visent à améliorer la capacité de l'approche et à la rendre plus générique, les recherches menées dans cette thèse ont conduit à proposer des perspective comme suite :

- **La terminaison de l'approche proposée** : qui n'a pas été terminé faute de temps,
- **L'amélioration de l'approche** Le comportement de l'approche en présence de nombreuses et complexes requêtes n'a pas été étudié.
- **Utiliser un algorithme de décomposition des requêtes et d'optimisation** aucun algorithme n'a été utiliser
- **Utiliser la semi-jointure** et essayer d'optimiser encore plus,
- **Amélioré la recherche local.** utiliser un algorithme génénetique qui optimise la recherche dans les BDD
- **Compresser les données transporter.** et ainsi diminuer la surcharge sur le réseau en transportant les données vers d'autres sites.

– **Ajouter le nombre de tables et de fragments sur les sites** amener l'approche a être utiliser dans un environnement réel.
– **Essayer la distance entre les sites**,
– **Utiliser l'approche sur des BD hétérogènes.** Notre approche a été faite sur des BD homogènes

Bibliographie

[1] K. Benfifi. *Une architecture à base d'agent pour le workflow inter-organisationnel lâche.* PhD thesis, 2012.

[2] *Sign of a Revolution on Computer Science and Software Engineerine*, Reading, Massachusetts, 2002. Addison-Wesley.

[3] Jaques Ferber. *Les Systèmes Multi-Agents vers une intelligence collective.* Addison-Wesley, Reading, Massachusetts, deuxi'eme edition, 1995.

[4] Michael Wooldridge. *An Introduction to Multi-Agent Systems. Wiley Editions.* Addison-Wesley, England, 1995.

[5] Gerhard Weiss. *Multiagent Systems.* Paperback, 2000.

[6] B. Fayech. *Régulation des réseaux de transport multimodal : Systèmesmulti-agents et algorithmes évolutionnistes.* PhD thesis, Ecole Centrale de Lille/ Université des Sciences et Technologies de Lille, 2003.

[7] G. Shi. Data integration using agent based mediator-wrapper architecture, tutorial report for agent based software engeneering. Technical report.

[8] et Z.He H. Wang, J. Li. An effective wrapper architecture to heterogeneous data source. 2003.

[9] *Adaptive agents for information gathering from multiple distributed information sources.* Springer, 1999.

[10] *Using mobile crawlers to search the web efficiently.* Springer, 2000.

[11] G. Wiederhold. Mediators in the architecture of future information systems. *Computer*, 1992.

[12] K. Sycara et D. Zeng. 12 th eurpean conference on artificial intelligence. *John Wiley and Sons*, 1996.

[13] C. A. Knoblock et J. L. Ambite. Agents for information gathering. *MIT Press*, 1997.

[14] *Citeseer : An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications.* Springer, 1998.

[15] *"The Do-I-Care Agent : Effective Social Discovery and Filtering on the Web.* Springer, 1997.

[16] *Letizia : An Agent That Assists Web Browsing.* Springer, 1995.

[17] *Mobile Agents for World Wide Web Distributed Database Access.* IEEE, 2000.

[18] L. Boszormenyi H. Kosch, M. Doller. *Content-based indexing and retrieval supported by mobile agent technology.* Prentice Hall Internationa, 2001.

[19] S. Fournier F. Freitas, B. Espinasse. Agathe : une architecture générique à base d'agents et d'ontologies pour la collecte d'information sur domaines restreints du web. *Springer*, 2007.

[20] *Une technique multi-agent pour rechercher des informations réparties.* Springer, 1997.

[21] S. Leriche F. Migeon M. Pantel J. P. Arcangeli, V. Hennebert. principes, installation, utilisation et développement d'applications. Technical report.

[22] N. Troudi M. Côté. Une architecture multiagent pour la recherche sur internet. Technical report, Université Laval. Département d'informatique. Pavillon Pouliot. Ste-Foy, Canada, 1998.

[23] H. H. Hoang S. J. Pelletier, S. Pierre. Modeling a multi-agent system for retrieving information from distributed sources. Technical report.

[24] Zeus.e.a Nwama.H. A toolkit for building distributed multi-agent systems. *Applied Artificial Intelligence Journal*, 1999.

[25] Mesbahi.N. Une approche multi-agents pour la modélisation d'un progiciel de gestion intégrée. 2009.

[26] Ramdani.L. Jade (développement et implémentation de systèemes multi-agents). 2007.

[27] S Tanenbaum, A and L. VAN STEEN. Distributed systems : principles and paradigms. 1999.

[28] V. Bernsteinl, V. Hadzilacos, and N Goodman. Concurrency control and recovery in database systems. 1987.

[29] D. D. Chamberlin R. A. Lorie T. G. Price P. Griffiths Selinger, M. M. Astrahan. Acces path selection in a relational database management system. 1979.

[30] Kossmann.D. The state of the art in distributed query processing. 2000.

[31] I SARR. Routage des transactions dans les bases de données à large echelle. 2010.

[32] C Navas, J and M. Wynblatt. The network is the database : Data management for highly distributed systemse.

[33] N Travers and T. ans Liu T Dang-Ngoc, T. Tgv : An effcient model for xquery evaluation within an interoperable system. int. journal of interoperability in business information systems (ibis). 2006.

[34] L Tianxiao. Proposition d'un cadre générique d'optimisation de requêtes dans les environnements hétérogènes répartis.

[35] M. Gregory. Algorithme génétique pour l'optimisation des requêtes des bases de données distribuées. 1998.