République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider
BISKRA
Faculté des Sciences et des sciences de l'ingénieur
Département de l'informatique

N° d'ordre : ……
Série : ……

Mémoire

En vue d'obtention du diplôme de Magister en informatique
Option : Systèmes d'Information Avancés et Intelligence Artificielle

Une Approche Basée Agent Pour le Data Warehousing

Réalisé par :

GUESBAYA ABDELAZIZ

Membres du jury :

Mr. SAHNOUN Zaidi	Président	Professeur, Université de Constantine
Mr. KAZAR Okba	Rapporteur	Maître de conférences, Université de Biskra
Mr. DJEDI Noureddine	Examinateur	Professeur, Université de Biskra
Mr. CHAOUI Allaoua	Examinateur	Maître de conférences, Université de Constantine
Mr. CHERIF Foudil	Examinateur	Maître de conférences, Université de Biskra

Remerciements

Je tiens à remercier :

Mr. KAZAR Okba d'avoir accepté de m'encadrer et de gérer ce travail et pour ces précieux conseils.

Je remercie également les membres du jury :

Mr. SAHNOUN Zaidi	Professeur, Université de Constantine
Mr. DJEDI Noureddine	Professeur, Université de Biskra
Mr. CHAOUI Allaoua	Maître de conférences, Université de Constantine
Mr. CHERIF Foudil	Maître de conférences, Université de Biskra

D'avoir accepté l'évaluation de ce travail.

J'exprime ma profonde reconnaissance à tous ceux qui m'ont aidé à l'élaboration de ce mémoire de prés ou de loin.

Dédicaces

A toi ma chère mère, pour tes encouragements, ton soutien inconditionnel et surtout ta présence à mes cotés dans les moments les plus difficiles,
Que dieu te bénisse et te garde.

A mon cher père, pour ton soutien constant et inconditionnel, pour tes conseils et tes encouragements.

A mes chères frères, à ma très chère sœur et à toute ma famille.

A tous mes amis et collègues d'études.
A tous ceux qui m'ont aimé et me souhaitent le bonheur et la réussite.

Je vous dédie ce modeste travail
Guesbaya Abdelaziz

Sommaire

Liste Des Figures

Liste des tableaux

Introduction Générale

INTRODUCTION GENERALE

1. Contexte général

Les enjeux des différents secteurs économiques ont en général comme points communs des clients de plus en plus exigeants, des changements de plus en plus rapides et une concurrence de plus en plus forte.[1]

Face à ces enjeux, la prise de bonnes décisions est devenue cruciale pour les dirigeants de l'entreprise, tous les utilisateurs de l'entreprise ayant à rendre des décisions doivent pouvoir accéder aux données de l'entreprise, pouvoir les traiter, et en extraire l'information pertinente afin de prendre les bonnes décisions. Ils se posent alors des questions du type : « Quelle est l'évolution des chiffres d'affaires par type de magasin et par période », ou encore : « Quels sont les résultats des ventes par gamme de produit et par région pour l'année dernière ? » … l'efficacité de cette prise de décision se repose essentiellement sur la mise en place d'informations pertinentes et correctes, et d'outils adéquats. Le problème des entreprises est d'analyser un volume très important de données stockées dans ces systèmes opérationnels, l'exploration et l'exploitation de ces données pour prendre des décisions se montrent difficiles voire impossibles, elles sont réalisées le plus souvent d'une manière imparfaite grâce à des moyens classiques.

Dans ce contexte, les systèmes traditionnels s'avèrent inadaptés pour servir de support à la prise de décision. Ces bases opérationnelles gèrent l'activité quotidienne de l'entreprise (commerciale, production, comptabilité…), mais paraissent peu adapté au décisionnel parce que la réponse est à la fois simple et embarrassante : le système d'information «classique» est un système... qui n'informe pas.[6]

Ces systèmes d'information opérationnels ne peuvent satisfaire ces besoins pour aux moins deux raisons principales:[7]

- les bases de données opérationnelles sont trop complexes pour pouvoir être appréhendées facilement par tout utilisateur.
- le système opérationnel ne peut être interrompu afin de pouvoir répondre à des questions qui nécessitent des calculs importants.

Une nouvelle tendance a commencé à apparaître, afin de résoudre ces problèmes, consistant à développer un système d'information orienté vers la décision et séparer du système d'information opérationnel. Et pour cela, il faut donc garder un historique et restructurer les données de

production, éventuellement récupérer des informations démographiques, géographiques et sociologiques. L'entrepôt de données ou « le Data Warehouse » reste l'élément principal dans l'élaboration d'un tel système.

Le Data Warehouse n'est pas une usine à produire l'information, mais plutôt un moyen de la mettre à la disposition des utilisateurs de manière efficace et organisée. La mise en œuvre d'un Data Warehouse est un processus complexe. Il permet la mise en place d'un outil décisionnel s'appuyant sur les informations pertinentes pour l'entreprise, centrées sur le métier utilisateur.

L'objectif des entrepôts de données est d'être un support pour les applications de traitement analytique en ligne (OLAP, de l'anglais On-Line Analytical Processing). Ce type d'applications se caractérise par une vision multidimensionnelle des données de l'entrepôt et l'analyse de ces données à travers de l'interrogation interactive. Il s'agit essentiellement de pouvoir analyser des indicateurs (ou mesures) d'une activité selon différents axes d'analyse (ou dimensions). Par exemple, la direction de la chaîne de magasins peut analyser les ventes par produit et par magasin et ceci dans le temps.

2. Problématique

L'un des domaines où le nombre d'informations et de connaissances est très important, est le domaine de l'informatique décisionnel et plus précisément l'entreposage de données « Data Warehousing », ce qui génère une certaine difficulté de stockage, de maintenance, de gestion, de sélection et d'intégration des informations adéquates, et aussi une volume des données trop grand et trop varié de telle façon qu'il sera impossible pour l'être humain de suivre ce qui se passe. Le pire, c'est que prochainement les logiciels conventionnels ne seront plus capables de maîtriser la situation, par conséquent une nouvelle structure pour l'entreposage de données s'avère, dès aujourd'hui, nécessaire. Une telle structure facilitera la tâche et fera abstraction des différentes techniques. Ce type d'abstraction est comparable à celui avec lequel les langages de programmation de haut niveau ont débarrassé les programmeurs de tous les problèmes de bas niveau (registres et appareils).

Etant donné, cependant, que le processus de réflexion relatif à ces idées est très récent, a contribué à la naissance d'une nouvelle approche, c'est celle des systèmes basés sur les agents. Cette approche, a cependant émergé, elle consiste en un ensemble d'agents logiciel qui communiquent entre eux en vue d'une résolution coopérative de problèmes. C'est ce que nous appelons les systèmes multi-agents.

Les systèmes multi-agents ont été mis au point dans un but méthodologique dans le domaine de l'informatique distribuée en intelligence artificielle. Ils ont ensuite évolué pour devenir à la fois une méthode de gestion et une technique d'ingénierie logicielle. Ils visent à étudier la résolution de problèmes par une collection d'agents autonomes. Concept central de ce domaine, le terme d'agent peut être défini comme une entité physique ou virtuelle capable d'agir dans un environnement et de communiquer avec d'autres agents.

L'intérêt d'aborder le processus de data warehousing avec une approche multi-agents réside dans la compatibilité et les convergences entre ces deux domaines. En effet, si l'on considère qu'un agent représente une entité porteuse d'information ou de connaissances alors un système multi-agents couvre parfaitement les caractéristiques de l'entreposage de données en entreprise : distribution des données et des connaissances, autonomie des entités ainsi que les différentes interactions entre entités (agent), etc.

3. Objectifs

L'objectif de notre travail est de pouvoir, d'une part, impliquer les agents pour la conception et l'implémentation d'une architecture du data warehousing capable d'intégrer les informations des bases de données réparties à grande échelle et hétérogènes dans le contexte de data warehousing. D'autre part et de bénéficier au maximum des avantages offerts par la technologie agent.

Partant de là, notre étude consiste à proposer une architecture basée sur les agents et adaptée à l'entreposage de données dans une entreprise ou association répartie. Cette architecture est basée sur une architecture modulaire d'entreposage de données proposée par le projet WHIPS, et on la fusionne avec l'architecture du système multi-agents NetSA qui est destinée aux environnements riches en informations hétérogènes comme celle de notre cas.

4. Organisation du mémoire

Pour l'élaboration de ce mémoire, nous avons été conduits à le scinder en cinq chapitres :

Les deux premiers chapitres comportent une étude théorique concernant un état de l'art sur les différents concepts liés au domaine de l'informatique décisionnelle et utilisés pour le développement d'un Data Warehouse, en montrant les concepts de base d'un entrepôt de données, la modélisation dimensionnelle, le concept OLAP, le processus d'alimentation ou de chargement, et les techniques de navigation dans les données.

Dans le troisième chapitre, nous commençons par introduire l'intelligence artificielle distribuée. Ensuite, nous donnerons une vue d'ensemble sur les notions d'agents et des systèmes multi-agents en mettant l'accent sur l'évolution de l'aspect individuel (le comportement d'un agent) vers l'aspect collectif (son comportement dans une société d'agents).

Le quatrième chapitre est consacré à la conception d'une architecture à base d'agents, dédiée à l'entreposage de données. En premier lieu, nous présentons deux modèles d'architectures de référence, WHIPS et NetSa, nécessaires pour mettre en œuvre notre architecture. Dans la deuxième section nous montrons les différentes étapes parcourues pour atteindre une architecture affinée avec une description de haut niveau de l'architecture. Enfin, la dernière section est consacrée à une conception détaillée des agents de notre architecture et on termine par quelques diagrammes UML de fonctionnement.

Le dernier chapitre sera consacré à la mise en œuvre d'une implémentation avec étude de cas pour valider notre solution.

Enfin nous terminerons notre mémoire avec une conclusion et des perspectives.

CHAPITRE I 1

Etat de l'art sur le data warehousing

CHAPITRE I : ETAT DE L'ART SUR LE DATA WAREHOUSING

1. INTRODUCTION

Les projets *Data warehouse* sont pilotés par les besoins de gestion. Au sein de l'organisation, un Data warehouse met à la disposition des décideurs une information précise extraite des systèmes opérationnels.

L'idée de constituer une base de données orientée sujet, intégrée, contenant des informations datées, non volatiles et exclusivement destinées aux processus d'aide à la décision fut dans un premier temps accueillie avec une certaine perplexité.

Mais l'économie actuelle en a décidé autrement. Les entreprises sont confrontées à une concurrence de plus en plus forte, des clients de plus en plus exigeants, dans un contexte organisationnel de plus en plus complexe et mouvant. Pour faire face aux nouveaux enjeux économiques, l'entreprise doit anticiper. L'anticipation ne peut être efficace qu'en s'appuyant sur de l'information pertinente. Cette information est à la portée de toute entreprise qui dispose d'un capital de données gérées par ses systèmes opérationnels et qui peut en acquérir d'autres auprès de fournisseurs externes. [1]

Dans ce chapitre, nous commencerons par une illustration et des définitions du contexte général puis on va donner une présentation de l'évolution du concept « Data Warehouse » ensuite on donnera un aperçu sur la signification du terme Data Warehouse ainsi que sa structure de données et les différentes architectures utilisées pour son implémentation qui seront suivi par un schéma générale de développement d'un Data Warehouse. On terminera ce chapitre par un aperçu sur différents types d'architecture.

2. INTELLIGENCE ECONOMIQUE, BUSINESS INTELLIGENCE ET SYSTEME D'INFORMATION DECISIONNELLE

2.1. Intelligence économique

L'intelligence économique est un outil de compétitivité qui permet de fournir à l'entreprise une compréhension fine de son environnement, en utilisant tous les moyens d'information disponibles, en traitant l'information pour agir et appréhender les stratégies des concurrents. C'est anticiper sur les marchés à venir et prendre les meilleures décisions dans un contexte économique fluctuant.[2]

Le groupe de travail présidé par Henri Martre [4], en 1994, retient de l'intelligence économique la définition suivante : «L'intelligence économique l'IE est un *ensemble des actions coordonnées de recherche, de traitement et de* diffusion *de l'information utile aux acteurs économiques en* vue *de son exploitation à des fin stratégiques et opérationnelles. Ces diverses actions sont menées légalement avec toutes les garanties de protection nécessaires à la préservation du patrimoine de l'entreprise, dans les meilleures conditions de qualité, de délais et de coût…*» Carlo Revelli [5] propose une définition qui tient compte de ces concepts : «Processus de collecte, traitement et diffusion de l'information qui a pour objet la réduction de la part d'incertitude dans la prise de toute décision stratégique. Si à cette finalité on ajoute la volonté de mener des actions d'influence, il convient de parler alors d'intelligence économique».

Celle-ci ne se résume évidemment pas à la surveillance des activités des concurrents. C'est l'ensemble de l'environnement de l'entreprise qui est concerné. Ces diverses actions sont menées en toute légalité et en préservant les informations stratégiques de l'entreprise.[6]

L'intelligence économique a pour objectif de permettre aux décideurs et managers de l'entreprise de disposer d'une information de valeur, à laquelle ils puissent se fier dans le cadre de leurs prises de décision. Pour cela, il s'agit de produire de l'information pertinente et à forte valeur ajoutée. Cette exigence doit se retrouver à travers les différentes phases du processus :

- Collecte de l'information,
- Traitement,
- Diffusion.

Nous allons utiliser le terme « *entreprise* » pour englober tous les organismes socio-économiques, ceci pour montrer que le processus d'IE ne se limite pas aux entreprises de production ou de service, mais qu'il concerne tous les organismes socio-économiques.[7]

2.2. Business intelligence

La Business Intelligence est l'équivalent anglais du terme Veille Economique. Le business intelligence est devenu un quasi-synonyme de l'information décisionnelle. Le business intelligence peut traite des flux d'information provenant essentiellement de quatre sources :[3]

- Le système d'information interne de l'organisation,
- Les partenaires : institutions privées ou publiques,
- Des fournisseurs de données institutionnels : organisations professionnelles, instituts de sondage, producteurs de bases de données,
- Les supports d'information publics : la presse.

2.3. Système D'Information Décisionnel

Dans le monde des entreprises grandes ou petites dont l'infrastructure repose en grande partie sur des moyens informatiques, la mise en œuvre de la business intelligence est explicite et passe de plus en plus par un Système d'Information Décisionnel (SID). Un Système d'information décisionnel, bien entendu, n'est pas un système qui prend les décisions. En Anglais, on dit Décision Support System (DSS), il s'agit d'un système de «soutien» à la décision. La décision elle-même est humaine ; la vocation du système d'information décisionnel n'est pas d'automatiser la décision, elle est d'automatiser le processus de recherche d'information et la mise en forme des données nécessaires à la prise de décision. La décision elle-même est un processus socio-technique, dans lequel les acteurs humains sont en inter-action de plus en plus étroite avec des systèmes automatisés.

La notion de système d'information décisionnel s'est développée dans la dernière décennie du 20ème siècle, alors que celle de Système d'Information (SI) existait déjà. Pourquoi, alors ayant déjà «inventé» le système d'information, il a fallu en inventer un autre, qualifié de décisionnel ? Que demander de plus que des «informations» pour pouvoir décider ? La réponse est à la fois simple et embarrassante : le système d'information «classique» est un système... qui n'informe pas.[2]

3. LES SYSTEMES D'INFORMATIONS ACTUEL [6]

L'informatique de gestion a gagné sa place dans l'entreprise depuis les années 60 par une succession de progrès technologiques, logiciels et méthodologiques qui ont tous contribué à une réduction des coûts d'exploitation. L'invention du compilateur et de la compatibilité des séries de machines dans les années 60 a permis aux grands comptes de s'équiper. Le microprocesseur et les bases de données dans les années 70 ont rendu l'informatisation accessible aux moyennes et grandes entreprises. Les bases de données relationnelles, les progiciels de gestion, ainsi que les premiers

micro-ordinateurs des années 80 ont largement contribué à l'équipement des petites et moyennes entreprises, commerces, administrations.

Jusque là, la plus grande partie des applications était dédiée au traitement des données directement liées à l'activité quotidienne des organisations : paie, comptabilité, commandes, facturation. On regroupe ces applications sous le terme d'Informatique de Production ou d'Informatique Opérationnelle. L'architecture générale était l'architecture maître-esclave, avec le maître, un puissant ordinateur (mini ou gros système) en site central et les esclaves, terminaux passifs en mode texte.

Avec l'apparition des ordinateurs personnels et des réseaux locaux, une autre activité a émergé, tout à fait distincte de l'informatique de production. Dans les secrétariats, les cabinets, on utilise des tableurs et des logiciels de traitements de texte, des petites bases de données sur des machines aux interfaces graphiques plus agréables. Jusqu'aux années 90, ces deux mondes bureautique/informatique se sont ignorés.

La montée en puissance des micro-ordinateurs et l'avènement de l'architecture client-serveur a permis un décloisonnement remarquable entre bureautique et informatique. Le but principal est de fournir à tout utilisateur reconnu et autorisé, les informations nécessaires à son travail. Cette nouvelle approche de l'information fait naître une nouvelle informatique, intégrante, orientée vers les utilisateurs et les centres de décision des organisations. C'est l'ère du client- serveur qui prend vraiment tout son essor à la fin des années 90 avec le développement des technologies Internet.

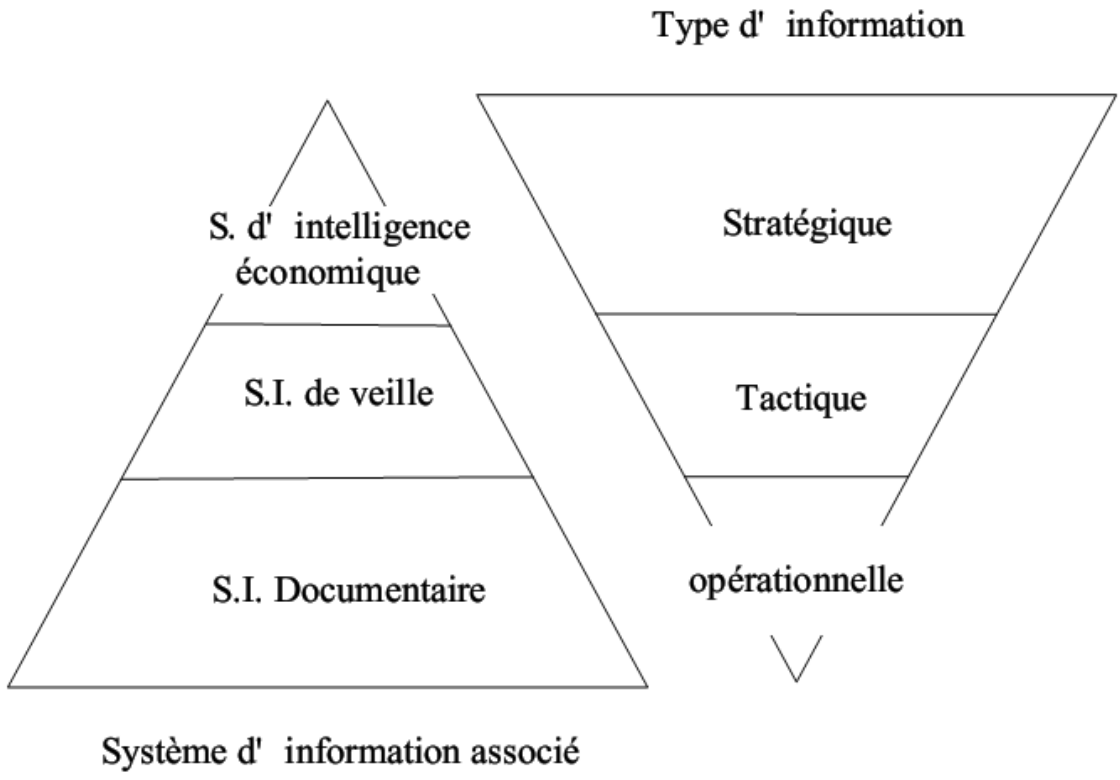


Figure 1: Type d'Information Et Système d'Information.[6]

4. POURQUOI UN DATA WAREHOUSE

4.1 La problématique des entreprises

Depuis plusieurs dizaines d'années, une importante masse d'informations est stockée sous forme informatique dans les entreprises. Les systèmes d'information sont destinés à garder la trace d'événements de manière fiable et intègre. Ils automatisent de plus en plus les processus opérationnels.

Parallèlement, les entreprises réalisent la valeur du capital d'information dont elles disposent. Au delà de ce que l'informatique leur apporte en terme fonctionnel, elles prennent conscience de ce qu'elle pourrait apporter en termes de contenu informationnel. Considérer le système d'information sous cet angle en tant que levier pour accroître leur compétitivité et leur réactivité n'est pas nouveau. Par contre, étant donné l'environnement concurrentiel actuel, cela devient une question de survie.

L'informatique a un rôle à jouer, en permettant à l'entreprise de devenir plus entreprenante et d'avoir une meilleure connaissance de ses clients, de sa compétitivité ou de son environnement. [9]

4.2 La réalité des systèmes d'informations

A première vue, les systèmes opérationnels seraient des mines d'or informationnelles. En fait, il n'en est rien.

Les données contenues dans ces systèmes sont :

- **Eparpillées** : il existe souvent de multiples systèmes, conçus pour être efficace pour les fonctions sur lesquelles ils sont spécialisés.
- **Peu structurées pour l'analyse** : la plupart des systèmes informatiques actuels ont pour objet de conserver en mémoire l'information, et sont structurés dans ce but.
- **Focalisées pour améliorer le quotidien** : toutes les améliorations technologiques se sont focalisées pour améliorer cette capacité en terme de volume, qualité, rapidité d'accès. Il manque très souvent la capacité à nous donner les moyens de tirer parti de cette mémoire pour prendre des décisions.
- **Utilisées pour des fonctions critiques** : la majorité des systèmes existants est conçue dans le but unique de nous servir avec des temps de réponse corrects.

Get the Data In

- Prendre une commande
- Traiter une réclamation
- Faire un envoi
- Générer une facture
- Réserve dans une compagnie aérienne

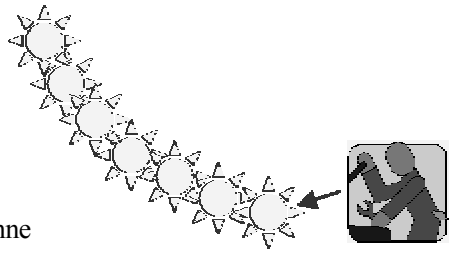


Figure 2 : les systèmes opérationnels. [9]

Donc on a besoin de concevoir et établir des systèmes d'information :[9]

- qui servent a des objectifs différents;
- dont les emplois sont différents;
- dont le contenu des données est différent;
- lorsque les modèles des données sont différents;
- lorsque les types d'accès aux données sont différents.

A ce titre, l'entreprise construit un système décisionnel pour améliorer sa performance. Elle doit décider et anticiper en fonction de l'information disponible et capitaliser sur ses expériences.

4.3 L'informatique décisionnelle [10]

L'informatique décisionnelle désigne les moyens, les outils et les méthodes qui permettent de collecter, consolider, modéliser et restituer les données d'une entreprise en vue d'offrir une aide à la décision et de permettre aux responsables de la stratégie d'une entreprise d'avoir une vue d'ensemble de l'activité traitée.

Ce qui caractérise d'abord les besoins, c'est la possibilité de poser une grande variété de questions au système, certaines prévisibles et planifiées comme des tableaux de bord et d'autres imprévisibles. Si des outils d'édition automatiques préprogrammés peuvent être envisagés, il est nécessaire de permettre à l'utilisateur d'effectuer les requêtes qu'il souhaite, par lui-même, sans l'intervention de programmeurs Il sera souvent nécessaire de filtrer, d'agréger, de compter, sommer et de réaliser quelques statistiques élémentaires (moyenne, écart-type,...). La structure logique doit être prévue pour rendre aussi efficace que possible toutes ces requêtes. Pour y parvenir, il est nécessaire d'introduire de la redondance dans les informations stockées en mémorisant des calculs intermédiaires. On rompt donc avec le principe de non redondance des bases de production.

La cohérence assurée par les systèmes de production est toute relative. Elle se contrôle au niveau de la transaction élémentaire mais pas au niveau global et des activités de l'organisation. A l'inverse des systèmes d'informatique décisionnelle, la cohérence requise doit être interprétable par l'utilisateur. Par exemple, si les livraisons n'ont pas été toutes saisies dans le système, comment garantir la cohérence de l'état du stock ?

Les systèmes d'informatique décisionnelle doivent donc assurer plutôt une cohérence globale des données. Pour ce faire, leur alimentation doit être une opération réfléchie et planifiée dans le temps. Les transferts de données du système opérationnel vers le système décisionnel seront réguliers avec une périodicité bien choisie dépendante de l'activité de l'entreprise. Chaque transfert sera contrôlé avant d'être diffusé.

Une dernière caractéristique importante de l'informatique décisionnelle, qui est aussi une différence fondamentale avec les bases de production, est qu'aucune information n'y est jamais modifiée. En effet, on mémorise toutes les données sur une période déterminée, les données ne seront jamais remises à jour car toutes les vérifications utiles à la cohérence globale sont procédées lors de l'alimentation.

L'utilisation se résume donc à un chargement périodique, puis à des interrogations non régulières, non prévisibles, parfois longues à exécuter. Et le Data Warehouse doit être rapproché de tous les concepts visant à établir une synergie entre le système d'information et sa stratégie.

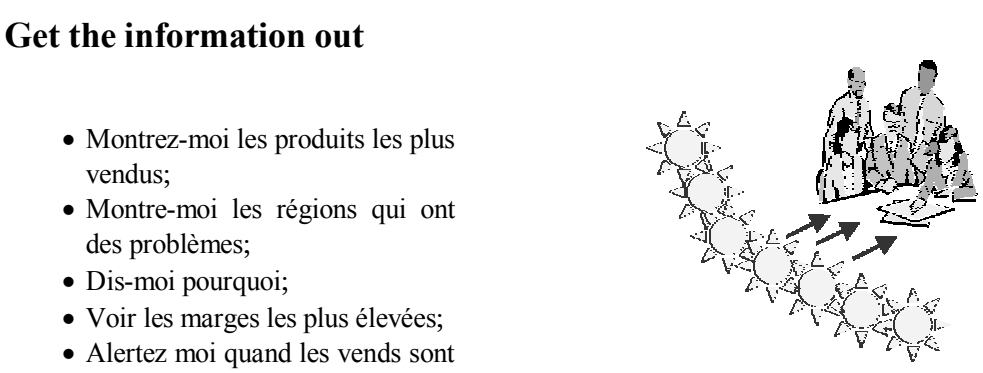


Figure 3 : L'informatique décisionnelle [9]

4.4. Système Décisionnel Versus Système Opérationnel

Afin de mieux saisir les différences qui existent entrent un système opérationnel (base de production) et un système décisionnel (Data Warehouse), nous donnons le tableau suivant qui les résume.

	Base de production	**Data Warehouse**
Données	Atomique	Résumé
	Oriente application	Orienté sujet
	A jour	Historiques
	Dynamique	Statique
Utilisateurs	Employés de bureau	Analystes
	Nombreux	Peu
	Mise à jour	Interrogations
	Accès à peu de données	Accès à beaucoup d'informations
	Réponses immédiates	Réponses moins rapides
Mode de fonctionnement	On line Transaction Processing (OLTP)	On Line Analytic Processing (OLAP)

Tableau 1 : Comparaison entre base de production et Data Warehouse.

5. PRESENTATION DU DATA WAREHOUSE

5.1 Définitions du Data Warehouse

D'une manière générale, on peut définir le Data Warehouse ou en français « Entrepôt de données » comme une base de données spécialisée dans lequel est centralisé l'ensemble des données que possède une entreprise et qui ont été jugées pertinentes à la prise de décision.

Plusieurs auteurs ont défini le concept de « Data Warehouse », nous donnerons les définitions les plus importantes et les plus citées dans la littérature :

Peut-être défini selon le grand dictionnaire terminologique de l'office québécois de la langue française, comme " Structure informatique dans laquelle est centralisé un volume important de données consolidées à partir des diverses bases de données internes et externes d'une entreprise, et qui est conçue pour offrir un accès rapide à l'information stratégique nécessaire à la prise de décision.

Dans cette pièce maîtresse de l'informatique décisionnelle, les données sont sélectionnées et préparées (pour répondre aux questions vitales de l'entreprise), intégrées (à partir des différentes sources de renseignements) et datées (elles gardent la trace de leur origine).". [11]

Alors que, Chaudhuri définit « Le Data Warehouse collecte les données en provenance de différentes applications de production, les intègre, les stocke de façon à ce qu'elles soient accessibles par un utilisateur final et délivre des analyses. ». [12]

Et d'après Georges Gardarin « un ensemble de données historisées variant dans le temps, constitué par extraction à partir de bases applicatives ou de fichiers, organisé par sujets spécifiques, consolidé dans une base de données unique, géré dans un environnement de stockage particulier, et aidant à la prise de décision dans l'entreprise » [13].

Barry Devlin décrier le data warehouse comme " un unique, complet et consistant stock de données obtenues à partir des sources divers et les faire disponible à l'utilisateur finale dans le contexte du Bisness"[14]

Pour Bill Inmon qui est le fondateur de ce concept on 1990 avec Hackarton « Le Data Warehouse est une collection de données ***orientées sujet***, ***intégrées***, ***non volatiles*** **et** ***historisées***, organisées pour le support d'un processus d'aide à la décision. ». [1]

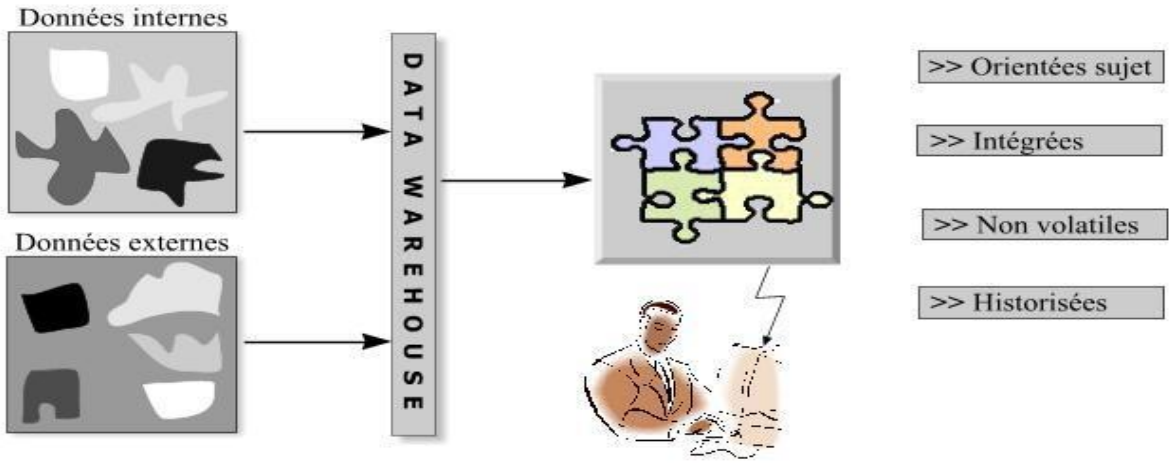


Figure 4 : Définition de l'entrepôt de données.

5.2 CARACTERISTIQUES DU DATA WAREHOUSE

D'après la définition de Inmon [1], qui englobe différents termes nous explicitons ici les caractéristiques de DW.

5.2.1 Les données doivent être orientées sujet

L'orientation « SUJET » signifie que le Data Warehouse (DW) est organisé autour des sujets majeurs de l'entreprise. Ainsi les données sont structurées par thèmes contrairement aux données des

organisations traditionnelles généralement organisées par processus fonctionnel comme indiqué dans la figure 5.

L'intérêt de cette organisation est de disposer de l'ensemble des informations utiles sur un sujet, le plus souvent transversal aux structures fonctionnelles et organisationnelles de l'entreprise.

Cette orientation sujet va également permettre de développer le système décisionnel via une approche par itérations successives, sujet après sujet. L'intégration dans une structure unique est indispensable car les informations communes à plusieurs sujets ne doivent pas être dupliquées. Dans la pratique, une structure supplémentaire appelée Data Mart (magasin de données) peut être créée pour supporter l'orientation sujet. [8]

Par exemple, une compagnie d'assurance peut avoir les processus fonctionnels suivants : véhicules à moteur, responsabilité civile, vie individuelle ou vie collective. Alors que ses principaux sujets sont le produit, la police d'assurance, la prime et le sinistre. [15]

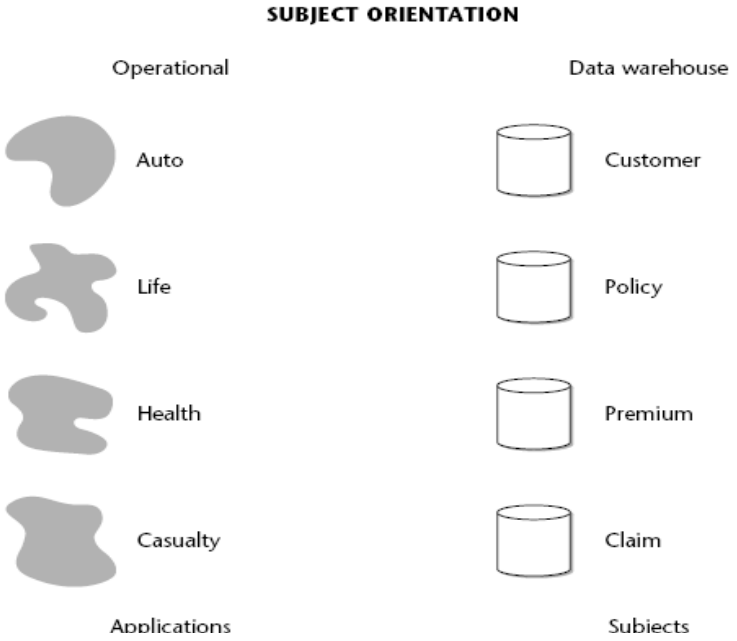


Figure 5 : Données orientées sujets [1]

5.2.2 Les données doivent être intégrées

Les données alimentant le DW proviennent de multiples applications ayant des sources hétérogènes. Il n'existe pas d'homogénéité entre les différents systèmes opérationnels de l'organisation, ils ont :

- des conventions différentes,
- des valorisations différentes,
- des structures de données différentes,

Ces données sont traitées (converties, reformatées et nettoyées) avant de remplir la base du Data Warehouse de manière à obtenir un ensemble de données homogène, qui deviennent ainsi comparables, additionnables,… (**Figure 6**). **[15]**

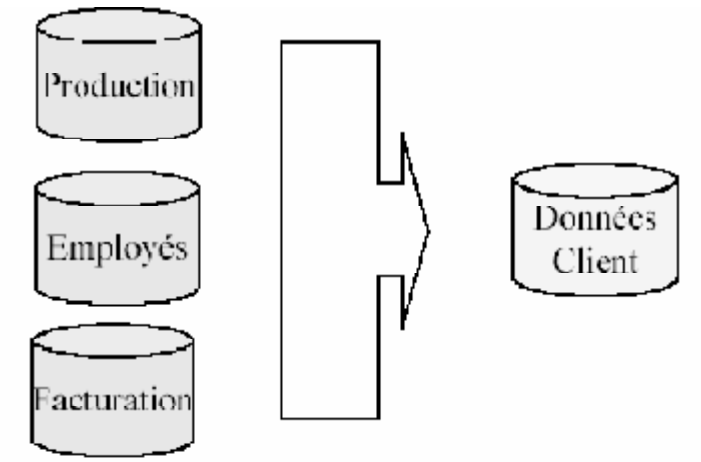


Figure 6 : Données orientées sujets. [1]

Donc l'intégration consiste à résoudre les problèmes d'hétérogénéité des systèmes de stockage, des modèles de données, de sémantique de données [16]. C'est pour cette raison que le Data Warehouse est un système intégré et non pas une simple juxtaposition de données d'origines diverses.

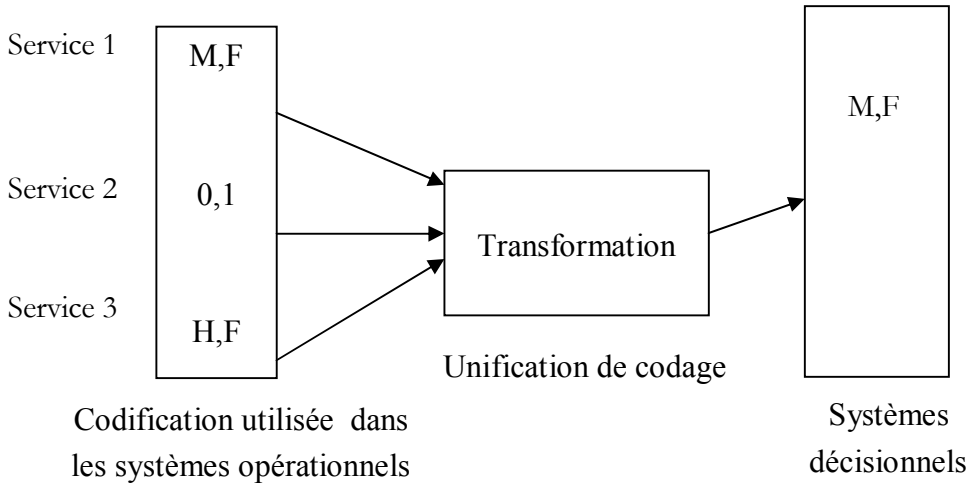


Figure 7 : Intégrité des données.

L'exemple illustre que, si on a une codification différente de la personne dans chaque système opérationnel au niveau de l'organisation alors la codification doit être unifiée avant de remplir le Data Warehouse c-à-d la personne a une seule codification dans le Data Warehouse au niveau de cette organisation.

Dans la réalité des projets cette phase d'intégration est très complexe, longue, fastidieuse et pose souvent des problèmes de qualification sémantique des données à intégrer. Elles représenteraient 60 à 90% de la charge totale d'un projet. [8]

5.2.3 Les données doivent être historisées

Dans un système de production, la donnée est mise à jour à chaque nouvelle transaction. L'ancienne valeur est perdue et la donnée reste constamment à jour. Les systèmes de production conservent assez rarement l'historique des valeurs de cette donnée.

Dans un Data Warehouse, la donnée ne doit jamais être mise à jour. Elle représente une valeur insérée dans le système décisionnel à un certain moment.

Il est évident qu'un référentiel de temps doit être associé à la donnée afin d'être capable d'identifier une valeur particulière dans le temps. Chaque nouvelle insertion de données provenant du système de production ne détruit pas les anciennes valeurs, il crée une nouvelle occurrence de la donnée (**Figure**8).

La prise en compte de l'évolution des données est essentielle pour la prise de décision qui par exemple utilise des techniques de prédiction en s'appuyant sur les évolutions passées pour prévoir les évolutions futures.[16]

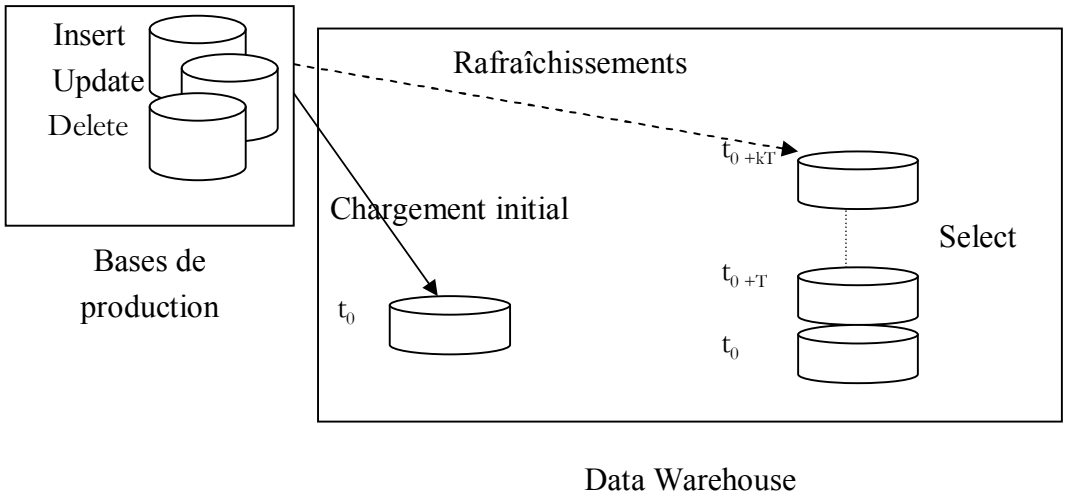


Figure 8 : Données historisées.

5.2.4 Les données doivent être non volatiles

La non-volatilité des données est en quelque sorte une conséquence de l'historisation décrite précédemment. Les données de l'entrepôt sont essentiellement utilisées en mode de consultation. Elles ne sont pas modifiées par les utilisateurs.[16] Ainsi, une même requête effectuée à trois mois

d'intervalle en précisant naturellement la date de référence de l'information recherchée donnera le même résultat (cohérence du système d'aide à la décision). [8]

Dans un système de production, l'information est volatile, la donnée est régulièrement mise à jour. Les requêtes portent sur les données actuelles et il est impossible de retrouver un ancien résultat. Au contraire des bases de données opérationnelles dont les données sont constamment modifiées, cela permet d'avoir à un moment donné (durant la période de rafraîchissement) un ensemble de valeurs stables non volatiles sur lesquelles il est possible d'effectuer des calculs.

5.2.5 Les données sont résumés

Les informations issues des sources de données doivent être agrégées et réorganisées afin de faciliter le processus de prise de décision. [17]

5.2.6 Disponible pour l'interrogation et l'analyse

Les utilisateurs doivent pouvoir consulter les données réorganisées de l'entrepôt en fonction de leur droit d'accès au travers d'outils interactifs d'aide à la manipulation et l'analyse. **[17]**

5.2.7 Une solution, n'est pas un produit

Souvent nous pensons qu'un entrepôt de données est un produit, ou groupe de produits, que nous pouvons acheter pour nous aider à obtenir des réponses à nos questions et à améliorer notre capacité de prise de décision. Mais, elle n'est pas aussi simple. Un entrepôt de données peut nous aider à obtenir des réponses pour une meilleure prise de décision, mais c'est seulement une part d'un processus global. Comme exemples, d'où les données dans l'entrepôt de données sont-elles venues?

Comment est-il entré dans l'entrepôt de données? Comment est met à jour? Comment les données sont structurées dans l'entrepôt de données? Qu'est ce qu'il y'a actuellement dans l'entrepôt de données? Ce sont toutes les questions qui doivent être répondues avant qu'un entrepôt de données puisse être construit. Nous préférons discuter l'environnement plus global, et nous nous référons à lui comme "Data warehousing" ou encore en français "entreposage de données".[**14**]

5.2.8 Data warehousing

L'entreposage de données ou le Data warehousing [18] [12] entoure et englobe des frameworks, architectures, algorithmes, outils, et des techniques pour rassembler des données nécessaires à la création d'un data warehouse ainsi qu'à son exploitation.

C'est la conception et l'implémentation des processus, des outils, et des équipements pour manager et fournir l'information complète, opportune, précise, et compréhensible pour la prise de décision. Il comprend toutes les activités qui permettent à une organisation de créer, manager, d'alimentation un data warehouse ainsi que la possibilité de faire des requêtes.[14].[15]

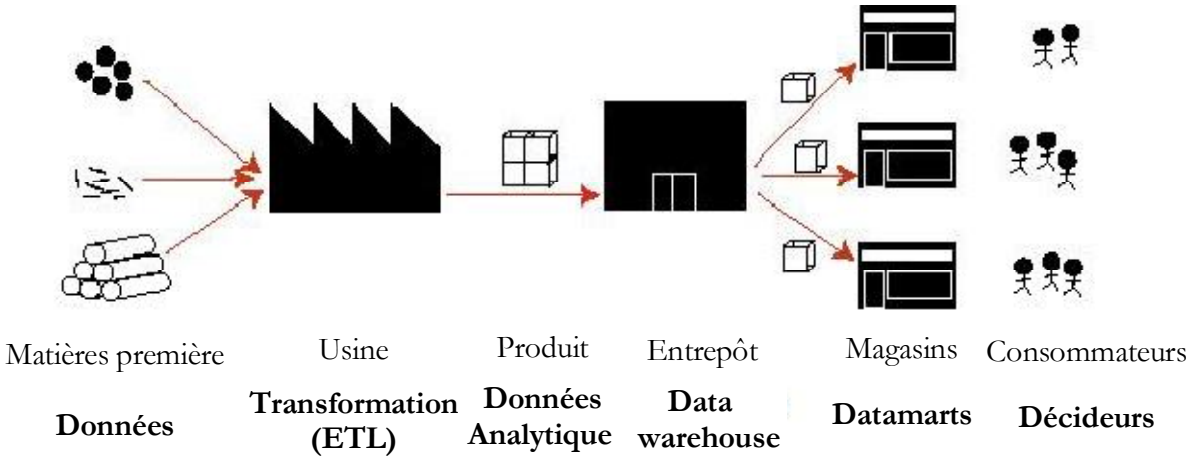


Figure 9 : La supply-chain de l'information [19]

5.3 Les objectifs du data warehouse[15]

Voici les objectifs du data warehouse tels que définis par Ralph Kimball, dans son livre « Entrepôts de données, Guide pratique du concepteur de data warehouse », [20].

5.3.1 Accès aux informations de l'entreprise

L'entrepôt de données assure l'accès aux informations de l'entreprise et de l'organisation.

Le contenu de l'entrepôt doit être compréhensible et l'utilisateur final doit pouvoir y naviguer facilement et avec rapidité.

5.3.2 Les informations d'un entrepôt de données sont cohérentes.

Cela signifie que les requêtes faites à des moments différents doivent fournir les mêmes résultats. La cohérence veut aussi dire que lorsque les personnes demandent la définition de l'élément « contrat », elles obtiennent une réponse leur permettant de savoir ce qu'elles obtiendront de la base de données. La cohérence implique que les données sont chargées dans leur totalité. Les données d'un entrepôt doivent pouvoir être séparées et combinées au moyen de toutes les mesures possibles de l'activité.

5.3.3 Les outils de présentation d'informations font partie du data warehouse

L'entrepôt de données ne comporte pas seulement des données, mais aussi un ensemble de requêtes, d'analyses et de présentations des informations.

5.3.4 Les données publiées sont stockées dans le data warehouse

L'entrepôt de données est le lieu où sont publiées des données qui ont déjà servi. Les données sont soigneusement rassemblées à partir de sources d'informations situées à différents endroits de l'organisation. Elles sont nettoyées, leur qualité est vérifiée et elles ne

sont diffusées que si elles sont prêtes à être utilisées. Si l'information est peu fiable ou incomplète, les données ne peuvent être publiées à destination de la communauté des utilisateurs.

5.3.5 Qualité de l'information d'un data warehouse

La qualité de l'information d'un entrepôt de données est très importante. En effet, comment obtenir des analyses fiables si les données brutes sont de mauvaise qualité ? L'entrepôt de données ne peut remédier à la mauvaise qualité des données ou à l'absence d'une donnée. La seule façon de remédier à la médiocre qualité des données consiste, pour les personnes concernées par la saisie des données et pour le management, à retrouver la source des informations et à mettre en place de meilleurs systèmes ou à mieux faire comprendre l'importance de la qualité des données.

6. ARCHITECTURE FONCTIONNELLE D'UN ENTREPOT DE DONNEES

L'architecture de l'entrepôt de données comporte trois niveaux fonctionnels essentiels : le niveau acquisition des données, le niveau stockage des données et le niveau analyse de données.[21] La figure représente l'architecture 3 tiers du Data warehouse et s'est majeurs composants:

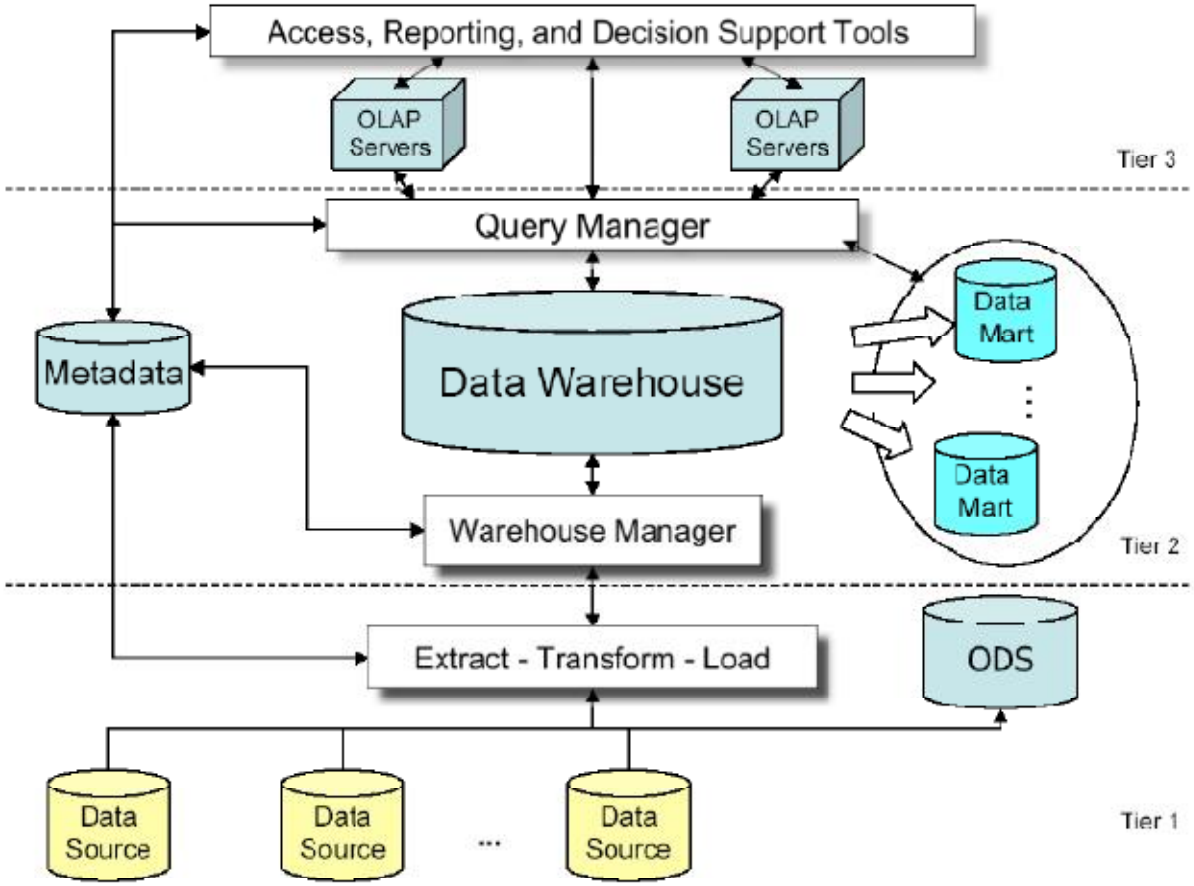


Figure 10 : L'architecture du Data Warehouse[22]

6.1. Le niveau acquisition des données

Est effectuée à l'aide d'un outil ETL (Extract, Transform, Load), consiste à aller chercher les données là où elles se situent, à les trier, et à les transformer éventuellement afin d'effectuer un prétraitement pour faciliter l'analyse. Dans cette phase se fait également le nettoyage des données : l'homogénéisation, la suppression des doublons, la détection de données non conformes. Ensuite, les données sont centralisées dans le DatawareHouse. [19]

6.2. Le niveau stockage des données

Le stockage sert à la fusion et chargement des données extraites et les stocke dans la base de données du Data Warehouse ; il procède au chargement initial du Data Warehouse, puis à son rafraîchissement au fur et à mesure des mises à jour.

Ici également intervient de manière très forte la structuration physique de l'entrepôt de données, et ceci pour garantir des performances stables dans le temps, des indexations, des restructurations plus faciles.

6.3. Le niveau analyse de données

C'est le niveau de la **restitution** ou **reporting**, qui consiste en l'analyse des données et à la diffusion des informations. C'est la face visible du décisionnel, la partie que voient les utilisateurs. Nous distinguerons 3 sortes d'analyse de données : la simple requête vers une base (multidimensionnelle ou non), l'analyse multidimensionnelle et la constitution de tableaux de bord, et le datamining (fouille de données), consistant à mettre en relation des corrélations éventuelles entre les données afin de dégager une tendance. [19]

Les métadonnées

Toutes les informations de l'environnement du data warehouse qui ne constituent pas les données proprement dites [21]. Ce sont les « données sur les données ».

Les méta-données, stockées dans un répertoire différent de celui des données de l'entrepôt, sont une sorte de médiateur entre l'utilisateur et l'entrepôt. Elles renferment des informations sur la création et la gestion du Data Warehouse. [23]

7. LA STRUCTURE DE DONNEES DU DW

Un DW se structure en quatre classes de données, organisées selon un axe historique et un axe synthétique.

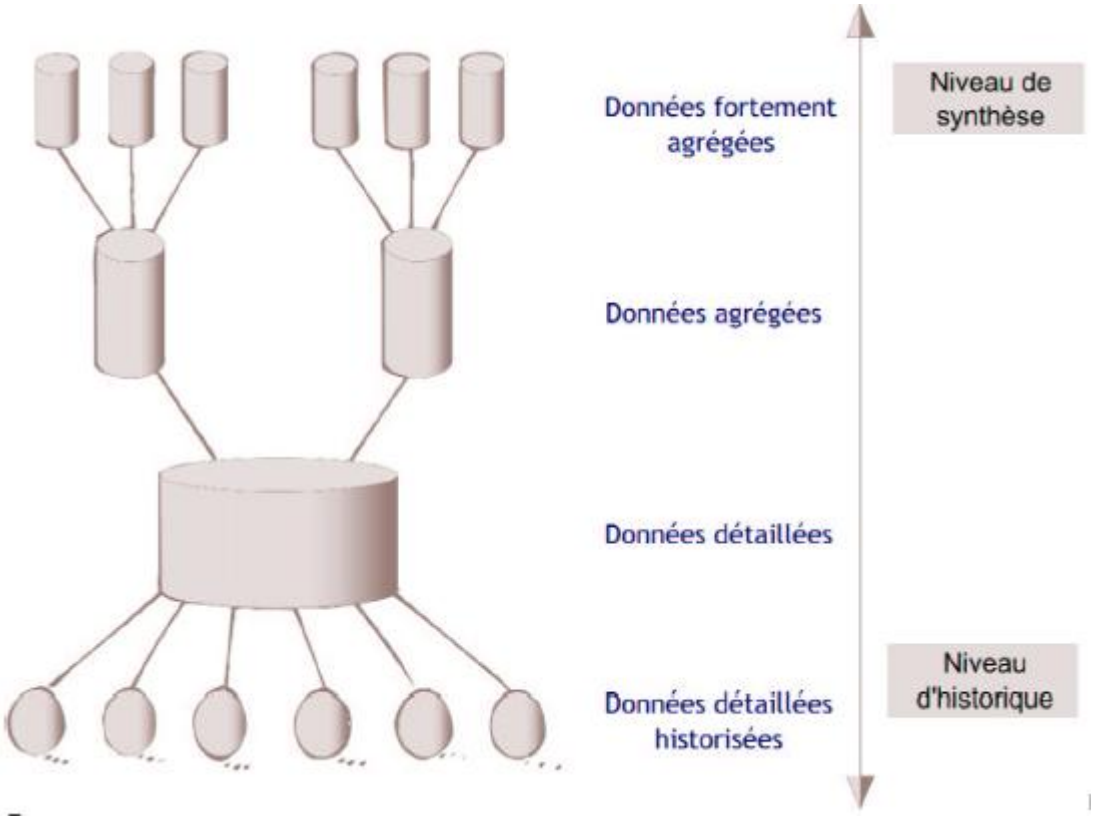


Figure 11 : La structure des données d'un Data Warehouse. [1]

7.1. Axe historique

Les volumes à traiter et à manipuler dans cet axe sont plus importants que ceux gérés en transactionnel. Le niveau de détail géré dans l'entrepôt de données n'est pas forcément identique au niveau de détail géré dans les systèmes opérationnels. On trouve ici deux types de données : les données détaillées et les données détaillées historisées.

- **Les données détaillées**

  Elles reflètent les événements les plus récents. Les intégrations régulières des données issues des systèmes de productions vont habituellement être réalisées à ce niveau.

- **Les données détaillées historisées**

Elles reflètent les transactions historiques et détaillées de l'entreprise. L'entrepôt de données ou le Date Warehouse doit conserver l'historique des données et l'insertion de nouvelles valeurs ne doit jamais annuler l'anciennes existant.

7.2. Axe synthétique

Elles correspondent à des éléments d'analyse représentant les besoins des utilisateurs. Elles constituent déjà un résultat d'analyse et une synthèse de l'information contenue dans le système décisionnel. Elles doivent être accessibles et compréhensibles. On trouve ici deux types de données : les données agrégées et les données fortement agrégées.

- **Les données agrégées**

  Dans le cas des données agrégées, l'information est composée par exemple de la somme des ventes et de l'unité (par mois, par produit, ...).

- **Les données fortement agrégées**

  Les données fortement agrégées sont presque comme les données agrégées, elles ont une incidence énorme sur les performances ; dans certains cas elles permettent d'exécuter les requêtes 100 fois, voire 1000 fois plus rapidement.

La logique d'accès aux données la plus utilisée est la suivante : les utilisateurs commencent à rechercher les données par le niveau le plus agrégé, puis approfondissent leur recherche vers les données les plus détaillées.

7.3. Les métadonnées

Les métadonnées contiennent des informations sur la création, la gestion et l'utilisation de l'entrepôt. Celles-ci peuvent être classées en trois catégories : spécifiques à l'application, d'audit et d'administration.

- La première catégorie concerne les ontologies et les connaissances particulières au domaine d'application.
- Les métadonnées d'audit comprennent des statistiques sur l'entrepôt.
- Les métadonnées d'administration peuvent être encore classifiées en données schématiques, sémantiques, conceptuelles et relatives aux utilisateurs.
- Les métadonnées schématiques comprennent des informations sur les éléments du schéma de l'entrepôt, les schémas des sources et les règles de correspondance entre ceux-ci et le schéma de l'entrepôt. On trouve également l'information concernant l'extraction de données sources, les règles de transformation et les politiques de rafraîchissement. Les métadonnées sémantiques comprennent les règles de correspondance entre entités similaires en provenance

de sources différentes. Les métadonnées conceptuelles contiennent les informations concernant les dimensions de l'entrepôt. Finalement, les métadonnées sur les utilisateurs de l'entrepôt contiennent des informations sur les groupes et les droits d'accès associés.

On remarque que les métadonnées associées à un entrepôt sont plus nombreuses que celles associées à une base de données classique. Pour cette raison, elles sont souvent gérées par un système (nommé référentiel) qui autorise le partage des métadonnées entre les composants des systèmes pour la mise en place et l'utilisation des entrepôts.

8. LES ARCHITECTURES D'IMPLEMENTATION

Pour implémenter un DW, trois types d'architectures sont possibles : [8]

8.1 L'architecture réelle

C'est l'architecture qui est généralement retenue pour les systèmes décisionnels. Le stockage des données de l'entrepôt de données est réalisé dans une base de données séparée du système de production. Cette base de données est alimentée par des extractions périodiques. Avant le chargement, les données subissent d'abord :

- un processus d'intégration,
- un processus de nettoyage,
- un processus de transformation.

L'avantage de cette solution est de disposer de données préparées pour les besoins de la décision et répondant bien aux objectifs de l'entrepôt de données. La principale raison justifiant l'architecture réelle est l'inadaptation des données de production aux besoins des systèmes décisionnels. Les structures de données dans un système de production sont en effet complexes au niveau stockage et nécessitent une phase de programmation pour y accéder. Dans un contexte d'utilisation décisionnelle, les données doivent être compréhensibles par l'utilisateur. [6]

8.2 L'architecture virtuelle

Dans une architecture virtuelle, les données de l'entrepôt résident dans le système de production. Elles sont rendues lisibles par des produits middleware ou par des passerelles. Il n'y a pas, dans cette architecture, de coût de stockage supplémentaire et l'accès se fait en temps réel. Cependant, les nombreux désavantages de ce type d'architecture en empêchent fréquemment le choix. Les données ne sont pas préparées. Les accès décisionnels risquent de perturber les performances du système de production d'autant plus que les processus de transformation et d'intégration sont ici forcément liés au processus d'accès. Enfin, pour le cas où la gestion d'historique n'est pas prévue dans le système de production, il est impensable de l'y intégrer.

8.3 L'Architecture remote data access

L'architecture remote est une combinaison des deux architectures décrites précédemment. L'objectif est d'implémenter physiquement les niveaux agrégés, niveaux de données les plus souvent utilisées, afin d'en faciliter l'accès, de garder le niveau de détail dans les systèmes de production en y donnant accès par le biais des middlewares ou des passerelles. Cette architecture est également très rarement utilisée.

9. LE DATAMART (MAGASIN DE DONNEES)

Les Data Warehouses nécessitent de puissantes machines afin de gérer de grandes bases de données contenant les données historisées. A coté et souvent en complément se développent des bases de données ciblées sur quelques sujets limitées, appelées Datamarts (magasins de données). Ces petits Data Warehouses offrent des données aux décideurs de l'organisation pour l'analyse.

	Data Warehouse	**Data Mart**
Cible utilisateur	Toute l'entreprise	Département
Implication du service informatique	Elevée	Faible ou moyen
Base de données d'entreprise	SQL type serveur	SQL milieu de gamme, bases multidimensionnelles
Modèles de données	A l'échelle de l'entreprise	Département
Champ applicatif	Multi sujets, neutre	Quelques sujets, spécifique
Sources de données	Multiples	Quelques unes
Stockage	Base de données	Plusieurs bases distribuées
Taille	Centaine de GO et plus	Une à 2 dizaines de GO
Temps de mise en place	9 à 18 mois pour les 3 étapes	6 à 12 mois (installation en plusieurs étapes)
Coût	> 6 millions de francs	500.000 à 3 millions de francs
Matériel	Unix	NT, petit serveur Unix

Tableau 2 : Finalités des data marts et data warehouse[8]

On distingue deux types du Datamarts :

9.1 Datamarts dépendants

Comprend des données provenant du Data Warehouse ciblées sur un sujet particulier , ce type est utilisé lors de l'exploitation du Data Warehouse pour limiter l'espace d'analyse.(**figure 12**)

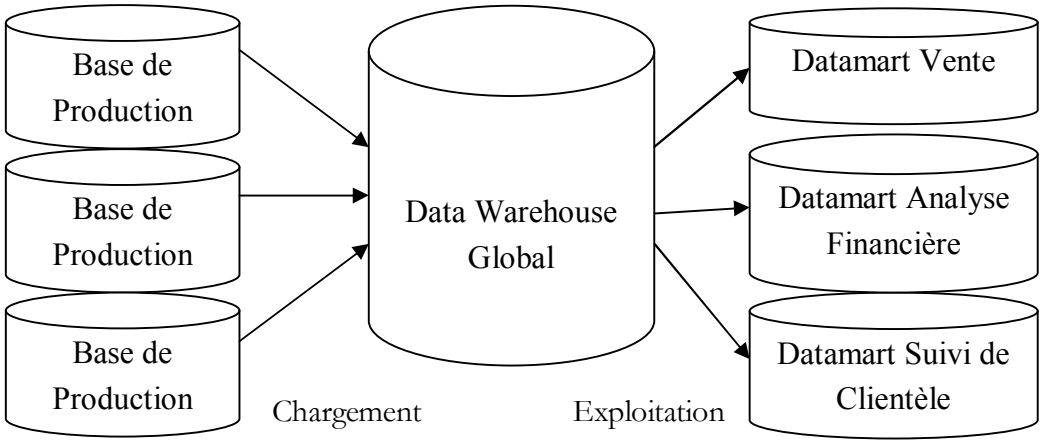


Figure 12 : Datamarts dépendants.

9.2 Datamarts indépendants

Comprend des données provenant directement de la base de production, ce type est utilisé si l'organisation s'intéresse par un sujet particulier, donc il est conçu de la même manière qu'un Data Warehouse.

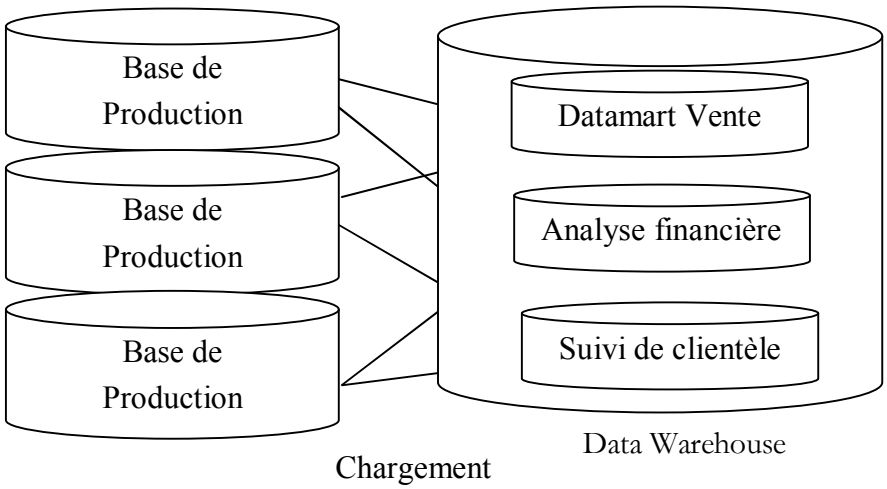


Figure 13 : Datamarts indépendants.

Ce type est utilisé lors du chargement du Data Warehouse, donc le Data Warehouse est considéré comme un ensemble des Datamarts indépendants (**Figure 13**).

10. ODS [30]

10.1. La Définition

L'ODS est l'acronym de "Operational Data Store", ce qui peut se traduire par "Dépot de données opérationnelles".

L'ODS est "Un environnement qui sert à la prise de décisions tactiques, qui contient des données détaillées, extraites en presque temps-réels des systèmes opérationnels pour répondre à des besoins de reporting immédiat et qui peut être mis à jour par les utilisateurs".

10.2. L'utilité de l'ODS

Il est souvent mis en place pour répondre à au moins un des besoins suivants :

-Intégrer les données provenant de plusieurs sources. Normalement ce genre d'intégration devrait être réalisé dans les systèmes sources, mais parce que cela peut coûter chère (temps, dispo et rentabilité) on mets en place un ODS.

-Fournir les données pour prendre des décisions tactiques (reporting)

-Permettre de consolider les mises à jour communes aux systèmes sources.

Un ODS peut servir de "staging area" pour alimenter un DW, cependant cela ne doit pas être sa raison d'être.

11. CONCLUSION

Dans ce chapitre, nous nous sommes intéressés à expliciter la notion de Data Warehouse, et ces composants qui permet au décideur de travailler dans un environnement informationnel, référencé, homogène et historisé.

A travers ce chapitre, nous avons montré que les entrepôts de données et leurs outils sont indispensables dans le monde des entreprises. En effet, pour rendre la prise de décision facile et meilleure au sein de l'entreprise, un entrepôt de données extrait, intègre, classifie par thème et analyse les données des systèmes sources.

C'est ainsi que nous avons commencé par définir le DW, ensuite nous avons donné la structure de ce que devrait avoir un DW, ainsi que ses différents types d'architecture.

Dans le chapitre suivant nous allons illustrer les différentes phases de construction et exploitation de données de data warehouse.

CHAPITRE II 2

Phases de construction et exploitation de données de « DW »

CHAPITRE II : PHASES DE CONSTRUCTION ET EXPLOITATION DE DONNEES DE « DW »

1. INTRODUCTION

L'entrepôt de données est donc bien différent des bases de données de production car les besoins pour lesquels on veut le construire sont différents. Il contient des informations historisées, globalement cohérentes, organisées selon les métiers de l'entreprise pour le processus de décision. Les données sont puisées dans les bases de production, nettoyées, normalisées, puis intégrées. Des méta-données décrivent les informations dans cette nouvelle base pour lever toute ambiguïté quant à leur origine et leur signification.

Nous décrivons dans ce chapitre, les phases à suivre pour construire un Data Warehouse. [20]

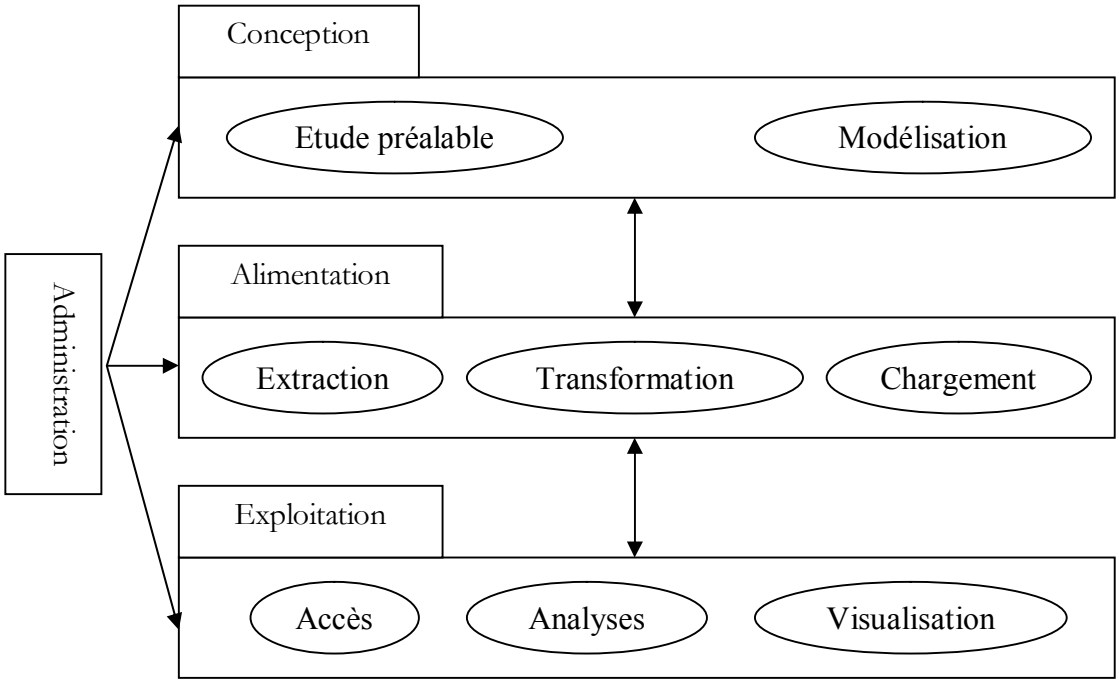


Figure 14 : Phases de construction d'un Data Warehouse.

On a relevé quatre phases interdépendantes permettant la construction d'un Data Warehouse qui sont :

- la phase de conception,
- la phase d'alimentation,
- l'administration,
- la restitution et exploitation

2. CONCEPTION

Cette phase est aussi divisée en deux étapes :

- L'étude préalable qui va spécifier les besoins de l'utilisateur, définir les objectifs, préciser la démarche.
- L'étude du modèle des données qui représente l'entrepôt conceptuellement et logiquement.

2.1 Etude préalable

Cette partie de l'étude ressemble à toutes les étapes préliminaires de l'implantation d'un nouveau système d'information automatisé. Les principes des méthodes connues pour le système de production restent valables ici (Merise,…).

2.1.1 Etude des besoins

L'étude des besoins doit déterminer le contenu de l'entrepôt et son organisation, d'après les résultats attendus par les utilisateurs des requêtes qu'ils formuleront et des projets qui ont été définis. Le besoin d'informatisation peut provenir du système de pilotage ou d'un service particulier de l'entreprise. C'est souvent un projet au sein d'un schéma directeur qui va déclencher l'étude et la réalisation d'un cahier de charges. L'existant dans ce domaine est généralement un ensemble de petits produits ou développements sans grande intégration ni relation, dissémines dans divers services. L'information est dupliquée, les traitements sont répétés et aucune stratégie d'ensemble n'est définie. Le projet va donc s'orienter vers la recherche d'une solution intégrante résolument tournées vers l'utilisateur.

L'expression des besoins par les utilisateurs met souvent en évidence la volonté d'obtenir : des analyses sur ce qui s'est passé (par exemple comparer les performances actuelles d'un magasin avec celles de l'année dernière) ou des analyses prédictives (par exemple déterminer les achats potentiels pour un type de client, déterminer les clients qui risquent d'abandonner l'organisation,…).

Les interviews doivent permettre de préciser les faits à suivre et dans quelles dimension. Il faut recenser les données nécessaires à un bon fonctionnement de l'entrepôt.

Il faut alors recenser les données disponibles dans les bases de production, toutes les données de production ne sont pas utiles dans l'entrepôt. Il faut aussi identifier les données supplémentaires requises et s'assurer la possibilité de se les procurer (achat de bases géographiques, démographiques,…).

L'examen des dimensions dans lesquelles les faits seront suivis doit donner lieu à une étude de l'unité de ces dimensions, de la granularité de ces faits. Par exemple, l'unité de temps doit-elle être le jour, la semaine, le mois, ou l'année ? Les produits sont-ils analysés par lot ou par marque ?

La variété des besoins, leur modularité peut entraîner un découpage de l'entrepôt en plusieurs parties : les Datamarts. Les Datamarts sont alimentés par un entrepôt et sont dédiés à une activité particulière pour un ou plusieurs services (le suivi des clients, la prévision des stocks). On peut mener l'analyse soit de façon ascendante (en commençant par les Datamarts), soit de manière descendant (déduction du Datamart à partir du Data Warehouse).

2.1.2. Coût de déploiement

Il faut une machine puissante, souvent une machine parallèle, spécialisée pour cette tache. Les tentatives de mixer à la fois l'informatique de décision et l'informatique de production au sein d'une seule machine sont souvent des échecs. Comme nous l'avons vu précédemment les utilisations sont trop différentes. La capacité de stockage doit être très importante.

Il faut noter que les prix de ces matériels ne cessent de décroître. En vertu de la loi de Moore, la puissance de stockage est multipliée par deux tous les deux ans, et cette tendance s'accélère même à prix constant. [20]

2.1.3 Bénéfices attendus

Il est possible de calculer les espérances de gain en prévoyant les performances du système. Il faut, pour cela, effectuer une étude exhaustive des services de l'entreprise demandeurs d'un environnement d'aide à la décision à partir de données. Pour chacun des services, il faut recenser des projets sur lesquels des bénéfices peuvent être retirés d'une telle implantation. Pour chacun des projets, il faut alors estimer les bénéfices attendus.

Prenons l'exemple du service de marketing direct qui veut augmenter le taux de réponses aux courriers qu'il envoi. A l'aide d'un entrepôt de données, il va mémoriser son activité et tenter d'établir des profils de sa clientèle. Si un outil de fouille de données permet de prédire avec un certain taux d'erreur, si un client répondra ou non au courrier, le service peut cibler son action. Avec des listes de clients bien ciblés, il peut réduire le nombre d'envois tout en conservant le même nombre de retour. L'économie peut alors être quantifiée et comparée aux coûts d'investissement.

Il est important de noter que, au vu des investissements nécessaires, il est fréquent de commencer par le développement de Datamart sur des projets clairement définis avant de passer à une généralisation à l'ensemble des services de l'entreprise.

L'étude préalable se conclue par la rédaction d'un cahier des charges comprenant la solution envisagée, le bilan du retour sur investissement et la décision d'implantation.

2.2 Modélisations des données du Data Warehouse

La conception d'un entrepôt est très différente de celle d'une base de données pour un système OLTP. Les concepts sont plus ouverts et plus difficiles a définir. De plus , les besoins des utilisateurs de l'entrepôt ne sont pas aussi clairs que ceux des utilisateurs des systèmes OLTP. Les modèles de données utilises dans la conception des systèmes transactionnels traditionnels ne sont pas adaptes aux requêtes complexes. En effet, les transactions dans les systèmes OLTP sont simples, alors que, dans les entrepôts, les requêtes utilisent beaucoup de jointures, demandent beaucoup de temps de calcul et sont de nature ad-hoc. Pour ce type d'environnement on a suggère une nouvelle approche de modélisation: les modèles multidimensionnels. [25]

2.2.1 Modélisation multidimensionnelle

La modélisation multidimensionnelle consiste à considérer un sujet analysé comme un point dans un espace à plusieurs dimensions. Les données sont organisées de manière à mettre en évidence le sujet analysé et les différentes perspectives de l'analyse.

Ce modèle conceptuel a été simplifié au maximum pour permettre au plus grand nombre d'utilisateurs d'appréhender l'organisation des données et de comprendre ce que le Data Warehouse mémorise. [23]

A- Eléments de modélisation multidimensionnelle

Le modèle multidimensionnel est une représentation conceptuelle du Data Warehouse sous forme d'un ensemble de cubes reliés entre eux et adaptés à l'analyse multicritères.

Afin de comprendre la modélisation multidimensionnelle**,** on va décrire trois concepts : le cube, le fait et la dimension.

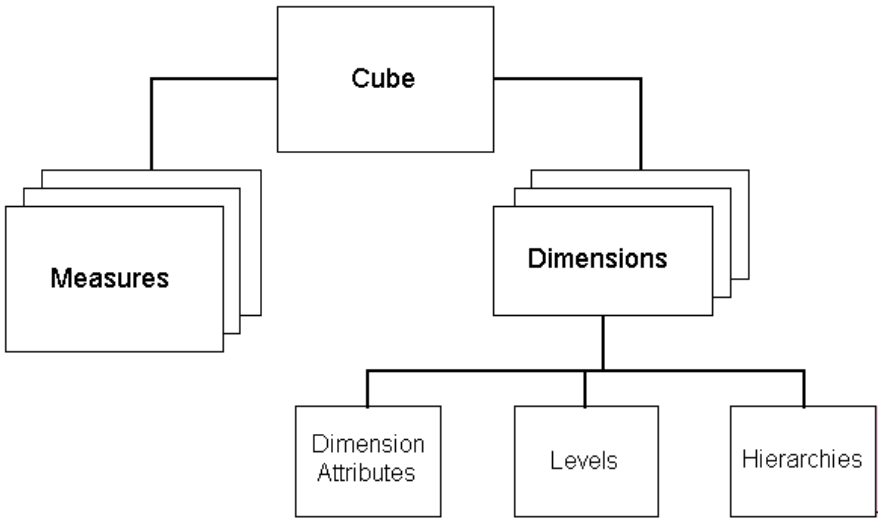


Figure 15 : Diagramme de la Modèle logique multidimensionnelle

A.1. Concept de fait

Le fait modélise le sujet de l'analyse. Un fait est une « observation du marché », il est formé de mesures correspondant aux informations de l'activité analysée.

Les mesures d'un fait sont numériques et généralement valorisées de manière continue [20][24]. Les mesures sont numériques pour permettre de résumer un grand nombre d'enregistrement, on peut les additionner, les dénombrer ou bien calculer le minimum ou le maximum. Les mesures sont valorisées de façon continue, elles sont aussi additives ou semi-additives afin de pouvoir les combiner au moyen d'opérations arithmétiques. [16]

A.2. Concept de dimension

Une dimension modélise une perspective de l'analyse. Le sujet à analyser, c'est-à-dire le fait, est analysé suivant différentes perspectives. Ces perspectives correspondent à une catégorie utilisée pour caractériser les mesures d'activité analysées.

Une dimension se compose de paramètres correspondant aux informations faisant varier les mesures de l'activité.

Chaque dimension est forme par un ensemble d'attributs et chaque attribut peut prendre différentes valeurs. Les attributs sont discrets, c'est-à-dire que les valeurs possibles sont bien déterminées et sont des descripteurs constants.

Les dimensions possèdent en général des hiérarchies associées qui organisent les attributs à différents niveaux pour observer les données à différentes granularités. Une dimension peut avoir plusieurs hiérarchies associées, chacune spécifiant différentes relations d'ordre entre ses attributs.[25]

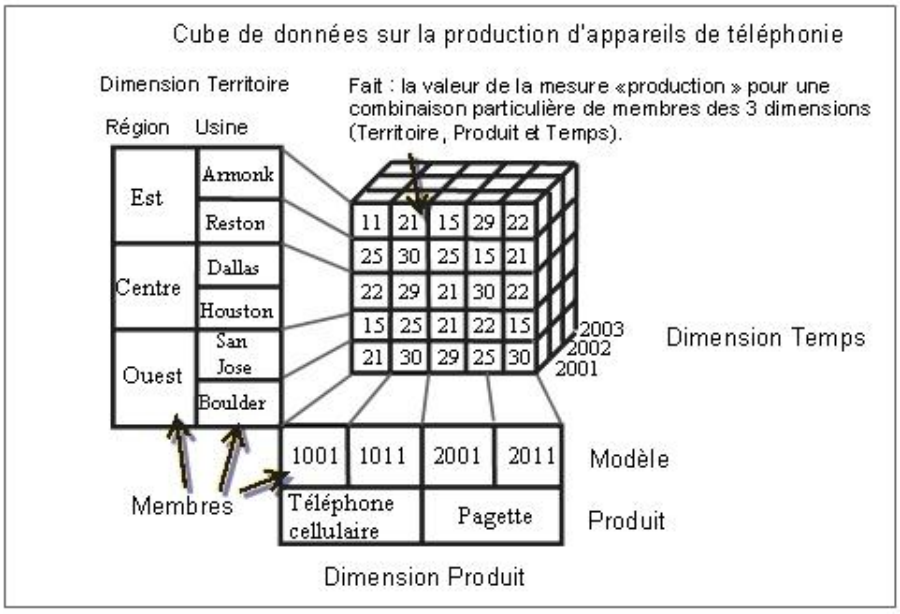


Figure 16 : schéma représentant le concept de cube multidimensionnel.[26]

Une **hiérarchie** organise les attributs d'une dimension selon une relation "*est_plus_fin*" conformément à leur niveau de détail.

les données sont généralement analysées en partant d'un faible niveau de détail vers des données plus détaillées pour "*forer vers le bas*". Pour définir ces différents niveaux de détail, chaque dimension est munie d'une (ou plusieurs) hiérarchie(s) des paramètres. La hiérarchie sert lors des analyses pour restreindre ou accroître les niveaux de détail de l'analyse. [16]

A.3. Cube

Le cube de données offre une abstraction très proche de la façon dont l'analyste voit et interroge les données.

Un *cube* est un ensemble de mesures organisées selon un ensemble de dimensions. C'est une représentation dimensionnelle d'un fait (sujet d'analyse) et de ses dimensions (critères d'analyse). Par exemple, un cube de ventes qui comprend les dimensions « Temps », « Produits » et « Magasins » et la mesure « Ventes en $ ». Pour chaque combinaison des trois dimensions (Magasins, produits, temps), on peut accéder à la mesure numérique associée au fait ventes (cellule non vide). Les interrogations s'interprètent souvent comme l'extraction d'un plan, d'une droite de ce cube (par exemple, lister les ventes du produit A), ou l'agrégation de données le long d'un plan ou d'une droite (par exemple, total des ventes du produit A). [24]

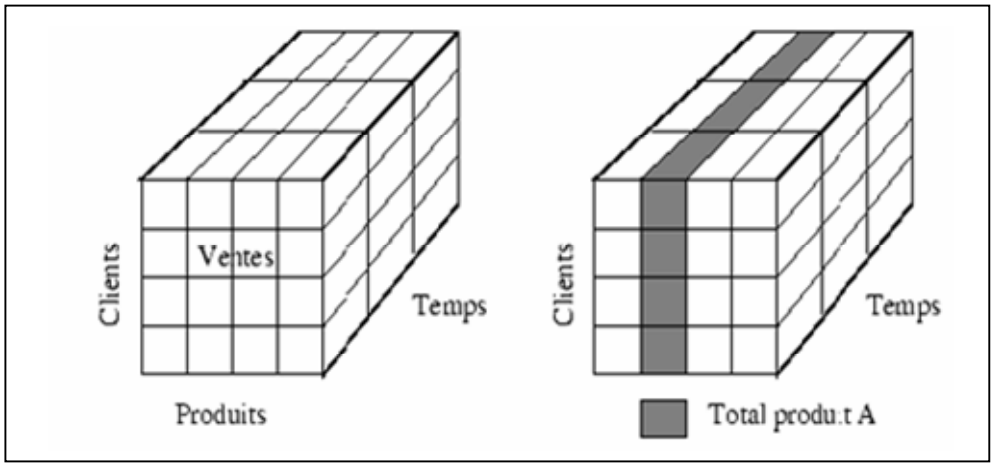


Figure 17 : Modèle en cube.

B. Schémas de modélisation

Partant du principe que les données sont des faits à analyser selon plusieurs dimensions, il est possible de réaliser une structure de données simple qui correspond à ce besoin de modélisation multidimensionnelle. Au niveau logique, cela peut se traduire par trois modèles différents : en étoile, en flocon de neige ou en constellation. [16]

B.1. Modèle en étoile

Le *modèle en étoile* tire son nom de sa configuration. Il contient un objet central, nommé table des faits. Cet objet central est connecté à un certain nombre d'objets de manière radiale, les tables de dimension. La table des faits, comme son nom l'indique, contient les faits. Les tables de dimensions contiennent les attributs définissant chacun des membres des dimensions. Elles sont dénormalisées.

Toutes les données décrivant les attributs de la dimension sont stockées à l'intérieur d'une même table de dimension. La définition des niveaux s'effectue à l'aide de champs de la table. Pour chacun des niveaux, un champ identifiant les attributs du niveau est obligatoire. [26]

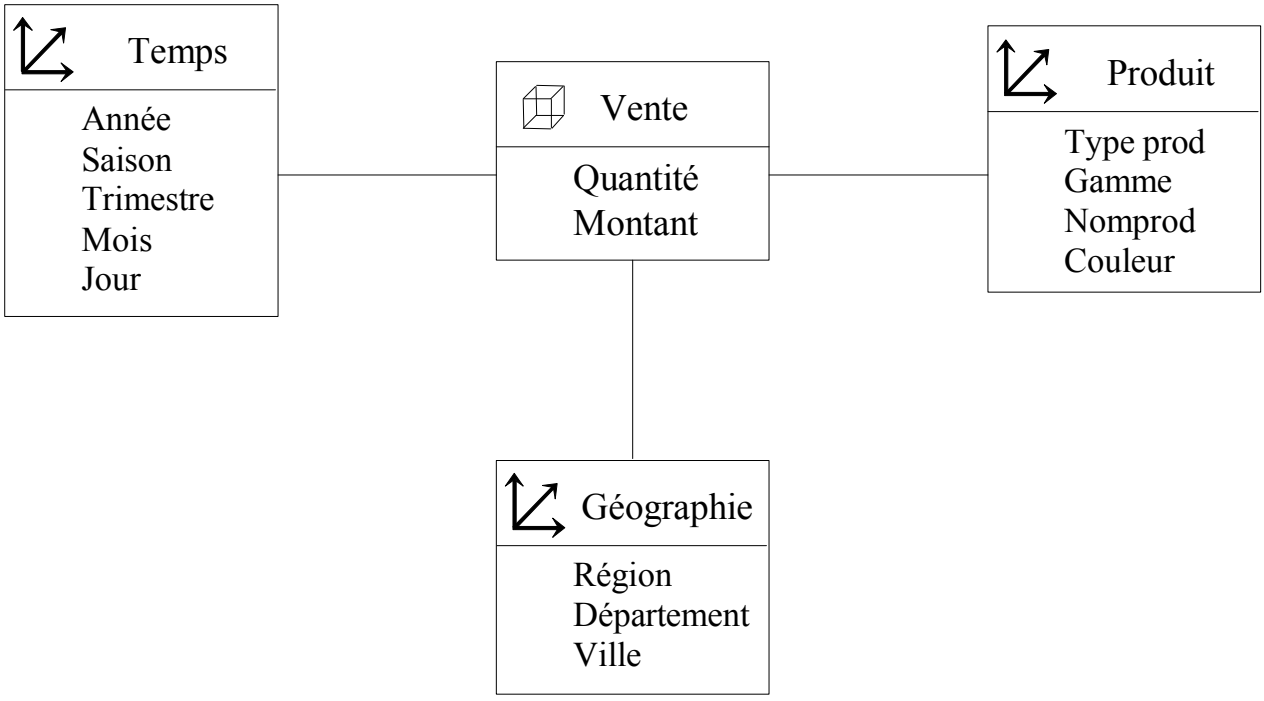


Figure 18 : exemple de modèle en étoile.

Les plus grands avantages de ce type de modèle sont la lisibilité et la performance:[23]

- *La lisibilité :* ce modèle est très parlant pour l'utilisateur et sa finalité est évidente. Il est naturellement orienté sujet et définit clairement les indicateurs d'analyse.
- *La performance :* les chemins d'accès à la base de donnés sont prévisibles.

B.2. Schéma en flocon

Une modélisation en flocon consiste à décomposer les dimensions du modèle en étoile en sous hiérarchies, le fait est conservé et les dimensions sont éclatées conformément à sa hiérarchie des paramètres. L'avantage de cette modélisation est de formaliser une hiérarchie au sein d'une dimension. Par contre, la modélisation en flocon rend pratiquement la présentation des informations plus complexe en terme de lisibilité et de gestion. [16] [23]

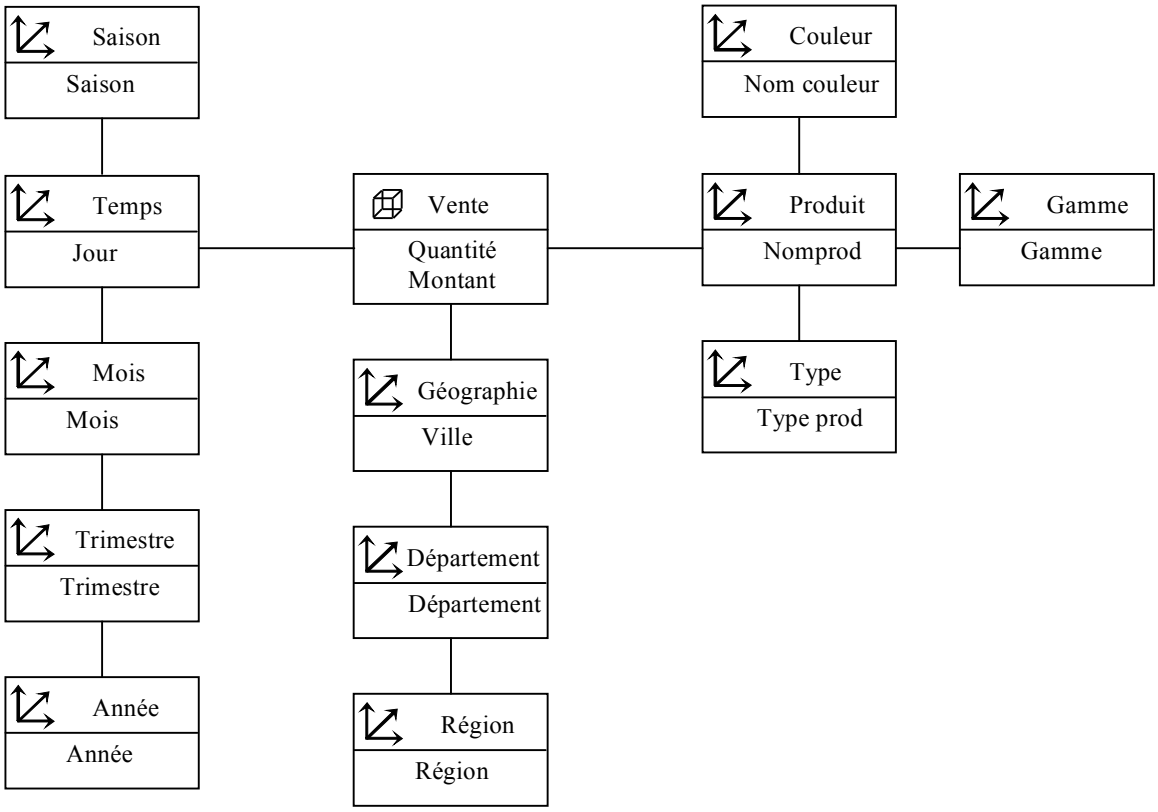


Figure 19 : exemple de modèle en flocon.

B.3. la modélisation en constellation

Il s'agit de fusionner plusieurs modèles en étoile qui utilisent des dimensions communes. Un modèle en constellation comprend donc plusieurs faits et des dimensions communes ou non.

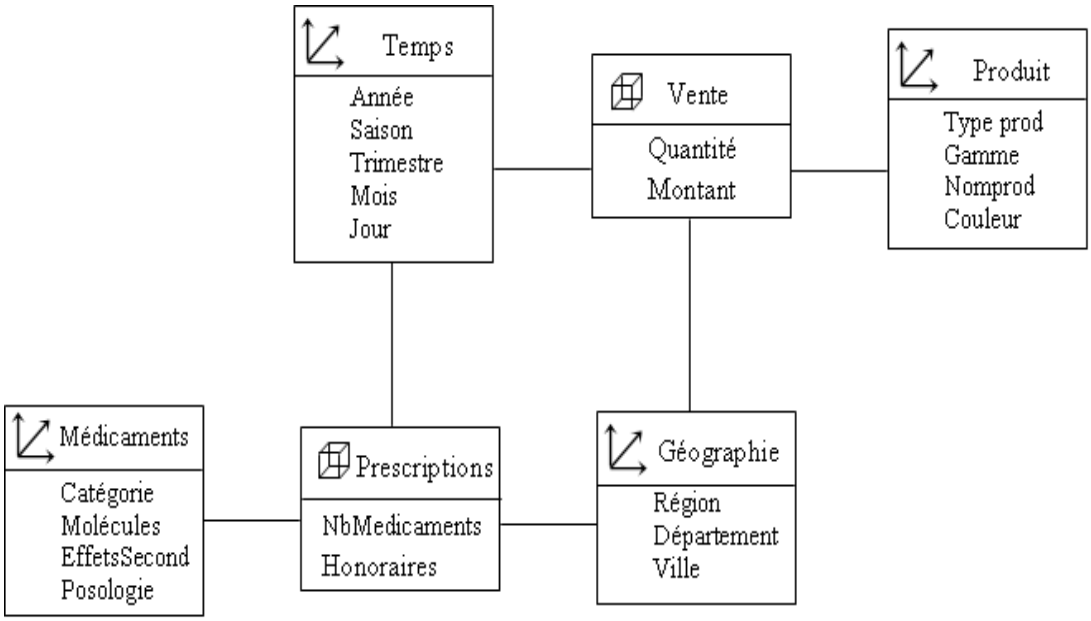


Figure 20 : exemple de modèle en constellation.

2.2.2 Modélisation logique [16]

Au niveau logique plusieurs possibilités sont envisageables pour la modélisation multidimensionnelle. Il est possible d'utiliser :

- un système de gestion de bases de données (SGBD) existant tel que les SGBD relationnels(ROLAP) ou bien les SGBD orientées objet (OOLAP),
- un système de gestion de bases de données multidimensionnelles (MOLAP).

A. ROLAP et OOLAP

L'approche la plus couramment utilisée consiste à utiliser un système de gestion de bases de données relationnelles, on parle de l'approche **ROLAP** ("*Relational On-Line Analytical Processing*"). Le modèle muldimensionnel est alors traduit de la manière suivante :

- chaque fait correspond à une table, appelée **table de fait**,
- chaque dimension correspond à une table, appelée **table de dimension**.

Ainsi, la table de fait est constituée d'attributs représentant les mesures d'activité et les attributs clés étrangères de chacune des tables de dimension. Les tables de dimension contiennent les paramètres et une clé primaire permettant de réaliser des jointures avec la table de fait.

Plus récemment, une autre approche s'appuie sur le paradigme objet ; on parle de l'approche **OOLAP** ("*Object On-Line Analytical Processing*").

Le modèle multidimensionnel se traduit ainsi :

- chaque fait correspond à une classe, appelée **classe de fait**,
- chaque dimension correspond à une classe, appelée **classe de dimension**.

B. MOLAP

Une alternative à ces deux approches consiste à utiliser un système multidimensionnel "*pur*" qui gère des structures multidimensionnelles natives ; on parle de l'approche **MOLAP** ("*Multidimensional On-Line Analytical Processing*").

Les structures multidimensionnelles natives utilisées sont des tableaux à *n* dimensions. Dans la littérature, les termes de cube, hypercube et table multidimensionnelle sont utilisés de manière interchangeable. Nous utiliserons le terme d'hypercube pour désigner des structures à deux, trois ou à plus de trois dimensions.

Cette approche permet de stocker les données de manière multidimensionnelle. L'intérêt est que les temps d'accès sont optimisés, mais cette approche nécessite de redéfinir des opérations pour manipuler ces structures multidimensionnelles.

3. PROCESSUS D'ALIMENTATION DU DW

L'alimentation est la procédure qui permet de transférer des données du système opérationnel vers l'entrepôt de données tout en les adaptant. La conception de cette opération est une tache assez complexe. Elle doit être faite en collaboration avec les administrateurs des bases de production qui connaissent les données disponibles et vérifient la faisabilité des demandes. [27]

Il est nécessaire de déterminer quelles données seront chargées, quelles transformations et vérifications seront nécessaires, la périodicité et le moment auxquels les transferts auront lieu.

L'alimentation du Data Warehouse passe par trois étapes successives (extraction, transformation, chargement).

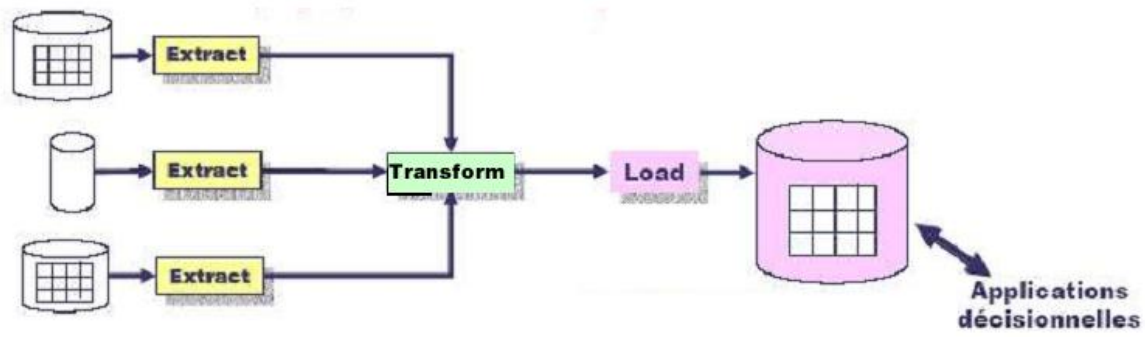


Figure 21 : Alimentation d'un data warehouse [28]

3.1 Extraction

L'extraction des données consiste à collecter les données utiles dans le système de production. Pour rafraîchir la base décisionnelle, il faut identifier les données ayant évolué afin d'extraire le minimum de données, puis planifier ces extractions afin d'éviter les saturations du système de production.

L'intégrité des données est indispensable et nécessite la synchronisation des différents processus d'extraction. Les problèmes liés à cette synchronisation peuvent être complexes, soit fonctionnellement, soit techniquement dans des environnements très hétérogènes. Un autre problème est de traiter les données externes. Il faut maintenir une surveillance du système d'information pour pouvoir les identifier et s'assurer que ce sont les bonnes données qui sont recensées. De plus, la forme des données externes, qui est souvent totalement anarchique accentue la difficulté. Pour être utiles, ces données nécessitent un reformatage pour pouvoir les incorporer dans une forme exploitable pour l'entreprise.

Enfin le troisième problème vient de l'apparition imprévisible de ces données qui les rend difficiles à capter. En conséquence, l'outil d'extraction doit attaquer toutes sortes de sources de données sans être perturbé et s'adapter aux futures. Pour extraire les données sources, il y a plusieurs technologies utilisables :

- des passerelles, fournies principalement par les éditeurs de bases de données. Ces passerelles sont généralement insuffisantes car elles sont mal adaptées aux processus de transformation complexes.
- des utilitaires de réplication, utilisables si les systèmes de production et décisionnel sont homogènes et si la transformation à appliquer aux données est légère.
- des outils spécifiques d'extraction (Oracle Warehouse Builder). Ces outils sont certainement la solution opérationnelle au problème de l'extraction, mais leur prix relativement élevé est un frein à leur utilisation dans les premières applications.

Souvent, une donnée élémentaire ne peut pas être trouvée pour une ou plusieurs colonnes de cible (diagramme de données) ou un attribut est ambigu. Dans ce cas, les réalisateurs du Data Warehouse reviennent aux utilisateurs des applications pour déterminer si un autre attribut peut être substitué, un autre point d'émission de données peut être trouvé, ou le schéma en étoile (diagramme de données) sera modifié.

L'extraction peut être de deux types : [29]

- **totale** : on extrait la globalité du flux à chaque traitement. Technique adaptée dans le cas d'une volumétrie réduite.
- **incrémentale ou "Delta"** : on ne traite que les enregistrements créés, modifiés ou supprimés depuis la dernière extraction.

3.2 Transformations et vérifications

La forme, le contenu des données de production ne convient pas toujours immédiatement au format choisi pour les données de l'entrepôt. Par conséquent, la deuxième étape du processus d'alimentation (transformation et vérification) est souvent nécessaire.

3.2.1 Format

Le format physique des données provenant de la production peut ne pas être adéquat avec le système hôte de l'entrepôt. Des transformations de type sont parfois nécessaires.

Les données pouvant provenir de serveurs différents dans des services différents, il est nécessaire d'uniformiser les noms et les formats des données manipulées au niveau de l'entrepôt.

3.2.2 Consolidation

Selon les choix des unités pour les dimensions, des opérations de consolidation devront accompagner le chargement des données.

3.2.3 Uniformisation d'échelle

Pour éviter de trop grandes dispersions dans les valeurs numériques, une homogénéisation des échelles de valeurs est utile. Ne pas la réaliser peut pénaliser les outils d'analyse et de visualisation et peut être simplement remplir inutilement les espaces de stockages (disques).

Des transformations qui permettent de mieux analyser les données sont aussi réalisées pendant la phase de chargement. Par exemple, la transformation de la date de naissance en âge, assure une plus grande lisibilité des données et permet de pallier les problèmes apparus avec l'introduction de la dimension temps.

Malgré les efforts réalisés pour assurer l'intégrité des données de production, des erreurs peut survenir, en particulier, lorsque les données proviennent de sources différentes.

Parmi les points à vérifier après les transformations, on peut citer :

- Erreurs de saisie ;
- intégrité de domaine ;
- informations manquantes.

Pour ces raisons, un annuaire spécialisé conserve toutes les informations (les métas donnés) au sujet du système d'information qui régit l'entrepôt. Cet annuaire contient :

- les règles de transformation, de vérification ;
- l'historique de l'entrepôt avec les dates de chargement initial et de rafraîchissement ;
- les utilisateurs, autorisations,… ;
- les adresses et descriptions des objets de l'entrepôt : tables, champs,…

En particulier chaque donnée est définie par une description précise et claire, son origine,… ;

3.3 Chargement

Le composant de base pour le chargement de données de Data Warehouse est le SGBD. Il doit être spécifiquement adapté aux caractéristiques de l'accès décisionnel.

Du fait de l'importance de l'historique, la structuration physique des données est également très importante.

Le SGBD apporte la transparence à l'évolution matérielle, l'indépendance, que ce soit au niveau des types et du nombre de processeurs, des disques ou des mémoires, ainsi que la transparence à l'évolution des systèmes d'exploitation.

3.4. Qu'est ce qu'un ETL ?

Le système d'alimentation a pour but de transformer les données primaires en informations stockées dans le Data Warehouse. Pour cela il est nécessaire de réaliser des outils pour suivre les fonctions d'alimentation dites les outils d'ETL (Extraction Transform Loading) qui doit pouvoir se connecter aux sources des données interne ou externe.

Un système ETL est définit comme tout système qui permet :[30]

- d'offrir un environnement de développement, des outils de gestion des opérations et de maintenance.
- de découvrir, analyser et extraire les données à partir de sources hétérogènes;
- de nettoyer et standardiser les données selon les règles d'affaires établies par l'entreprise;
- de charger les données dans un entrepôt de données dans et/ou les propager vers les datamarts.

3.4.1 Les générations d'ETL [31]

Le livre blanc d'Acta recense cinq générations d'outils d'ETL.

La première génération a vu apparaître des ETL qui générait du code Cobol pour assurer la transition entre les données sources et celles du Data Warehouse. Même si ces outils ont permis de faciliter la tâche, ils péchaient dans l'automatisation des environnements d'exécution. L'intervention humaine était nécessaire pour assurer l'intégration des données. La mise en place d'un ETL générateur de code nécessitait un budget prohibitif.

La seconde génération a bénéficié de l'avènement des technologies client / serveur pour les applications propriétaires pour proposer un nouveau langage : le SQL. Cependant, l'apparition de solutions intégrées comme les ERP a créé des difficultés pour extraire les données sources.

La version suivante a permis de pallier à ce problème en proposant des adaptateurs spécifiques. La diversité des données sources est mieux supportée : les données non structurées (ie extraites d'Internet) commencent à être intégrées et des adaptateurs spécifiques permettent d'intégrer les données issues des ERP.

Dans un contexte où les délais pour prendre des décisions se raccourcissent, le quatrième type permet d'intégrer les données au fil de l'eau (intégration instantanée) et en batch (en différé). Cela permet à l'entreprise d'avoir des résultats en temps réel et aux dirigeants de prendre plus

rapidement les décisions adéquates. De plus, les messages brokers apparaissent, cela permet en cas de suractivité des ressources, de mettre les données à intégrer dans une file d'attente. Le langage XML fait aussi son apparition et permet de réaliser des modifications qui seront immédiatement prises en compte par l'outil.

La dernière génération dispose d'API (Application Programming Interface) qui permettent aux données échangées de pouvoir être dirigées vers des cibles multiples. Ensuite, on a réussi à standardiser l'utilisation des fonctionnalités de la précédente génération. On utilise une intégration au fil de l'eau pour les données opérationnelles car les contraintes de l'activité l'exigent et car ces données ne représentent pas un volume trop important. Les données analytiques sont quant à elles intégrées en batch car le contexte est opposé au précédent.

3.4.2 Approches d'ETL [31]

A. ETL « indépendants »

Il s'agit des applications dont nous avons parlé précédemment, ils sont indépendants des systèmes sources et des systèmes décisionnels. Cette indépendance permet d'assurer une capacité d'adaptation aux divers systèmes d'information en présence.

Des solutions open sources existent, Talend est le « dernier né » et semble, d'après les premiers retours, très fiable et très intuitif. Ceci permet d'assurer une véritable concurrence au sein des ETL, ce qui pousse les éditeurs d'applications à améliorer leurs solutions.

B. ETL intégrés

Ces ETL fonctionnement comme des générateurs de code : la solution de gestion des bases de données (SGBD) se charge d'effecteur les transformations et les agrégations. Tout cela est traduit en un code spécifique pour assurer l'alimentation de l'entrepôt de données. Cependant, cette approche est quelque peu réductrice : l'ensemble des fonctionnalités d'un ETL classique n'y sont pas reprises. Le codage manuel est toujours nécessaire et il est difficile de faire évoluer ces solutions lors de l'apparition d'une nouvelle version du SGBD. Le principal inconvénient de ce modèle est la dépendance du client : le code généré pour l'intégration ne peut être interprété que par la « marque » du Data Warehouse. On peut y voir une sorte d'assujettissement des entreprises vis à vis de l'éditeur de SGBD.

4. ADMINISTRATION

Comme tout autre système informatique, un Data Warehouse s'administre. Dès la phase de conception de l'architecture, il faut penser à l'administration des données : Cette fonction est d'autant plus importante que le Data Warehouse évolue au fur et à mesure de son utilisation.

Avant d'exploiter le Data Warehouse, une procédure de certification et de publication doit être suivie. Elle va par exemple vérifier la cohérence des données au niveau des activités. En cas de succès, la nouvelle version de l'entrepôt sera publiée. Dans le cas contraire, c'est souvent le chargement complet qui est annulé et repoussé à une date ultérieure. En fait, le chargement peut être vu comme une très grosse et très longue transaction.

Après la publication de Data Warehouse, le mise en place de ce dernier doit s'accompagner de celle d'un référentiel de données, permettant de décrire, stocker et diffuser les méta-données associées.

Cette mise en place passe par l'organisation d'une fonction d'administration des données à plusieurs niveaux, par la définition de normes et de procédure d'administration des référentiels.

4.1 Le référentiel du Data Warehouse

Le référentiel du Data Warehouse est l'ensemble des outils nécessaires à la mise en œuvre de la fonction d'administration de données.

Les objectifs de l'administration de données du Data Warehouse sont :

- Assurer la cohésion du système.
- Respecter la cohérence et la fiabilité des informations.
- Unifier la représentation des données.
- Respecter la cohérence des concepts.
- Vérifier la non redondance des informations.
- Déterminer la bonne période pour le rafraîchissement de données de Data Warehouse.
- Unifier la saisie et le stockage des informations.

4.2 Sécurité

Le Data Warehouse a pour vocation de laisser à l'utilisateur une totale autonomie en ce qui concerne la recherche et l'analyse des données. Cette liberté doit cependant être souvent restreinte, notamment pour des raisons de sécurité. L'outil doit donc permettre d'adapter l'environnement de travail à l'utilisateur qui s'y connecte selon sa fonction et donc ses droits. Afin de mener à bien cette politique de sécurité, les notions d'utilisateur et de groupe d'utilisateur sont indispensables. Cette notion doit néanmoins être dissociée de la notion d'utilisateur du serveur de données. Il est en effet préférable que l'utilisateur se connecte directement à l'outil d'aide à la décision et accède de manière transparente au serveur de données sans en connaître le nom d'utilisateur et le mot de passe.

5. EXPLOITATION DE DONNEES DU DW

Finalement, un Data Warehouse est destiné à être consulté et, donc, à aider l'utilisateur dans l'exécution de ses tâches. Dans de nombreuse des cas, le Data Warehouse constitue la source principale de données pour un système d'aide à la décision. A ce niveau, une distinction est souvent opérée entre les méthodes d'analyse basées sur la vérification, qui permettent à l'utilisateur de tester une hypothèse à partir de données issues du Data Warehouse, et celles basées sur la découverte, qui visent à mettre en lumière de nouveaux modèles de connaissances.

L'exploitation des données de Data Warehouse passe par deux étapes qui sont :

- La détermination de l'espace d'analyse ;
- L'analyse les données de cet espace d'analyse ;

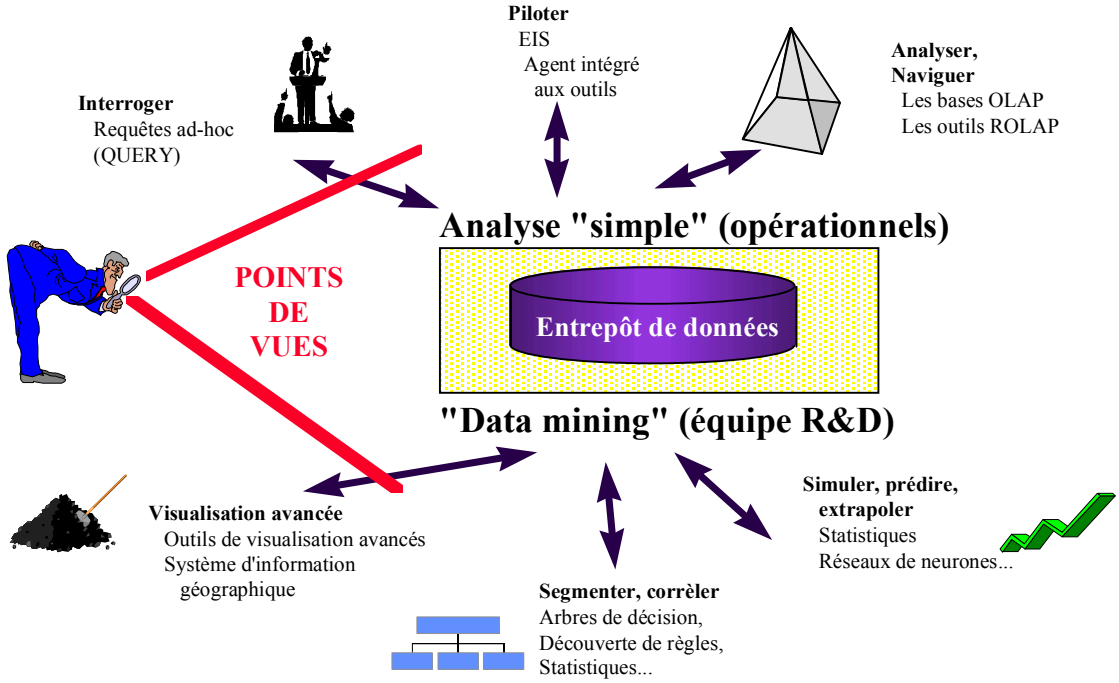


Figure 22 : data warehouse - points de vue pour les utilisateurs [8]

5.1 L'ESPACE D'ANALYSE

L'analyse de données ne se fait pas sur toutes les données de Data Warehouse, elle se fait sur seulement la partie contenant les données intéressantes pour cette analyse, cette partie de données s'appelle espace d'analyse.

Les espaces d'analyse utilisées sont :

- les Datamarts;
- les cubes de données;
- les tableaux.

5.1.1 Datamart

Le Datamart est une base de données moins coûteuse que le Data Warehouse, et plus légère puisque destinée à quelques utilisateurs d'un département. (**Figure 22**)

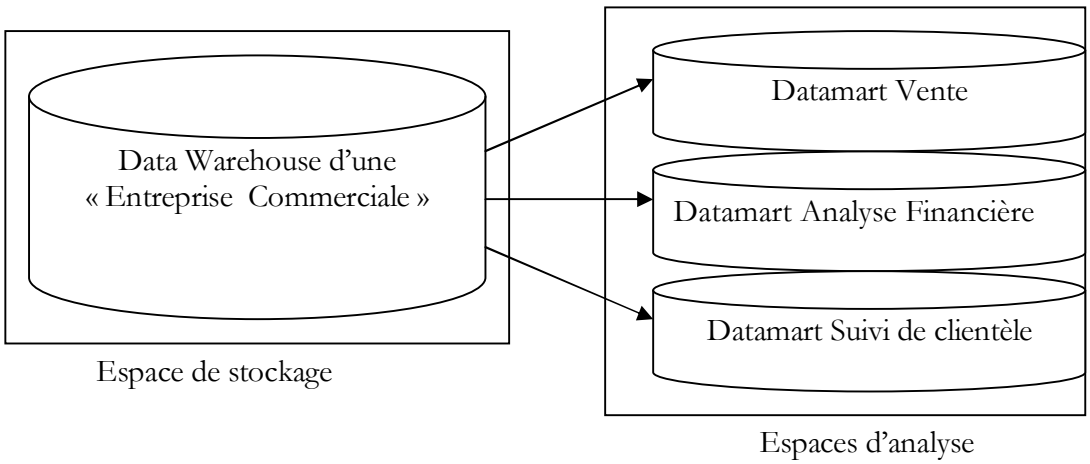


Figure 23 : Les Datamarts d'un Data Warehouse.

C'est une petite structure très ciblée et pilotée par les besoins utilisateurs. Il a la même vocation que le Data Warehouse (fournir une architecture décisionnelle), mais vise une problématique précise avec un nombre d'utilisateurs plus restreint. En général, c'est une petite base de données (Relationnelle ou Multidimensionnelle) avec quelques outils, et alimentée par un nombre assez restreint de sources de données.

5.1.2 Cubes de données

Il s'agit d'une modélisation multidimensionnelle des données facilitant l'analyse d'une quantité selon différentes dimensions (nombre de dimensions supérieure à deux)**.**

Le Cube de données peut être extrait d'un Datamart ou directement de Data Warehouse. (**Figure** 23)

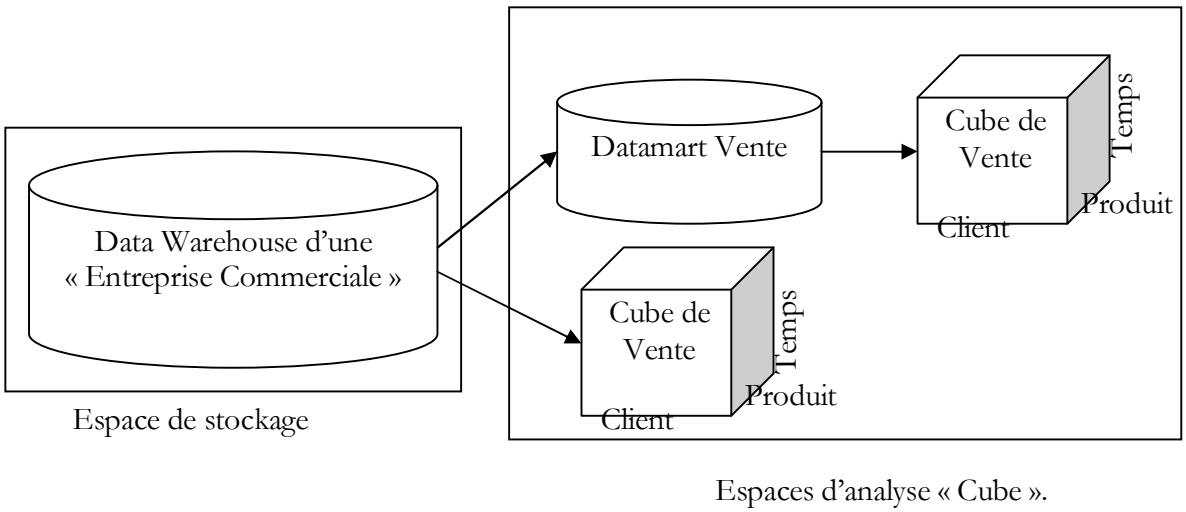


Figure 24 : Cube de données d'un Data Warehouse.

5.1.3 Tableaux (tableur)

Si le nombre de dimensions pour notre analyse est égal à deux (cas d'une modélisation multidimensionnelle), le cube de données devient un tableau.

On peut aussi déterminer les tableaux de données à partir d'un Datamart ou bien d'un Data Warehouse directement. (**Figure** 24)

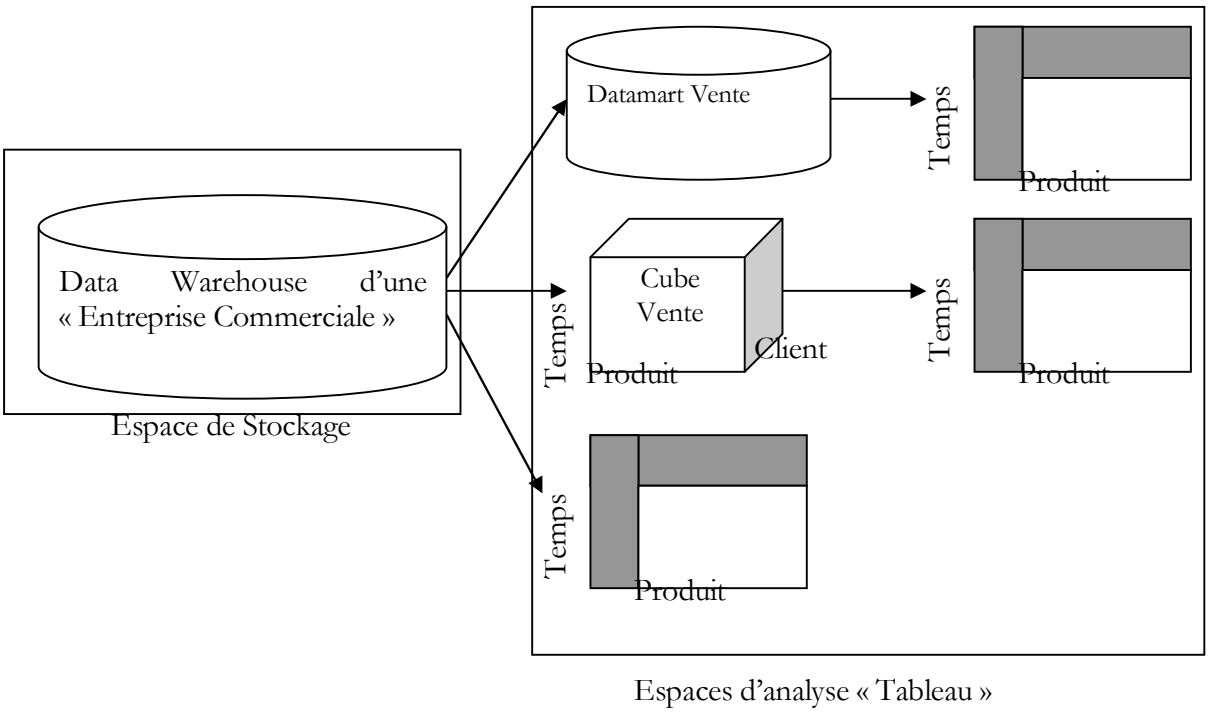


Figure 25 : Tableau d'un Data Warehouse.

5.2 L'ANALYSE DE DONNEES SELECTIONNEES

Pour exploiter les données entreposées, nous avons plusieurs outils d'analyse, de visualisation et de découverte des données. Parmi ces outils, on cite :

5.2.1 Les requêteurs

Le requêteur sera idéal pour les données qualitatives et pour toute analyse impromptue nécessitant l'autonomie de l'utilisateur (cas rencontré fréquemment pour le marketing ou la gestion du personnel).

5.2.2 L'analyse multidimensionnelle OLAP

Les outils OLAP (On Line Analytical Process) reposent sur une base de données multidimensionnelle, destinée à exploiter rapidement les dimensions d'une population de données en utilisant le cube de données.

Deux versions d'OLAP existent actuellement. Les outils MOLAP (Multidimensionnel OLAP) d'une part qui s'appuient sur une base de données multidimensionnelle et les outils ROLAP

(Relationnel OLAP) d'autre part, qui représente leur équivalent sur une base de données relationnelle.

A. Les outils « MOLAP »

MOLAP est conçue exclusivement pour l'analyse multidimensionnelle, avec un mode de stockage optimisé par rapport aux chemins d'accès prédéfinis. Ainsi, toute valeur d'indicateur associée à l'axe temps sera pré-calculée au chargement pour toutes ses valeurs hebdomadaires, mensuelles,...

En effet, MOLAP repose sur un moteur spécialisé, qui stocke les données dans un format tabulaire propriétaire (cube). Pour accéder aux données de ce cube, on ne peut pas utiliser le langage de requête standard SQL, il faut utiliser une API spécifique.

Le marché des bases MOLAP étant plus réduit, il est plus difficile pour les éditeurs qui le représentent d'investir sur de telles évolutions.

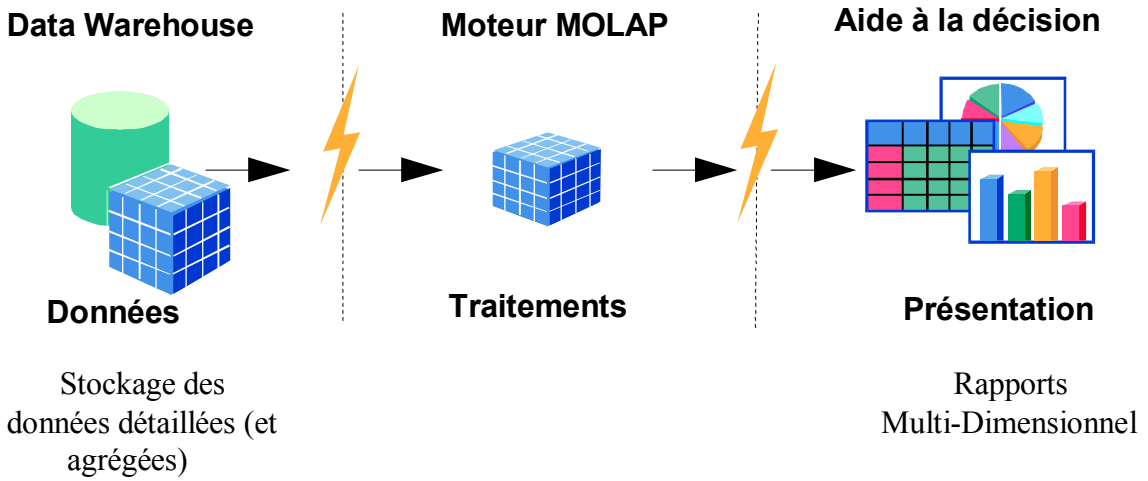


Figure 26 : Architecture d'un produit MOLAP.

B. Les outils ROLAP

Les outils ROLAP superposent au dessus des SGBD/R bidimensionnels un modèle qui représente les données dans un format multidimensionnel. Ces produits diminuent sensiblement le coût lié à la mise en œuvre d'un serveur de base de données multidimensionnelle supplémentaire. Au travers des méta-données, ils permettent de transformer l'analyse multidimensionnelle demandée par l'utilisateur en requêtes SQL. Pour cela, ces outils s'appuient pour la plupart sur une modélisation particulière des données, distinguant les axes d'analyse et les faits à observer. On parlera notamment de modèle en étoile et de modèle en floconou encore des techniques de définition physique d'agrégations. Ceci nous oblige à définir le modèle en fonction de l'outil à utiliser et des analyses à mener. Mais, le ROLAP est un gage de performance et de cohérence lors de l'utilisation de ce type de produits.

Cette contrainte exige un travail important des équipes informatiques et donc enlève beaucoup à l'intérêt d'utiliser un SGBD Relationnel comme support de stockage pour l'analyse multidimensionnelle.

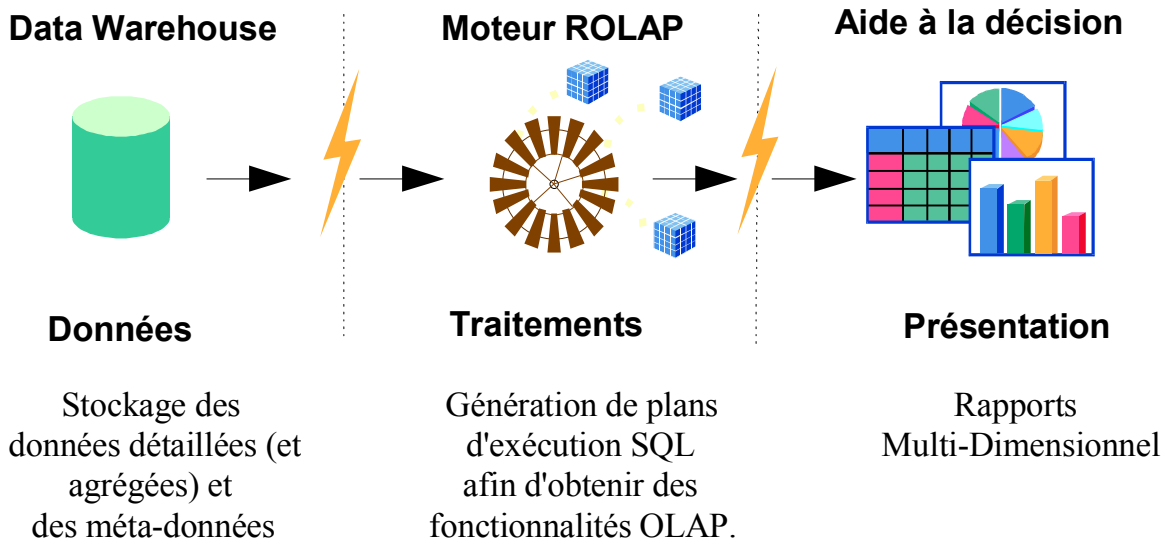


Figure 27 : Architecture d'un produit ROLAP.

C. Méthodes de navigation d'OLAP

Une fois cette structure multidimensionnelle établie, l'outil OLAP propose des méthodes de navigation dans les données, comme le "Drill-down" pour aller vers les informations détaillées dans une hiérarchie, le "Slice and dice" pour changer d'axe d'analyse.

C.1 Drill-down, Drill-up

Ce mécanisme est totalement basé sur la notion de hiérarchie. Chacun des axes d'analyse se décompose en attributs reliés entre eux par des relations père/fils. Une dimension doit normalement pouvoir comporter plusieurs hiérarchies. Par exemple, la dimension " produits " peut contenir une hiérarchie " Marque-Article " et une hiérarchie « Secteur-Segment-Article ».

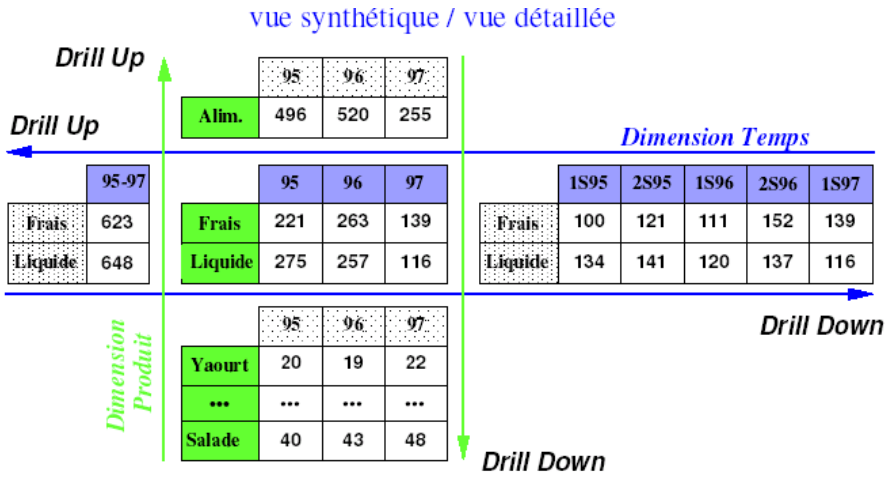


Figure 28 : Drill-down et Drill-up (Rollup).

C.2 Rotation : CUBE

Cette méthode de navigation permet de représenter les données d'un CUBE selon les dimensions qui nous s'intéresse pour l'analyse.

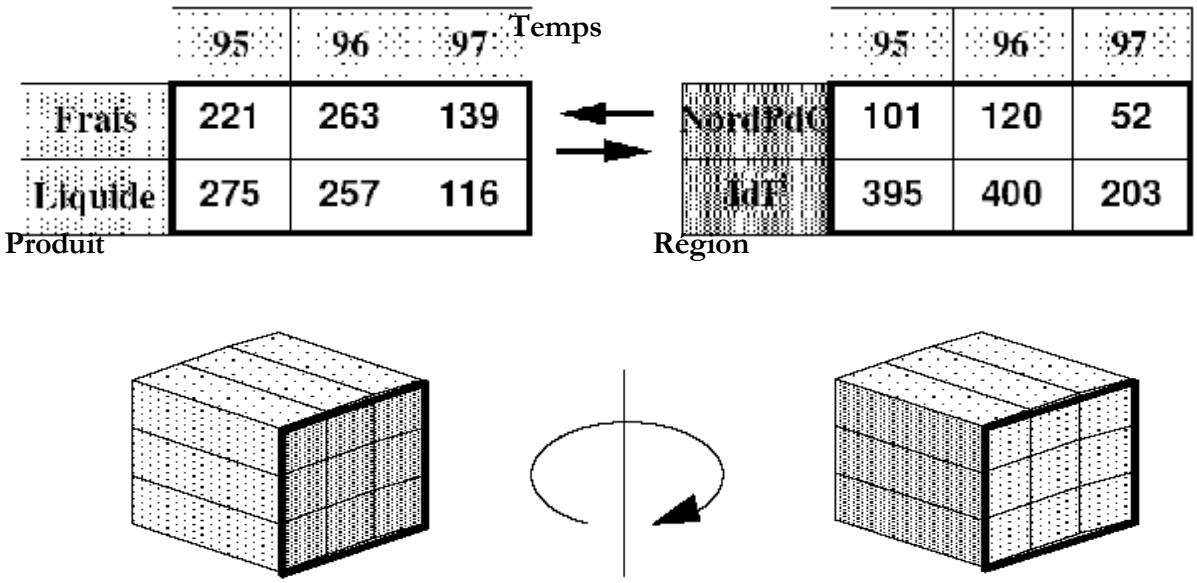


Figure 29 : Rotation du CUBE.

C.3 Slice and dice

Cette méthode consiste à filtrer une dimension selon une valeur ou une plage de valeurs afin de retenir et de se concentrer sur l'analyse d'une partie de données.

Si on ne s'intéresse que pour les ventes durant l'année 1996, on fait une coupe du cube de telle sorte qu'on ne trouve dans la dimension temps que l'année 1996 (**figure 29**).

		1995	1996	1997
Frais	NordPdc	220	265	284
	Idf	225	245	240
Liquide	NordPdc	163	152	145
	Idf	187	174	184

		1996
Frais	NordPdc	265
	Idf	245
Liquide	NordPdc	152
	Idf	174

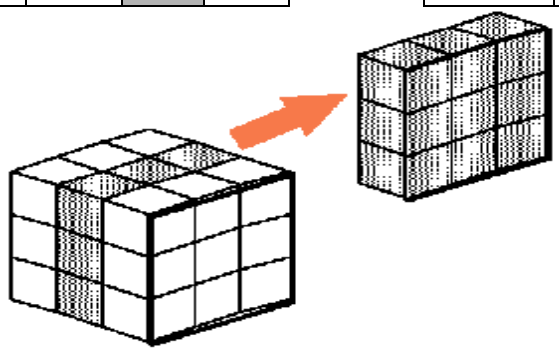


Figure 30 : Coupe du cube (Slice and dice).

Si on ne s'intéresse que pour les ventes du produit liquide, on fait une coupe du cube de tel sorte qu'on ne choisi que les produits liquides de la dimension Produit.

C.4 Data Surfing

Le data surfing est la possibilité laissée à l'utilisateur de circuler librement, de manière intuitive et ergonomique dans un modèle dimensionnel, au delà d'un simple « Drill-down / Drill-up» ou « Slice and dice ». L'utilisateur peut alors modifier dynamiquement ses axes d'analyse ou appliquer un nouveau filtre à ses données.

5.2.3 Les tableaux de bord & Reporting

Simple visualisation des données, les outils de visualisation sont très importants dans le processus de décision et peuvent intervenir à plusieurs niveaux. Ils sont utiles pour :

- découvrir de nouvelles informations, parce qu'une représentation permet de repérer plus simplement des anomalies. Dans ce cas, ils sont intégrés dans les outils d'analyse et doivent supporter des opérations comme comparer, modifier les échelles, …
- présenter des résultats, parce qu'un graphique est plus accessible qu'un tableau de chiffres ;

5.2.4 Le data mining: *(Méthodes d'analyse basées sur la découverte).*

Nous avons beaucoup de définition du Data Mining, parmi ces définitions on cite:

- Fouille de données historisées pour rechercher des règles d'évolution, de comportement,…
- Analyse et compréhension du passé pour la prédiction du futur.
- Exploration et analyse, par des moyens automatiques ou semi-automatiques, d'une masse importante de données dans le but de découvrir des tendances cachées ou des règles significatives.
- Un ensemble de techniques permettant d'extraire des modèles d'une base de données historisées afin de décrire le comportement actuel et/ou de prédire le comportement futur d'un procédé. Le Data Mining, en français la Fouille de données ou exploration de données se fait donc sur un ensemble de techniques d'extraction de connaissances appliquées aux bases de données. L'extraction s'opère par induction, c'est-à-dire que l'on généralise un peu abusivement les données afin de découvrir des propriétés générales, souvent quantifiées par des statistiques.

Les connaissances sont exprimées sous forme des modèles présentés à l'utilisateur pour examen (**Figure 30**). Les modèles peuvent être des modèles de calculs (des équations) ou des modèles logiques (règles d'induction et de déduction).

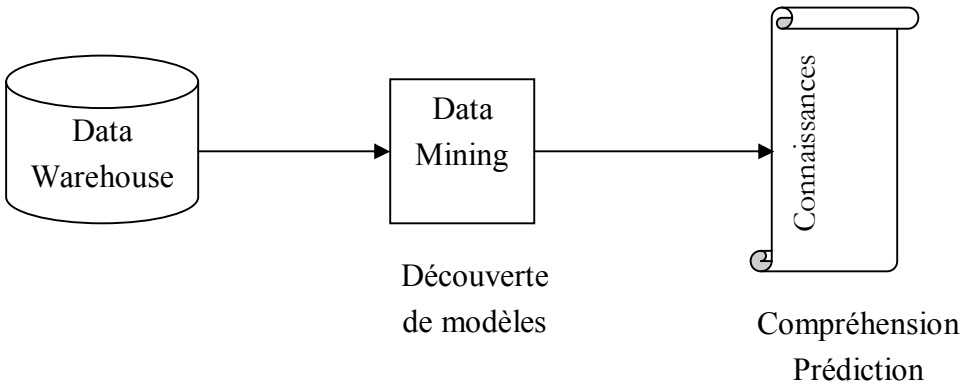


Figure 31 : Le processus de Data Mining.

6. CONCLUSION

Dans ce chapitre, nous nous sommes intéressés à trouver le diagramme de données du DW qui répond aux différents besoins de l'entreprise suivi par une longue procédure qui permet d'extraire et de transformer les données sources avant de les charger dans le DW. Après le chargement initial de données, l'administrateur du DW doit trouver la bonne période de rafraîchissement sans perturber les applications sources. On terminera la partie par « Exploitation de données » donne une vue globale sur chaque méthode utilisées pour exploiter les données du Data Warehouse.

On peut englober les phases de construction de DW citées au ce chapitre dans le tableau suivant :

Phase	**Etape**	**Description**
Conception	Etude préalable	Déterminer les besoins utilisateurs.
	Modélisation	Le modèle en étoile du Data Warehouse.
Processus d'alimentation	Extraction	Accès aux différentes sources de données.
	Transformation	Gestion des inconsistances de données sources
		Unification des formats de données.
		Détection des valeurs non valides.
	Chargement	Stocker les données au niveau de DW
Administration	Administration	La sécurité du Data Warehouse
		La bonne période de rafraîchissement du DW

Exploitation de données	Interrogation et rapports	Requête sur des données de détail et peu consolidées
	OLAP	Analyse, détection de problèmes et opportunités
	Data Mining	Découverte de tendances cachées, règles significatives

Tableau 3 : Les phases de construction du DW.

CHAPITRE III 3

Agent et Système Multi-Agents (SMA)

CHAPITRE III : AGENT ET SYSTEME MULTI-AGENTS (SMA)

1. INTRODUCTION

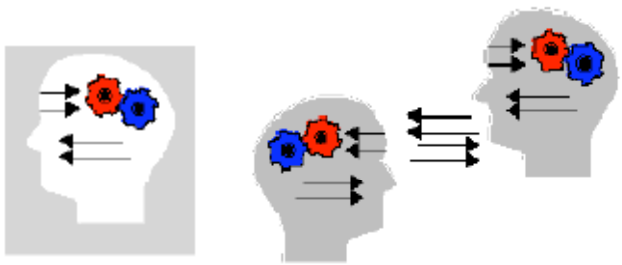
Dans ce chapitre, nous présentons les concepts d'agent et systèmes multi-agents. Avant de commencer, il serait intéressant de comprendre quelques concepts comme l'intelligence artificielle distribuée (IAD) et ses axes fondamentaux dans la recherche. Nous présentons par la suite l'évolution de l'aspect individuel (le comportement d'un agent seul) vers l'aspect collectif (son comportement dans une société d'agents).

2. L'INTELLIGENCE ARTIFICIELLE DISTRIBUEE

2.1. Définition de l'IAD

L'expression Intelligence Artificielle (IA) est employée pour la première fois (1955-1970) par John McCarthy. Il fonde l'Intelligence Artificielle sur le postulat mécaniste qui veut que toute activité intelligente soit modélisable et reproductible par une machine. « L'IA a pour but de faire exécuter par l'ordinateur des tâches pour lesquelles l'homme, dans un contexte donné, est aujourd'hui meilleur que la machine».

Mais l'évolution des domaines d'application de l'intelligence artificielle (IA) pour recouvrir des domaines complexes et hétérogènes a montré les limites de l'approche classique de L'IA qui s'appuie sur une centralisation de l'expertise au sein d'un système unique.



Métaphore de l'IA : Le penseur isolé

Métaphore de l'IAD : la communauté de penseurs

Figure 32 : La philosophie de l'IA et IAD

Les travaux menés au début des années 70 pour palier aux insuffisances de l'IA ont contribué à la naissance d'une nouvelle discipline : l'Intelligence Artificielle Distribuée. Donc l'IAD a pour but de remédier aux insuffisances de l'approche classique de l'IA en proposant la distribution de l'expertise sur un groupe d'agents devant être capables de travailler et d'agir dans un environnement commun et résoudre des problèmes complexes. D'où la naissance de notions nouvelles en IA, telles que la coopération et la coordination. **[32].**

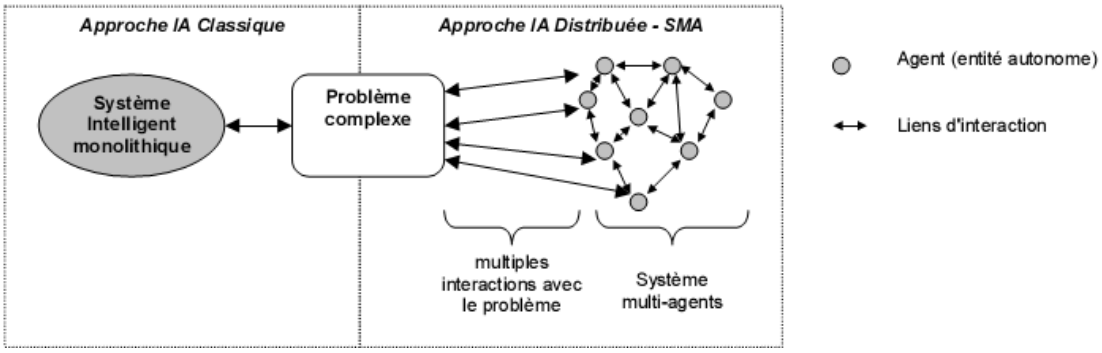


Figure 33: Comparaison entre les approches IA et IAD / SMA [33]

2.2. Thèmes de recherche de l'IAD

Après avoir défini l'IAD, nous présentons les trois axes fondamentaux dans la recherche en IAD[32]:

- Les Systèmes Multi-Agents (SMA) : Il s'agit de faire coopérer un ensemble d'agents dotés d'un comportement intelligent et de coordonner leurs buts et leurs plans d'actions pour la résolution d'un problème.
- La Résolution Distribuée des Problèmes (RDP) : Elle s'intéresse à la manière de diviser un problème particulier sur un ensemble d'entités distribuées et coopérantes. Elle s'intéresse aussi à la manière de partager la connaissance du problème et d'en obtenir la solution.
- L'Intelligence Artificielle Parallèle (IAP) : Elle concerne le développement de langages et d'algorithmes parallèles pour l'IAD. L'IAP vise l'amélioration des performances des systèmes d'intelligence artificielle sans toutefois s'intéresser à la nature du raisonnement ou au comportement intelligent d'un groupe d'agents.

La différence notable entre la RDP et les SMA est que la RDP possède une approche descendante («top-down») et les SMA une approche ascendante («bottom-up»).[34]

3. LE CONCEPT D'AGENT

3.1. Définitions

Si on se réfère à la définition de Ferber [35] : Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle même et sur son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

Pour Jennings [36], un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu.

A partir de ces définitions, l'agent est définit comme une entité :

a) qui est capable d'agir dans un environnement,
b) qui peut communiquer directement avec d'autres agents,
c) qui est mue par un ensemble de tendances (sous la forme qu'elle cherche à optimiser),
d) qui possède des ressources propres,
e) qui est capable de percevoir (mais de manière limitée) son environnement,
f) qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
g) qui possède des compétences et offre des services,
h) qui peut éventuellement se reproduire,
i) dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

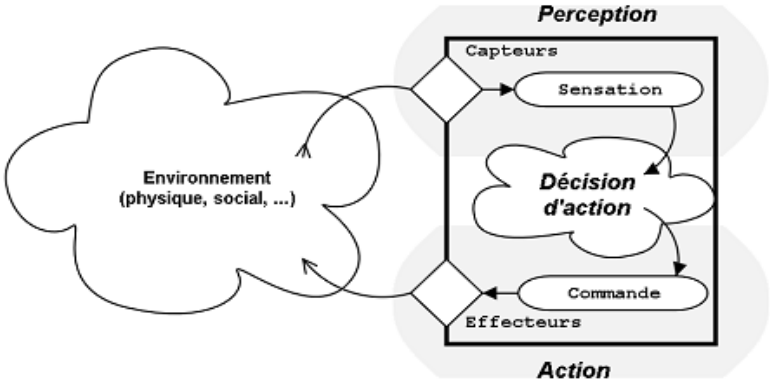


Figure 34 : la structure d'un agent [37]

3.2. Caractéristiques d'agent

Un agent doit posséder les caractéristiques suivantes : [38]

- Situé : l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemples: systèmes de contrôle de processus, systèmes embarqués, etc.… ;
- Autonome : l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;
- Proactif : l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au "bon" moment;
- Social : l'agent doit être capable d'interagir avec les autres agents quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.
- Coopération : capable de faire la coordination avec d'autres agents pour atteindre un objectif commun.
- Apprentissage : L'agent est capable d'apprendre et d'évoluer en fonction de cet apprentissage, il est capable de changer de comportement (en fonction des expériences passées).
- Mobilité : l'agent est capable de se déplacer vers un autre environnement
- Rationalité : l'agent est capable d'agir en fonction de ses objectifs internes et ses connaissances.

Bien entendu, selon les applications, certaines propriétés sont plus importantes que d'autres. Il peut même s'avérer que pour certains types d'applications, des propriétés additionnelles soient requises. Cependant, la présence des propriétés que l'on vient de voir comme l'autonomie, la flexibilité et la sociabilité donne naissance au paradigme agent tout en le distinguant des systèmes conventionnels comme les systèmes distribués, les systèmes orientés objets et les systèmes experts.[**38]**

3.3. Typologie des agents [39]

La granularité des agents impliqués dans une application varie selon deux écoles coexistantes aujourd'hui : l'école **cognitive** et l'école réactive. Suivant le type d'agent utilisé, on parlera de systèmes cognitifs ou de systèmes réactifs [32].

3.3.1. Agents cognitifs

Les systèmes d'agents cognitifs sont fondés sur la coopération d'agents capables à eux seuls d'effectuer des opérations complexes. Un système cognitif comprend un petit nombre d'agents qui disposent d'une capacité de raisonnement sur une base de connaissances, d'une aptitude à traiter des informations diverses liées au domaine d'application, et d'informations relatives à la gestion des interactions avec les autres agents et l'environnement. Chaque agent est assimilable, suivant le niveau de ses capacités, à un système expert plus ou moins sophistiqué, on parle d'agent de forte granularité. Ils regroupent plusieurs sous-types d'agents définis de la façon suivante [39]:

La figure 35 positionne globalement les différents types d'agents cognitifs selon leurs degrés d'autonomie, d'adaptation et de coopération.

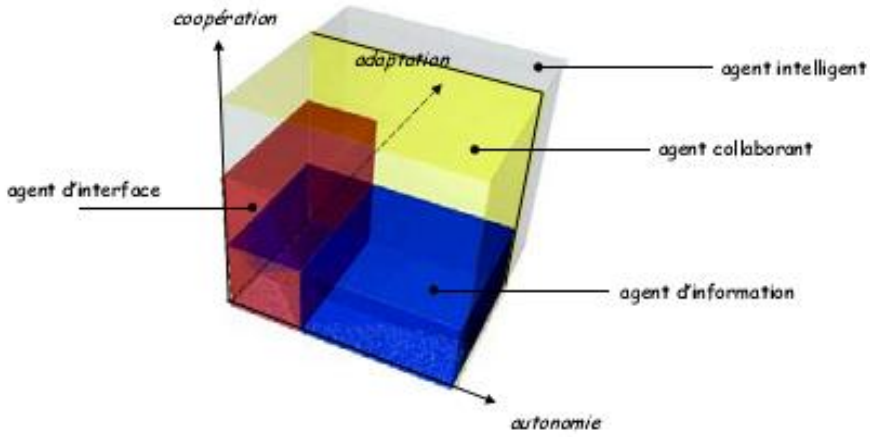


Figure 35 : Classification des principaux agents cognitifs

- **Agents intelligents** : qui combinent les trois caractéristiques (autonomie, coopération, adaptativité) à leur plus haut niveau. Ils sont capables de négocier avec d'autres agents et d'acquérir ou de modifier leurs connaissances. Ainsi, ils planifient leurs actions. Le terme d'agent intelligent est souvent utilisé pour caractériser des agents dotés de la capacité d'apprentissage.
- **Agents collaborants** : ce sont des agents cognitifs non apprenants. Ils sont donc à la fois fortement autonomes et coopérants, mais leur adaptativité ne va pas jusqu'à l'adaptation et à l'acquisition des connaissances. Les agents collaborants sont surtout utilisés dans les domaines qui nécessitent une décentralisation comme par exemple la maintenance de réseau.
- **Agents interfaces :** ils sont souvent utilisés dans les logiciels de bureautique. Leur fonction consiste à capturer les actions de l'utilisateur (le plus souvent les actions sur le clavier, la souris, la capture de la voix ou de l'expression du visage par utilisation d'une Webcam,

caméra dédiée aux communications Internet). Ces agents possèdent en général une capacité de coopération limitée à l'échange d'informations concernant les actions de l'utilisateur.

- **Agents informations :** ils sont dédiés à la recherche d'information, principalement sur Internet. Ces agents possèdent une grande autonomie; ils agissent seuls, soit en fonction d'un calendrier, soit en fonction d'un manque d'information, soit en fonction d'une nouvelle disponibilité d'information. Ils sont capables d'adapter leurs fonctionnements en fonction du besoin de l'utilisateur ou de la quantité ou pertinence de l'information (par exemple, si un nouveau site propose des informations plus pertinentes, ce site sera alors privilégié pour les recherches futures). Toutefois, ces agents agissent le plus souvent de manière isolée, ce qui peut entraîner un problème de redondance d'informations.

3.3.2. Agents réactifs

Les défenseurs de cette approche partent du principe suivant : dans un système multi-agents, il n'est pas nécessaire que chaque agent soit individuellement "intelligent" pour parvenir à un comportement global intelligent. En effet, des mécanismes simples de réactions aux événements peuvent faire émerger des comportements correspondant aux objectifs poursuivis. Cette approche propose la coopération d'agents de faible granularité (*fine grain*) mais beaucoup plus nombreux.

Les agents réactifs sont de plus bas niveau, ils ne disposent que d'un protocole et d'un langage de communication réduits, leurs capacités répondent uniquement à la loi stimulus/réponse.

3.3.3. Les agents cognitifs Vs réactifs

Dans le tableau (Tableau 4), nous résumons les différences entre les deux approches. C'est l'école cognitive qui, jusqu'à maintenant, a donné lieu aux applications les plus avancées. La complexité des systèmes réactifs exige le développement de nouvelles théories dans le domaine de la coopération, de la communication et de la compréhension de nouveaux phénomènes telle que l'émergence.

Systèmes d'agents cognitifs	Systèmes d'agents réactifs
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son histoire
Agents complexes	Fonctionnement stimulus/réponse
Petits nombre d'agents	Grand nombre d'agents

Tableau 4 : les agents cognitifs et réactifs [39]

Toutefois, il est possible de concevoir des systèmes hétérogènes comportant les deux types d'agents. Il est aussi possible de doter les agents cognitifs de capacités de réactions aux événements : on parlera alors " d'agents hybrides ".

3.4 Architectures d'agents [40]

Différents types d'architecture d'agents ont été définis. Les principales architectures développées sont basées sur les comportements réactifs, sur la logique ou sur les croyances et intentions (BDI) des agents.

3.4.1 Architectures réactifs

L'idée principale des architectures basées sur des comportements réactifs est que le comportement intelligent de l'agent émerge des interactions entre ses comportements plus simples. Ce type d'architecture est fondé sur le concept de comportement. Chaque comportement est considéré comme la réalisation d'une action basique. Un comportement est régi par un cycle de Perception/Action dans lequel l'agent effectue directement une action en fonction de sa perception, sans réaliser de réflexion. Comme plusieurs comportements peuvent être exécutables en même temps, l'agent doit disposer d'un moyen de choisir parmi les comportements entrant en conflit. Pour cela, l'architecture à subsomption, définie par Brooks propose d'organiser les comportements de manière hiérarchique dans laquelle les comportements de niveaux inférieurs peuvent inhiber les comportements de niveaux supérieurs (figure 36)

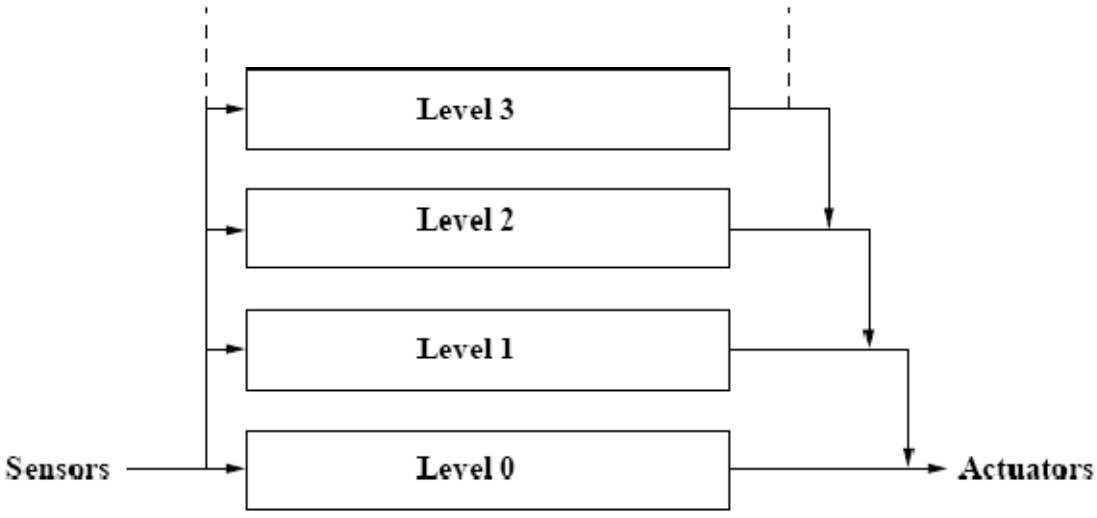


Figure 36 : Architecture à subsomption.

3.4.2. Les architectures logique

Dans les architectures fondées sur la logique, l'agent doit disposer de représentations symboliques de l'environnement. L'agent dispose donc un ensemble d'informations qui sont exprimées sous forme de prédicats de la logique du premier ordre (base de faits). Cette dernière est

vue comme la base des croyances chez l'humain. L'agent est également doté de règles d'inférence et la réalisation d'une action revient à appliquer une règle. La réalisation d'une action modifie l'environnement et donc la base de faits de l'agent.

Un problème survient quand plusieurs règles (ou aucune) peuvent être tirées en même temps, l'agent doit-il faire un choix, les exécuter toutes en parallèle ou les fusionner ? La manière dont l'agent résout ce problème peut être considéré comme son comportement.

3.4.3. Les architectures de type BDI

Les architectures de type BDI (*Believe, Desire, Intention*) sont basées sur un PRS (*Procedural Reasoning System*) où l'idée principale est de construire un plan d'actions pour atteindre le but fixé. Un tel système repose sur quatre modules :

- les croyances de l'agent (*Believe*) : ce sont les connaissances de l'agent sur l'état du monde (obtenues via ses capteurs et les communications),
- les désirs (ou buts) de l'agent (*Desire*) : c'est l'état du monde que l'agent souhaite atteindre,
- l'ensemble des plans : c'est la connaissance procédurale de l'agent,
- les intentions de l'agent (*Intention)* : c'est la liste des plans que l'agent a sélectionné en fonction de ses croyances et de ses désirs.

3.4.4. Les architectures multi-niveaux

Partant du principe qu'un agent est capable de comportements réactifs et de comportements pro-actifs, les architectures multi-niveaux décomposent le comportement de l'agent en couches de complexité croissante. Un tel système est composé d'au moins deux couches : une couche réactive et une couche pro-active. Parmi ces architectures, on peut distinguer trois types de système :

- *Système en couche horizontale* : chaque couche perçoit l'environnement et propose l'exécution d'une action. Il faut alors définir la manière dont l'agent choisit la couche prioritaire dans le cas de conflits. Un exemple d'une telle architecture(Touring Machine) proposée par Ferguson est donnée figure 37.
- *Système en couche verticale à une passe* : les informations perçues passent de couche en couche en commençant par celle de niveau inférieur. Chaque couche propose l'exécution d'une action à la couche de niveau supérieur et c'est la dernière couche qui désigne l'action à exécuter.

- *Système en couche verticale à deux passes* : dans cette architecture, la première passe est d'abord exécutée, le choix de l'action redescend de couche en couche et c'est la couche inférieure qui effectue l'action.

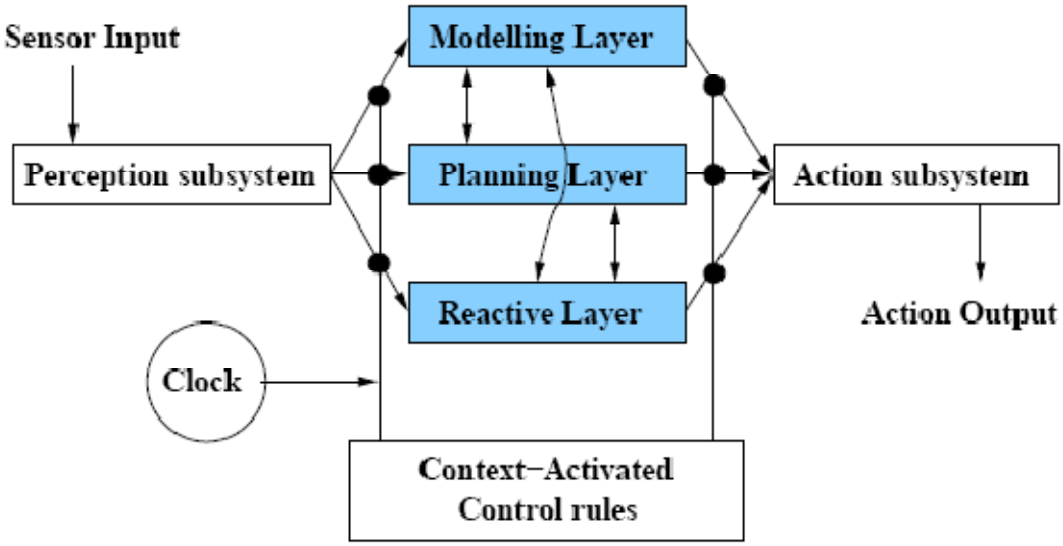


Figure 37 : Touring Machine.

3.5. Les propriétés des environnements

Les mondes dans lesquels les agents évoluent sont appelés des environnements. Les environnements peuvent être de nature très différente allant du plus simple au plus complexe. Il peut s'agir d'espaces concrets (comme le terrain sur lequel se déplace un véhicule) ou abstraits (comme les réseaux informatiques ou les espaces d'information).

Introduire les concepts de situation et de localité permet bien souvent de simplifier la complexité spatiale. On va lister les propriétés afin de caractériser les environnements : [33]

- **Accessible**: Si les organes sensitifs d'un agent lui donnent accès à l'état complet de l'environnement, alors l'environnement est dit accessible à cet agent. Dans ce cas, les capteurs détectent tous les aspects qui interviennent dans le choix d'une action. L'agent n'a pas à maintenir d'état interne pour garder une trace du monde.
- **Déterministe**: Dans un environnement déterministe, l'état du monde à l'instant suivant est complètement déterminé par l'état courant et par les actions effectuées par les agents. Episodique Dans un environnement épisodique, la vie de l'agent est divisée en épisodes. Ce qui se passe dans l'épisode ne dépend pas de ce qui s'est passé dans les épisodes précédents.
- **Dynamique**: Si l'environnement évolue pendant que l'agent décide quelle action accomplir, alors on dit que l'environnement est dynamique pour cet agent, sinon il est statique. Les

environnements statiques évitent donc à l'agent de regarder comment est le monde pour sélectionner une action. L'agent n'a donc pas à se soucier du passage du temps.

- **Discret**: Un environnement est dit discret s'il y a un nombre limité de perceptions et d'actions distinctes possibles. Dans un jeu d'échecs, l'environnement est discret car seul un nombre fixé de mouvements est possible à chaque tour. A l'inverse, la conduite d'un véhicule est continue car divers éléments comme la vitesse appartiennent à un intervalle de valeurs continues.

4. LES SYSTEMES MULTI-AGENTS

4.1. Définition

Un système multi-agent est un système distribué composé d'un ensemble d'agents. Contrairement aux systèmes d'IA, qui simulent dans une certaine mesure les capacités du raisonnement humain, les SMA sont conçus et implantés idéalement comme un ensemble d'agents interagissants, le plus souvent, selon des modes de coopération, de concurrence ou de coexistence.[41]

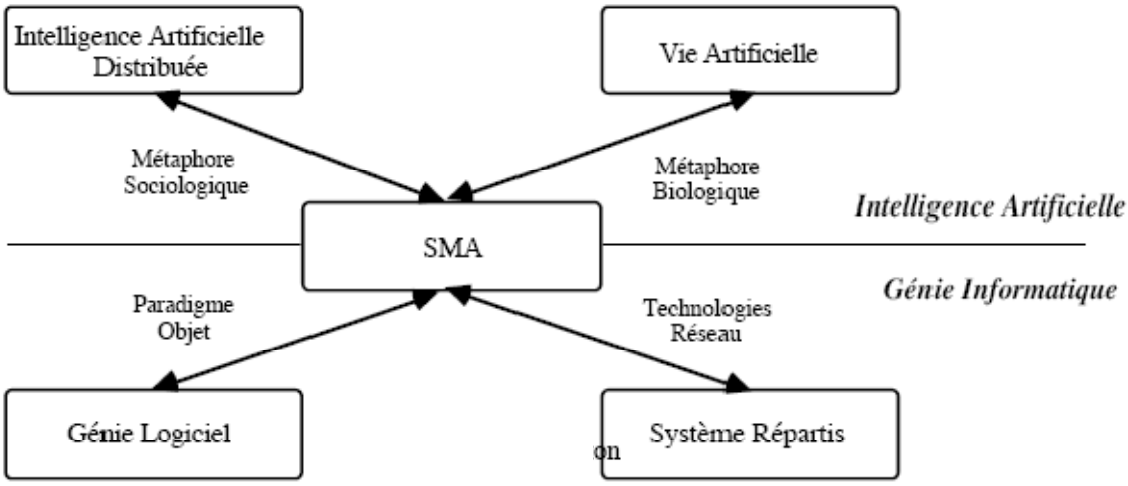


Figure 38 : SMA et les autres branches connexes

Ferber [35], définit un système multi-agents de la façon suivante :

On appelle système multi-agents (ou SMA), un système composé des éléments suivants (Figure 38):

1. Un environnement **E**, c'est-à-dire un espace disposant généralement d'une métrique.
2. Un ensemble d'objets **O** situés dans **E.** Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
3. Un ensemble **A** d'agents, qui représentent les entités actives du système.

4. Un ensemble de relations **R** qui unissent des objets (et donc des agents) entre eux.
5. Un ensemble d'opérations **Op** permettant aux agents de **A** de percevoir, produire, consommer, transformer et manipuler des objets de **O**.
6. Des opérateurs chargés de représenter l'application de ces opérations et la réaction de l'environnement envers les tentatives de modification.

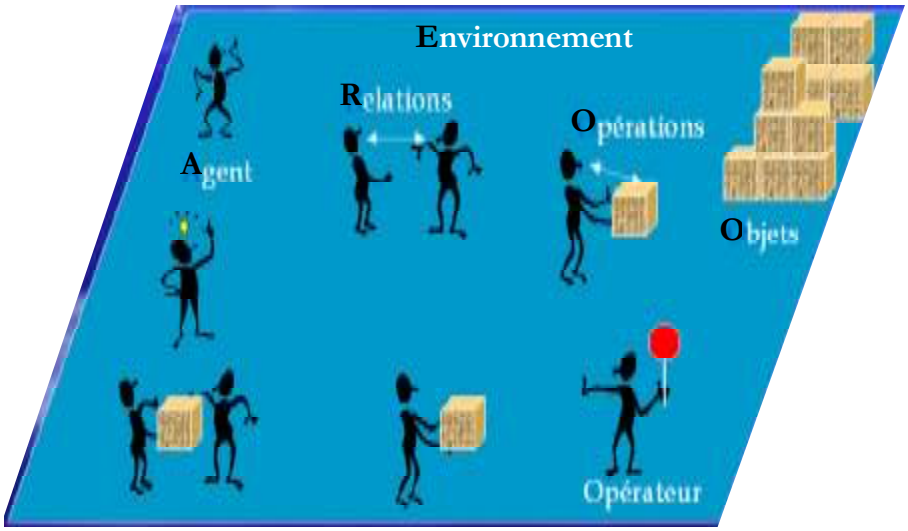


Figure 39 : Exemple d'un Système Multi-agents

4.2. Caractéristiques d'un SMA

Généralement, un SMA possède les caractéristiques suivantes :

- Les agents agissent et travaillent indépendamment les uns des autres.
- Chaque agent est une partie du système.
- Chaque agent travaille dans le but d'accomplir ses tâches.
- Chaque agent est capable de communiquer et d'interagir avec d'autres agents.
- Un agent coopère avec les autres agents lorsque c'est nécessaire.
- Un agent est capable de coordonner ses activités avec les autres agents pour accéder à des ressources et à des services partagés dont il a besoin (pour réaliser ses buts).
- Les agents ont un but commun.
- Chaque agent a une vue partielle du SMA.

4.3. Types d'architectures des SMA

Les SMA peuvent être conçus selon plusieurs types d'architectures [32]

4.3.1. Structure horizontale

Dans une architecture horizontale, tous les agents sont au même niveau, il n'y a pas d'agents maîtres et d'agents esclaves. C'est le cas, par exemple, d'un groupe d'agents ayant des spécialités différentes travaillant pour la résolution d'un même problème.

4.3.2. Structure verticale

Dans une architecture verticale, les agents sont structurés par niveaux. Dans un même niveau on retrouve localement une structure horizontale. Le fonctionnement des agents dans de telles sociétés est le suivant : l'agent reçoit le problème à résoudre par un autre agent qui lui est supérieur dans la hiérarchie, il le décompose en des sous problèmes auxquels il peut répondre localement, des sous problèmes qu'il pourrait résoudre en coopérant avec les autres agents du même niveau que lui, des sous problèmes qu'il fait suivre aux agents du niveau inférieur dans la hiérarchie.

4.3.3. Les systèmes *Blackboard* (tableau noir)

Le modèle de *Blackboard* définit une architecture qui organise la résolution de problèmes par coopération de plusieurs modules, appelés sources de connaissances KS(*Knowldge Source*), autour d'une base de données partagée appelée *Blackboard* (tableau noir). Chaque KS vient lire ou écrire sur le *Blackboard*. D'une façon générale, le rôle d'une KS est de résoudre un sous problème particulier en fonction de l'état du *Blackboard*.

5. L'INTERACTION DANS LES SMA

5.1. Définition

La notion d'interaction est au centre de la problématique des SMAs.

Selon Ferber [35] : une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensembles d'actions.

Principalement, un agent interagit avec les membres de son environnement pour quatre raisons principales :

- Partager de l'information ;
- Atteindre ses buts ;
- Eviter, autant que possible, les conflits ;

5.2. Types d'interactions

Il existe différents types d'interactions que les agents peuvent utiliser comme la coordination, la coopération et la communication [35].

5.2.1 Coordination

La coordination est le processus de construction de programmes en assemblant les parties actives.

Ainsi, la coordination détermine en quelque sorte quelles sont les règles de bon fonctionnement du système auquel les agents appartiennent. Selon la nature des agents et des interactions, plusieurs formes de coordination sont possibles. La coordination peut être imposée à un agent par une sorte de contrat, comme le respect d'un protocole d'interaction donné. Elle peut aussi être apprise, lorsque l'agent trouve un intérêt à participer à la collectivité.

5.2.2. Coopération

La coopération est une caractéristique très importante dans les systèmes multi agents. En effet, une résolution distribuée d'un problème est le résultat de l'interaction coopérative entre les différents agents. Dans le cadre d'une telle dynamique collective, un agent doit disposer en plus de la connaissance reflétant son degré d'implication dans cette dynamique (croyances, buts) d'un certain nombre de compétences nécessaires pour la coopération. Il doit pouvoir :

- mettre à jour le modèle du monde environnant,
- intégrer des informations venant d'autres agents,
- interrompre un plan pour aider d'autres agents.

5.2.3. Communication

Le système de communication qui lie un ensemble d'agents agit comme un système nerveux qui met en contact des individus parfois séparés. La communication en effet agrandit les capacités perceptives des agents en leur permettant de bénéficier des informations et du savoir-faire des autres agents. Les communications sont indispensables à la coopération et il est difficile de concevoir un système d'agents coopérants s'il n'existe pas un système permettant aux agents d'échanger des informations ou de véhiculer des requêtes. Il existe deux principaux modes de communication : la communication par envoi de messages et la communication par partage d'informations [35].

A. Communication par envoi de messages

Les agents sont en liaison directe et envoient leurs messages directement et explicitement au destinataire. La seule contrainte est la connaissance de l'agent destinataire: "Si un agent A connaît l'agent B alors il peut entrer en communication avec lui".

Les systèmes fondés sur la communication par envoi de messages relèvent d'une distribution totale à la fois de la connaissance, des résultats et des méthodes utilisées pour la résolution du problème.

B. Communication par partage d'information

Les agents ne sont pas en liaison directe mais communiquent via une structure de données partagée, où on trouve les connaissances relatives à la résolution (état courant du problème) qui évolue durant le processus d'exécution.

5.3Les langages de communication agent (ACLs) [42]

Dans les SMAs cognitifs, l'hypothèse la plus répandue est que la communication inter-agent sera plus fructueuse si elle se fait par l'intermédiaire d'un langage de communication explicite. La propriété essentielle qui rend le langage utile, c'est que le sens de ses signes soit partagé. Ceci est vrai pour les langues vivantes mais aussi pour tous les signes codés comme les coups de sifflet d'un arbitre de football, on utilise alors au minimum :

- un dictionnaire de vocabulaire/signes commun(s) : l'ontologie des services (dont le sens est au moins partiellement commun aux agents) qui permet une interprétation commune des contenus propositionnels des actes de langage.
- des actes de langage utilisant cette ontologie qui servent de briques de bases de la communication et correspondent aux attitudes propositionnelles transmises entre les agents.

Les langages de communication agent, notés ACL[Agent Communication Language] dans la suite , prennent donc place dans une couche logiquement supérieure à celle des protocoles de transfert (TCP/IP, HTTP, IIOP) et adressent le niveau intentionnel et social des agents. Ils se différencient donc des mécanismes de la théorie de l'information non seulement par leur structure et leur syntaxe plus complexe, mais aussi par leurs spécifications génériques et précises. Leur puissance expressive, héritée de la théorie des actes de langage permet d'envisager développer des SMAs théoriquement hétérogènes et ouverts. C'est-à-dire dont les agents (cognitifs), hétérogènes en architectures internes et issues de développeurs différents, peuvent se communiquer des messages mutuellement compréhensibles.[42]

Les deux ACLs les plus connus sont KQML et FIPA-ACL. Voyons en quoi ils consistent.

5.3.1 KQML [Knowledge Query and Manipulation Language]

Premier apparu, KQML fourni un ensemble d'actes de langage standards et utiles. Le langage est structuré selon trois niveaux enchâssés:

1. la couche de communication : renseigne la communication (identité du récepteur, de l'émetteur et nature de la communication). Elle est minimale car KQML ne prend pas en charge le transport lui-même (TCP/IP, SMTP, IIOP ou autres).
2. la couche message : donne des indications sur le contenu du message (langage et ontologie utilisé pour le contenu, type d'acte de langage attaché au contenu). C'est la couche centrale de KQML qui définit le type d'interaction que des agents-KQML pourront avoir.
3. la couche de contenu : contenu du message exprimé en KIF [Knowledge Interchange Format], Prolog, KQML ou autre. Notons que KQML ne traite pas cette couche si ce n'est pour savoir où le contenu commence et se termine. Comme le contenu du message est opaque, c'est à la couche message de le renseigner.

Les performatifs KQML sont divisés en trois catégories :

1. 7 performatifs de régulation de conversation traitent quelques cas particuliers (sorry, error) et permettent quelques variantes de la conversation (standby, ready, next, rest, discard) ;
2. 17 performatifs de discours permettent l'échange d'informations et de connaissances (ask-if, tell, deny, stream-all,...) ;
3. 11 performatifs d'assistance et de réseau pour étendre la conversation à plus de deux agents (forward, broker-all,...).

KQML est issu d'un projet de la DARPA, le KSE [Knowledge Sharing Initiative], initialement prévu comme moyen d'échange d'informations entre programmes à base de connaissances. Cependant, sa structure orientée message et la généricité de ses primitives lui permettent d'être utilisé comme ACL. C'est l'ACL le plus utilisé (et implémenté) dans la communauté SMA. Le développement de KQML n'est pas centralisé et ainsi plusieurs variantes incompatibles sont nées.

5.3.2 FIPA-ACL

C'est le seul effort réellement organisé pour créer un ACL standard. Comme FIPA-ACL est défini par une corporation de chercheurs, sa mise à jour est lente mais chaque version est scrupuleusement vérifiée. Il a été conçu pour palier aux faiblesses des différentes versions de KQML. FIPA-ACL diffère de KQML car il a été doté d'une sémantique. En effet, si KQML décrit

la syntaxe de ses messages, rien n'est dit sur leur sens précis (indépendamment qu'ils correspondent grossièrement à différents types d'actes de langage), c'est-à-dire de leur sémantique.

Celle de FIPA-ACL s'exprime en termes de conditions de faisabilité et d'effets rationnels. Un acte est sélectionné lorsque les conditions de faisabilité sont remplies et que l'effet rationnel correspond à une intention du locuteur.

FIPA-ACL définit des pré-conditions de faisabilité des messages et des post-conditions décrivant les effets rationnels attendus. Le langage sémantique de FIPA-ACL (SL) est une logique multimodale avec des opérateurs pour les croyances (B), les désirs (D), les croyances incertaines (U), les intentions (but persistant PG).

6. METHODOLOGIES DE CONCEPTION DES SMA :

Il existe plusieurs techniques ou méthodes de conception des systèmes multi-agents qui s'inspirent, généralement, de méthodologies de modélisation existantes utilisées dans le domaine de l'ingénierie des connaissances ou Orientées agents, en leurs rajoutant et/ou supprimant quelques modules nécessaires pour recouvrer les différents aspects des systèmes multi-agents. Comme il existe des méthodes directement centrées sur les agents comme l'illustre la figure suivante :

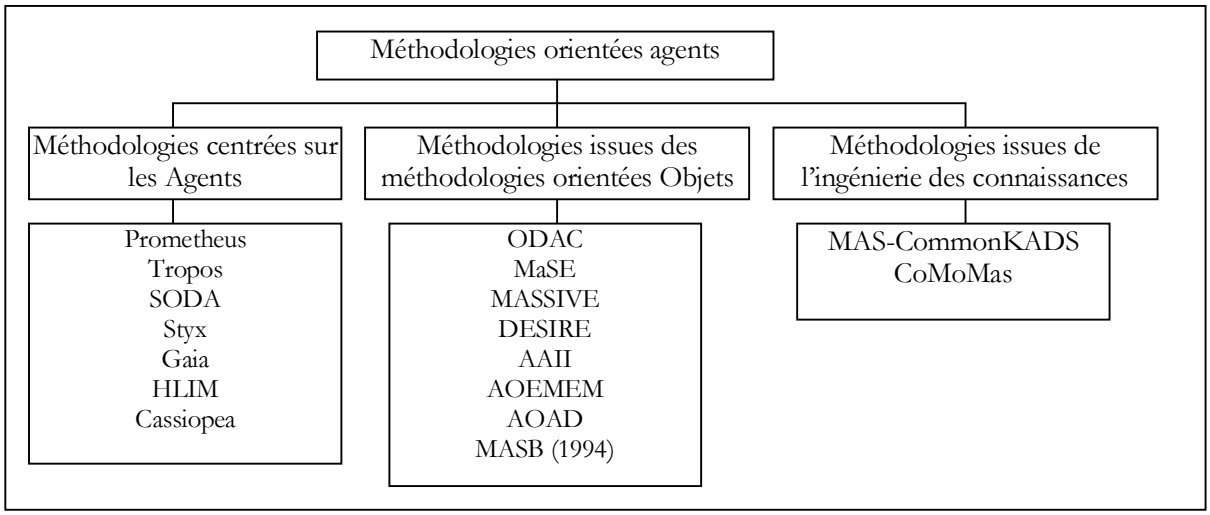


Figure 40 : Les méthodologies orientées agents [43]

6.1. Méthodologies centrées sur les Agents :

Ces méthodes sont caractérisées par la prise en compte des concepts du paradigme agent dès la phase d'analyse du système contrairement aux autres méthodes. Parmi ces méthodes on peut citer **[44]** :

6.1.1. La méthode GAIA :

Elle contient cinq modèles, et elle se décompose en deux phases principales :

- **La phase d'analyse :** elle produit les modèles suivants :
  - **Le Modèle de rôle :** identifie les principaux rôles du système;
  - **Le Modèle d'interaction :** qui est un ensemble de définitions de protocoles d'interaction.
- **La phase de conception :** Le processus de conception se base sur les trois modèles suivants :
  - **Le Modèle d'agent :** identifie les agents du système en les associant aux différents rôles;
  - **Le Modèle de service :** identifie les principaux services nécessaires pour accomplir les rôles des agents. Un service étant un bloc cohérent d'activités dans lequel un agent s'engagera.
  - **Le Modèle d'accointance** : définit les liens de communications entre les différents agents;

6.1.2. La méthode HLIM (High-Level and Intermediate Models) :

La méthodologie HLIM fait ressortir, à travers deux phases (découverte et définition), les aspects des agents tels que : leurs objectifs, leurs plans, leurs croyances et les rapports entre ces agents.

La phase de découverte, qui est une phase exploratoire du domaine, est la première phase de HLIM qui produit le modèle High-Level Model. Ce modèle capture la structure et le comportement du système. La deuxième phase de définition produit les quatre modèles suivants :

- Internal Agent Model;
- Relationship Model;
- Conversational Model;
- Contract Model.

6.2. Méthodologies issues des méthodologies orientées Objets :

Parmi ces méthodes on trouve AOEM et MaSE qui étendent des méthodologies orientées objets pour concevoir des SMA **[43].**

6.2.1. La méthode AOEM (Agent Oriented methodology for Enterprise Modelling) :

Elle combine des méthodologies orientées objets à l'approche de modélisation fonctionnelle IDEF (Integration DEfinition for Function modelling). Le processus de modélisation est constitué de quatre étapes successives qui sont :

- Identifier et décrire les fonctions du système par la conception de diagrammes IDEF0 (décomposition hiérarchique et fonctionnelle), ainsi elle identifie les agents du système ;

- Transformer les diagrammes IDEF0 en utilisant la notation des cas d'utilisation ;
- Décrire les interactions entre les cas d'utilisation et entre les acteurs et les cas d'utilisation ;
- Conception du système orienté agents. Les agents représentent les acteurs des cas d'utilisation et sont définis selon des architectures BDI (Belief-Desire-Intention).

6.2.2. La méthode MaSE (Multiagent Systems Engineering):

Elle est composée de deux phases principales : analyse (Identifier les buts, Appliquer les cas d'utilisation et Perfectionner les rôles) et conception (Créer les classes d'agent, Construire les conversations, Assembler les classes d'agent et Concevoir le système). Les étapes associées à cette méthodologie sont les suivantes :

- Identifier les buts du système afin de les structurer et de les représenter sous forme d'une hiérarchie.
- Identifier les rôles et leurs interactions en utilisant les Cas d'utilisation pour définir le comportement général du système, ses fonctionnalités, son environnement (utilisateurs et acteurs) et les rôles du système et les diagrammes de séquence pour représenter les messages échangés entre les rôles.
- Identifier la décomposition fonctionnelle du système en utilisant les Rôles et les Tâches Concurrentes. Chaque rôle correspond au moins à un but, auquel s'ajoute un ensemble de tâches correspondantes. Lors de la création des modèles de rôles, les interactions entre les rôles sont définies en connectant les tâches entre elles. Le modèle de Tâches Concurrentes permet de représenter, à l'aide d'automates à états finis, les tâches réalisées par les rôles pour l'accomplissement de leurs buts respectifs.
- Identifier le système multi-agents par la description des architectures d'agents et de leurs liens conversationnels. Une classe d'agent précise les rôles qui lui sont assignés.
- Définir les protocoles de coordination entre deux agents en expliquant les comportements des agents durant la conversation.
- Spécifier l'architecture interne des agents.
- Concevoir le système en spécifiant la distribution des agents selon l'architecture physique du système.

7. QUELQUES EXEMPLES D'APPLICATION DES SMA

De nos jours, la technologie multi-agents a trouvé sa place dans les systèmes manufacturiers, les systèmes financiers, les loisirs, les télécommunications, le contrôle commande, les systèmes embarqués et d'autres applications [32].

7.1. Application des SMAs aux Systèmes d'Informations Coopératifs (SIC)

Les SIC sont généralement caractérisés par la grande variété et le grand nombre de sources d'informations. Ces sources d'informations sont hétérogènes et distribuées soit sur un réseau local (*Intranet*) soit sur l'Internet. De tels systèmes doivent être capables d'exécuter principalement les tâches suivantes :

- la découverte des sources : trouver la bonne source de données pour l'interroger;
- la recherche d'informations : identifier les informations non structurées et semi structurées;
- le filtrage des informations : analyser les données et éliminer celles qui sont inutiles;
- la fusion des informations : regrouper les informations d'une manière significative.

Plusieurs systèmes touchent à ce genre d'application. Parmi ces applications, nous pouvons citer :

- InfoSleuth : C'est un système multi-agents pour la recherche coopérative d'informations dans des bases de données distribuées. Ce système a été appliqué aux domaines médicaux
- NetSA (pour "Networked Software Agents") : C'est un système proche de Infosleuth et dédié aux environnements riches en informations.

7.2. Application des SMAs aux systèmes médicaux

Le système GUARDIAN [32] a pour but de gérer les soins aux patients d'une unité de soins intensifs chirurgicale. Les principales motivations de ce système sont :

- Le modèle des soins d'un patient dans une unité de soins intensifs est essentiellement celui d'une équipe, où un ensemble d'experts dans des domaines distincts coopèrent pour organiser les soins des patients;
- Le facteur le plus important pour donner de bons soins aux patients est le partage d'informations entre les membres de l'équipe de soins critiques. En particulier, les médecins spécialistes n'ont pas l'opportunité de superviser l'état d'un patient minute par minute; cette tâche revient aux infirmières qui, quant à elles, ne possèdent pas les connaissances nécessaires à l'interprétation des données qu'elles rassemblent.

Le système GUARDIAN répartit donc le suivi des patients à un certain nombre d'agents de trois types différents. :

- Les agents perception/action qui sont responsables de l'interface entre GUARDIAN et le monde environnant.
- Les agents en charge du raisonnement ayant pour rôle d'organiser le processus de prise de décision du système.
- Les agents en charge du contrôle assurent le contrôle de haut niveau du système.

8. CONCLUSION

Dans ce chapitre, nous avons présenté une vue détaillée sur une approche qui est actuellement un champ de recherche très actif. Cette discipline s'intéresse tout d'abord aux notions d'agents et de systèmes multi-agents (SMAs), et détaille par la suite les différentes questions relevées par la problématique des SMAs, en particulier : les interactions entre agents c-à-d, la coopération, la coordination et la communication.

L'étude effectuée, a mis en évidence que les concepteurs de SMA font de nos jours face à deux difficultés majeures. Tout d'abord, l'absence de méthodologie systématique qui permettrait de spécifier et de structurer une application multi-agents. Ensuite, le manque d'outils commerciaux pour bâtir des SMA.

CHAPITRE IV 4

Conception

CHAPITRE VI : CONCEPTION

1. LA PROBLEMATIQUE

Le but de notre travail est de proposer une architecture qui permet à un décideur d'avoir accès à toutes les informations qui lui sont nécessaires pour sa prise de décision. Le profil d'un décideur est variable, dans le sens où celui-ci peut être un expert du domaine, un expert en informatique (par exemple un administrateur) ou bien un utilisateur non expert du domaine ni en informatique. Les informations recherchées sont stockées dans le Data warehouse qui est le résultat de l'intégration des différentes bases de données hétérogènes et réparties de l'association. Lors de l'interrogation de ces bases, plusieurs problèmes sont à prendre en compte:

- le problème du volume de données: les données sont très nombreuses, et le volume des données à considérer dans le domaine des entrepôts de données ne cesse de croître;
- le problème des données hétérogènes: les données de types différents sont stockées dans des bases de données distinctes, parfois sur des sites physiques différents et distants. Le problème des bases de données réparties se pose alors. De plus, il faut être capable de communiquer rapidement et de façon sûre avec chaque base de données afin de récupérer en réponse seulement les données pertinentes ;
- le problème de la représentation des données: l'utilisateur, humains ou logiciels, doit connaître une partie de la représentation des données pour formuler sa requête. Permettre à un utilisateur de formuler sa requête sans qu'il sache comment sont représentées les données donne une liberté plus grande à l'utilisateur et permet d'être moins dépendant des bases de données.

Il est donc nécessaire de permettre l'interrogation du Data warehouse par l'utilisateur, et des bases de données par le système de data warehousing sans connaître la représentation des données. Celle-ci n'est connue que par le système, qui peut ensuite faire traduire la requête en des sous requêtes destinées au data warehouse et dans d'autres cas de base de données en particulier, et rédigée en un langage compréhensible par le SGBD qui le gère. Dans notre étude, nous nous intéressons en particulier au problème d'interrogation et l'intégration de ces bases de données dans le data warehouse en ajoutant une fonction de recherche pour l'ensemble des informations liées par le sujet des requêtes d'interrogation de l'utilisateur dans les sources interne de l'association ou dans l'Internet.

L'objectif de notre travail est de pouvoir impliquer les agents dans la conception et l'implémentation d'une architecture de data warehousing capable d'intégrer les informations des bases de données réparties à grande échelle et hétérogène dans le contexte de data warehousing, et de bénéficier au maximum des avantages offert par la technologie agent.

2. L'APPORT DE L'UTILISATION DE LA TECHNOLOGIE AGENT

Les modèles distribués basés sur le client/serveur ont été largement étudiés dans la littérature, toutefois il s'est avéré cela dans des applications réelles qu'il a manqué d'une performance satisfaisante. C'était dû à l'augmentation du trafic de réseau pendant le processus d'intégration, de fusion et de diffusion. D'ailleurs, la concentration du processus de transformation et de fusion des données dans un ordinateur central abaisse la performance de ce modèle. Le besoin d'améliorer la performance nous a amené à étudier l'utilisation des agents logiciels pour la mise en place d'un tel système.

Notre architecture est basée sur le paradigme agent. Le choix des agents comme technologie de solution pour notre architecture a été motivé par ce qui suit:

- **Modularité:** nouveaux composants peuvent être ajoutés ou retirés sans effectuer l'exécution d'autres élément du système.
- **Extensibilité:** Pour permettre aux nouveaux éléments d'être facilement ajoutés au système.
- **Autonomie:** L'intégrité de la structure d'organisation existante et l'autonomie de ses parties doivent être maintenue.
- **Flexibilité** La capacité de traiter la nature dynamique du système.
- **Formes déclaratives de transmission**: la communication entre les composants dans notre système est due notamment en utilisant l'information et la connaissance. Les interactions sont assez sophistiquées, y compris le partage des informations, et la coordination. Ainsi, les protocoles de communication basés sur des langages de communication d'agent connus, sont plus appropriés pour notre but.

3. LES ARCHITECTURES DE REFERENCE

3.1. Architecture d'entreposage de données de référence WHIPS [47][48][49]

Le système WHIPS (WareHouse Information Prototype at Stanford) est un système de gestion d'entrepôts de données utilisé comme banc d'essai. Le but de ce projet est de développer des algorithmes pour collecter, intégrer et maintenir des informations émanant de sources hétérogènes,

distribuées et autonomes, et de les transformer et les résumer selon les spécifications du data warehouse, et les intégrer de façon incrémentale dans le data warehouse.

Rappelons que deux composants principaux constituent un data warehouse :

- un composant d'intégration, qui collecte et maintient les vues matérialisées, et
- un composant de requêtes et d'analyse, qui a pour but de compléter l'information demandée par l'utilisateur final.

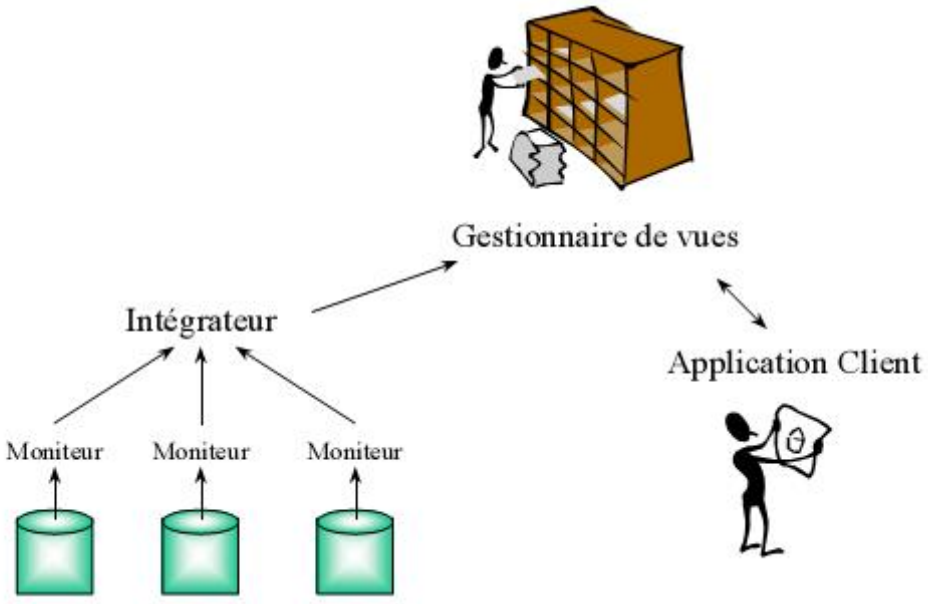


Figure 41 : schématisation du processus du système WHIPS

La modularité de l'architecture permet entre autres : d'ajouter et de supprimer des sources de données et des vues dynamiquement, de détecter automatiquement les changements dans les sources de données, d'avoir un data warehouse toujours cohérent avec les données des sources grâce à des algorithmes d'intégration. La Figure 40 illustre le principe de WHIPS, tandis que la Figure 41 présente le composant d'intégration d'un data warehouse, pour représenter l'architecture du système WHIPS.

Le système WHIPS est composé de nombreux modules distincts, qui communiquent les uns avec les autres bien qu'ils soient, potentiellement, sur des machines différentes. Les données du data warehouse sont représentées selon le modèle relationnel : les vues sont définies dans ce modèle et l'entrepôt stocke les relations. En ce qui concerne les divers modules, nous citons :

a. **moniteur** : le **moniteur** est responsable de la détection des modifications des données sources et de leur notification à l'intégrateur. Il existe un moniteur pour chaque source. Comme le wrapper tous les moniteurs supportent la même interface de méthodes bien que leur code interne dépende de leur source.

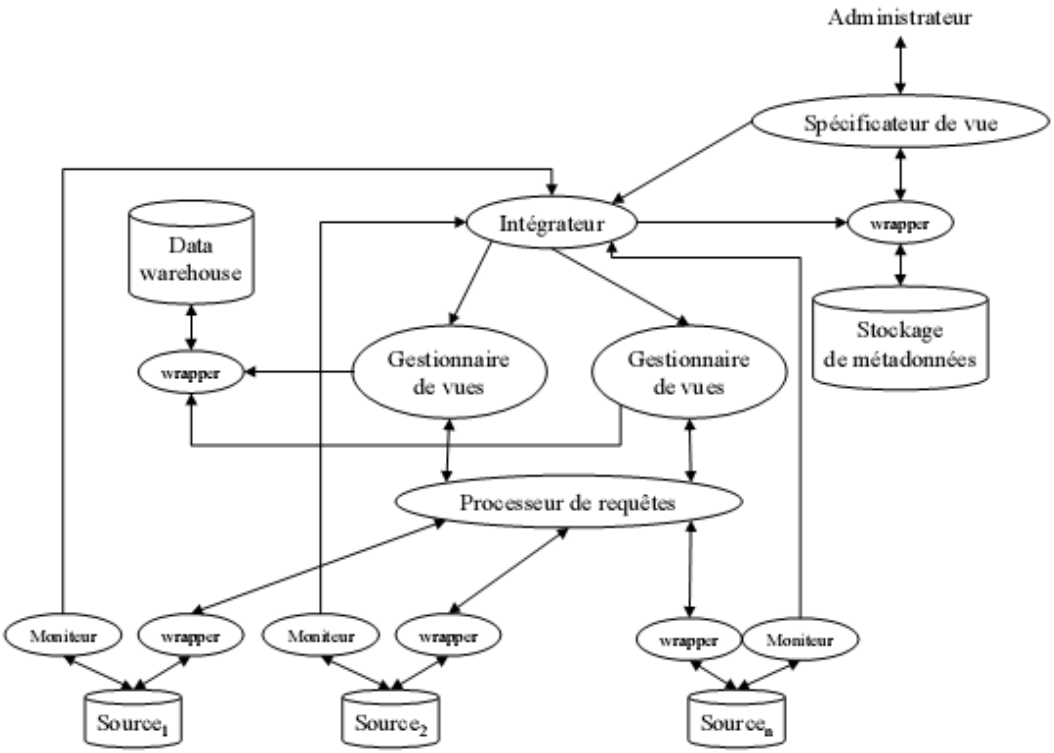


Figure 42 : Architecture du système WHIPS pour la maintenance de l'entrepôt de données

b. **wrappers**: les **wrappers**, quant à eux, traduisent des requêtes simples d'une représentation relationnelle interne en des requêtes dans le langage natif de la source. Par exemple, un wrapper de base de données relationnelle ne pourra traduire que des expressions relevant de l'algèbre relationnelle en SQL. L'utilisation d'un wrapper par source cache les détails d'interrogation (spécifique à la source) au processeur de requêtes et aux autres modules : tous les wrappers supportent la même interface de méthodes bien que leur code interne dépende de leur source.

c. **Le wrapper du data warehouse:** reçoit les définitions de vues et les modifications pour les données de vue dans un format (interne) canonique, et les traduit dans la syntaxe spécifique à la base de données de l'entrepôt. Le wrapper protège ainsi tous les autres modules dans le système WHIPS des détails de l'entrepôt. Toutes les modifications reçues par le wrapper de l'entrepôt dans un seul message sont appliquées à l'entrepôt en une seule transaction.

d. **spécifieur de vues**: Les vues sont définies par l'administrateur système au moyen d'un sous-ensemble de SQL. Lorsqu'une vue est définie, elle transmise au **module spécifieur de vues** qui la traduit, ajoute des informations pertinentes (comme les attributs clés, les types d'attributs, etc.),et qui la compile dans une structure interne, appelée arbre-vue qui comporte des informations issues d'un gestionnaire de méta données, puis l'envoie à l'intégrateur.

e. **l'intégrateur**: a pour principal rôle de faciliter la maintenance des vues en calculant quelles modifications de sources ont besoin d'être propagées dans les vues.

f. **Gestionnaire de vues:** c'est un module responsable du maintien chaque vue.

g. **Le processeur de requêtes**, quant à lui, reçoit les requêtes globales des gestionnaires de vues et pose les requêtes appropriées à chaque source aux wrappers afin qu'ils y répondent. Il passe ensuite les réponses aux gestionnaires de vues.

3.2. Architecture Multi-Agents de référence NetSA [45][46]

Parmi les logiciels capables d'opérer sur des sources d'information hétérogènes placées dans un environnement ouvert et dynamique (comme par exemple l'Internet), NetSA (**Net**worked **S**oftware **A**gent) est une architecture réutilisable multi agent conçue pour œuvrer particulièrement sur des sources hétérogènes placées dans un Environnement ouvert et dynamique.

Le projet NetSA a débuté en Janvier 1998 au laboratoire DAMAS (Data-mining Agents And Multi-agents) du département d'informatique de l'Université Laval, sous la direction du Pr. B. Chaib-draa.

3.2.1. Les trois couches de NetSA

NetSA est une architecture à trois niveaux d'abstraction. Chaque niveau constitue une unité abstraite qui occupe une tâche bien précise. Comme le montre la figure 3 nous retrouvons les unités suivantes:

a) Unité de communication avec l'utilisateur

Cette unité est chargée des communications entre NetSA et l'usager. Elle comprend des agents interagissant avec l'utilisateur pour l'aider à réaliser une tâche bien précise. Cette interaction se traduit par une transformation des requêtes de l'usager qui, transformées en des actes du langage KQML, facilitent la communication avec les agents de l'unité de traitement. L'unité vérifie également la consistance des données fournies par l'utilisateur.

b) Unité de traitement et de médiation

Cette unité reçoit de l'unité de communication les requêtes à satisfaire ainsi que les informations fournies par l'utilisateur. Elle décompose ces requêtes en sous-plans. Un sous-plan est une succession d'actions à exécuter dans le but d'atteindre un objectif intermédiaire. L'unité comporte également une partie ‹‹médiation›› pour rechercher des données dans le système multi-agents. Tel un patrouilleur, elle dirige les agents vers la ressource désirée en fournissant le nom de l'agent en charge de cette ressource.

c) Unité d'interrogation et d'extraction d'informations

Cette unité est composée d'agents formant une interface entre les bases de données et l'unité de traitement d'informations. Ces agents transforment les requêtes KQML reçues et les traduisent en requêtes SQL pour interroger des bases de données. De ces bases de données, les agents retirent l'information pertinente et la redirigent vers l'unité de traitement de l'information sous un format KQML. Ces agents peuvent également retirer l'information contenue dans une page HTML de l'Internet.

3.2.2. Les agents composant NetSA

Le système multi-agents NetSA comporte différents types d'agents, comme l'illustre la figure 3.

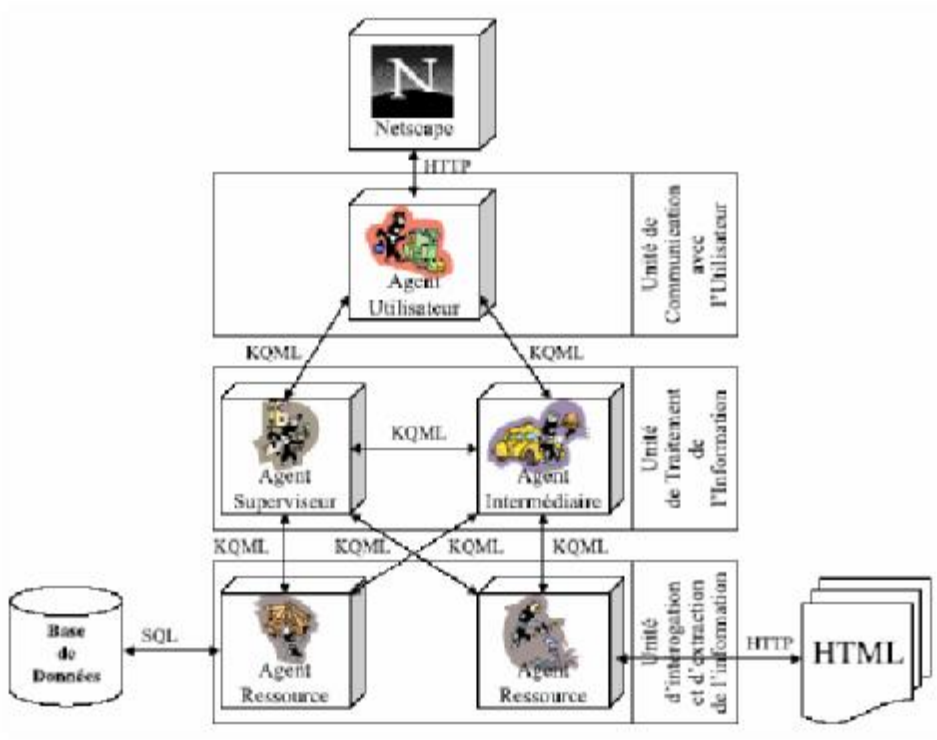


Figure 43 : Architecture NetSA

a) L'agent utilisateur

Il est la porte d'entrée des requêtes externes au système. Il fournit à l'utilisateur le bon formulaire HTML lui permettant de faire une requête. L'agent traduit ensuite le formulaire soumis dans le protocole KQML en vue de son utilisation dans le système multi-agents. L'agent utilisateur s'occupe de transmettre à l'utilisateur les pages Web requises pour la cueillette et l'affichage des informations pour le domaine auquel il a été destiné.

L'architecture interne de l'agent utilisateur, quant à elle, est composée de trois modules principaux et d'un registre de sauvegarde:

- **Le module de communication utilisateur** : C'est un Servlet qui communique avec le

module de traitement via des RMIs (Remote Method Invocation).

- **Le module de communication inter-agents:**
- **Le module de traitement:**
- **Le registre de sauvegarde:** Il a pour rôle de sauvegarder les données que l'utilisateur a fournies.

b) L'agent superviseur

Il est responsable de l'exécution à haut niveau des requêtes venant des agents utilisateurs. Lorsqu'il reçoit une requête, celle-ci est exécutée par le biais de plans prédéfinis. Majoritairement, un plan est composé de plusieurs sous-plans, et après l'exécution du plan, l'agent superviseur synthétise les réponses et transmet une réponse finale à l'agent appelant ayant fait la requête d'origine.

Comme tout agent de NetSA, l'agent superviseur a sa propre architecture interne. Celle-ci est divisée en trois modules accompagnés de deux bases de données :

- **Module de communication**: le module de communication est le lieu de transition de tous les messages entrant et sortant de l'agent superviseur.
- **Module de planification:** un module pour interpréter les plans.
- **Module d'évaluation d'expression:** permet l'exécution du contenu des plans.

c) L'agent intermédiaire

Il doit associer aux différentes requêtes les agents qui sont capables d'y répondre. Pour cela, les agents ressources doivent s'annoncer auprès de lui, en lui envoyant un message KQML l'avertissant du type de service qu'ils sont capables de fournir. Lorsqu'un agent utilisateur demande à l'agent intermédiaire s'il y a un agent capable de satisfaire sa requête, celui-ci lui répond en lui donnant le nom du ou des agents aptes à satisfaire ladite requête.

L'architecture de l'agent intermédiaire, comporte trois modules et deux bases de données. En ce qui concerne les bases de données, la première est une base de règles qui fournit à l'agent le protocole à suivre lors de la communication. La deuxième est une base de faits, dans laquelle l'agent emmagasine les connaissances qu'il a de son environnement. Quant aux trois modules, ils sont comme suit :

- **Le module de communication** : c'est une interface entre l'agent et son environnement. Il est utilisé pour transmettre et recevoir des messages sous la forme KQML.
- **Le module de réseaux à contrats** : il gère spécialement le protocole de réseau à contrats

dont les spécifications sont stockées dans la base de règles. Le réseau à contrats est un protocole de négociation.

- **Le module de traitement:** c'est un moteur d'inférence développé en Java et basé sur JESS.

d) L'agent ressource

Il reçoit des requêtes formulées en KQML et les transforme en requêtes SQL afin d'extraire l'information requise des bases de données ou de rechercher l'information demandée dans une page HTML. L'information trouvée est ensuite traduite en KQML pour répondre à la requête du demandeur.

L'architecture interne de l'agent ressource est, quant à elle, composée de deux modules et de deux interfaces:

- **Le module de communication :**
- **Le module de traitement** : Il traite les requêtes reçues par le module de communication. Pour cela, il les transforme en requête SQL et les achemine à l'interface avec la base de données où ordonne une recherche à l'interface HTTP. Il construit ensuite une réponse sous forme de message KQML qu'il donne au module de communication pour être expédiée.
- **L'interface avec la base de données** : L'interface exécute la requête SQL retourne les valeurs trouvées.
- **L'interface HTTP** : Elle reçoit du module de traitement une demande de recherche de variable dans une page Web. Elle parcourt alors la page Web à la recherche d'une étiquette ‹‹VAR›› définissant la variable demandée. Elle retourne ensuite la valeur de cette variable au module de traitement.

4. LE MODELE PROPOSE

4.1. Contribution

D'après l'étude effectuée sur les différentes étapes nécessaires pour élaborer un environnement à base d'agents capable de prendre en charge notre architecture; et en basant sur l'architecture de l'entreposage de données proposée par le projet WHIPS [47][48][49], on la fusionne avec l'architecture du système multi agent NetSA [45][46] précédemment présentée qu'est destinée aux environnements riche en informations hétérogènes comme celle de notre.

La décomposition de notre architecture peut être faite suivant trois couches essentielles:

- couche de communication avec l'utilisateur

- couche de traitement intermédiaire, et stockage du Data warehouse.
- couche d'interrogation et extraction d'informations

L'aspect modulaire de l'architecture WHIPS peut être directement projeté sur une approche orientée agent, nous avons choisi l'architecture NetSA comme une architecture référentielle. Pour cela nous allons attribuer un agent NetSA à chaque module ayant une caractéristique fonctionnelle correspondante à la capacité de cet agent. Dans le cas où il n'y a pas de telle correspondance nous allons définir un agent adéquat pour accomplir cette tache. Donc en va citer les agents comme suit:

Les Agents NetSA sont :

- agent ressource: wrapper,
- agent intermédiaire: intégrateur et Gestionnaire de vues
- agent superviseur : Le processeur de requêtes
- Agent Utilisateur

Les agents complémentaires à NetSA pour garder les fonctionnalités de base par rapport à WHIPS et le DW en général :

- agent moniteur: moniteur.
- Agent d'Administration: spécifieur de vues.

Et en plus de ça, nous rajoutons d'autres agents correspondants à la recherche et l'extraction des données provenant de l'environnement extérieur de l'association, tels que les sites des partenaires ou bien autres ressources d'information à partir d'Internet; et une interface avec les applications client du data warehouse; ces agents sont:

- Agent Extraction d'Information
- Agent Recherche d'Information
- Agent d'Application
- Agent de Décision

4.2. La structure de notre architecture

Le schéma général de notre architecture est donné par la figure (Figure 44).

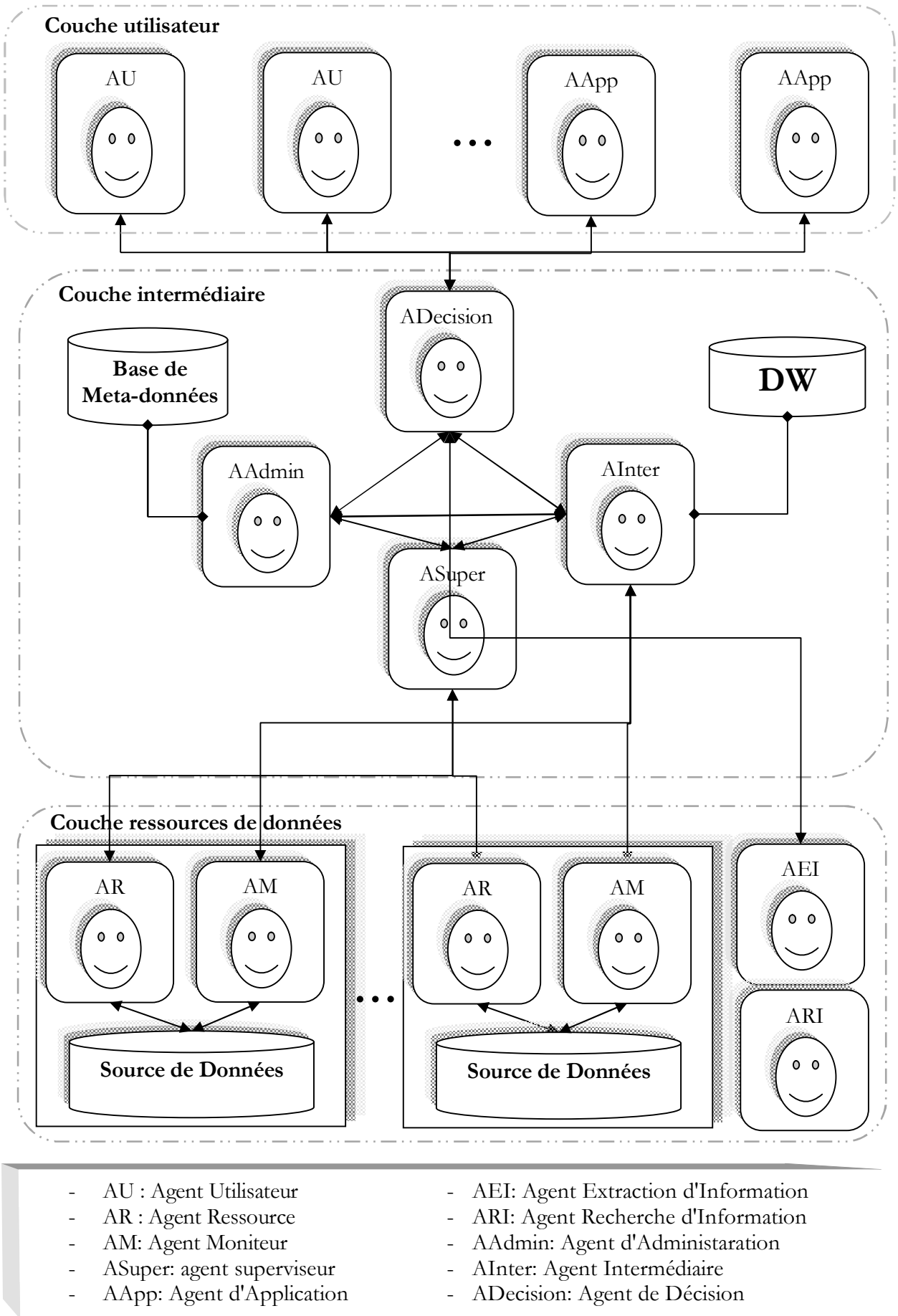


Figure 44 : Schéma générale de l'architecture proposée

4.3. Description de haut niveau de l'architecture

4.3.1. Les trois couches de l'architecture

Nous distinguons trois couches dans l'architecture proposée qui sont:

a) Couche utilisateur

Cette couche est chargée des communications avec l'usager et l'environnement tel que des utilisateurs humains ou bien d'autres applications (Datamining, outils OLAP, SIG…). Elle comprend :

- des agents utilisateurs interagissant avec l'utilisateur pour l'aider à réaliser une tâche bien précise (recherche d'information) en consultant son profil et historique. Cette interaction se traduit par le résultat (les valeurs de la requête).
- Des agents d'application qui sont responsables de la présentation d'interaction et des données. Il soumit la requête de l'application au tiers intermédiaire qui l'exécute et renvoi le résultat à leur destination.

b) Couche intermédiaire

C'est le cœur de notre architecture, dans cette couche la tâche d'aide à la prise de décision est reçue de la couche utilisateur et les données appropriées sont traitées, et produisent des résultats à l'utilisateur d'une voie plus complète. Elle va joindre les résultats avec des données semi ou non structuré issu de l'Internet ou intranet.

Des données sont extraites, transformées, et ici normalisées et fusionnées vers un seul endroit qui est le Data warehouse. Il y a également toutes les qualifications au sujet de la navigation de données et de leur manipulation et de faire des requêtes.

c) Couche ressources de données

Cette couche inférieure représente toutes les sources de données situées dans l'environnement de l'entreprise, il comprend les bases de données OLTP, Internet, intranet, etc.

Cette couche transforme les requêtes reçues de la couche intermédiaire en requêtes reconnues par la source pour l'interroger par un langage interne. De même, elle renvoie les résultats obtenus de ces requêtes vers la couche intermédiaire (par l'intermédiaire des agents ressources).

D'autres part, elle fait des recherches sur Internent et extrait des informations utiles, et l'envoi à la couche intermédiaire.

4.3.2. Types d'agents du modèle proposé

Nous avons classés chaque agent de notre approche selon la couche à laquelle il appartient, ainsi, il y'a :

- des agents demandeurs de données au niveau de la couche utilisateur, qui sont:
    - AU : Agent Utilisateur
    - AApp: Agent d'Application
- des agents fournisseurs de données au niveau de la couche ressources de données, qui sont:
    - AR : Agent Ressource
    - AM: Agent Moniteur
    - AEI: Agent Extraction d'Information
    - ARI: Agent Recherche d'Information
- des agents de traitement, de coordination des interactions, et de médiation, au niveau de la couche intermédiaire, qui sont:
    - ASuper: agent superviseur
    - AAdmin: Agent d'Administration
    - AInter: Agent Intermédiaire
    - ADecision: Agent de Décision

5. DESCRIPTION DETAILLEE DES AGENTS COMPOSANT NOTRE ARCHITECTURE

Chaque agent de notre architecture remplit une fonction spécifique, nous les décrivons un par un.

5.1. Agent Utilisateur (AU)

L'agent utilisateur, situé dans la couche utilisateur, est la porte d'entrée des utilisateurs au système, il interagit avec l'utilisateur en l'aidant à exécuter les activités d'aide à la décision. L'utilisateur peut fournir une description générale du problème en termes de buts et objectifs de niveau élevé. L'agent offre le résultat retourné, puis il réagit comme passerelles entre les utilisateurs et le système en termes : de questions qui peuvent être formulées, et d'explications exigées d'une vue des résultats finals.

L'agent utilisateur est persévérant et autonome dans le sens où il est capable de garder le profil de l'utilisateur au fur et à mesure que celui-ci utilise le système. Il est capable de stocker l'information pour l'utilisateur. Cet agent est la porte d'entrée au système, auquel plusieurs tâches sont attribuées :

- il contrôle l'autorisation d'utilisateur et décide s'il peut exécuter directement la demande.
- Il fournit à l'utilisateur le formulaire lui permettant de faire introduire ces demandes de service.
- Il permet l'extraction à partir des formulaires soumis, les éléments nécessaires pour construire une requête qui va être transmise par la suite vers le site intermédiaire.
- Il ramène au client les résultats d'exécution de la requête à partir du site intermédiaire.
- Il est responsable de recevoir les caractéristiques d'utilisateur, de maintenir également les préférences et le profil de l'utilisateur, et de faire automatiquement des requêtes de recherche à travers les sites intranet ou Internet sur le sujet d'intérêt de l'utilisateur en collaboration avec l'agent de décision.
- L'agent incorpore le résultat du processus décisionnel et produit des résultats finals d'une voie plus compréhensible, le rendant plus attrayant aux utilisateurs. Cet agent contient des modèles pré-formatés de rapports, aussi bien une suite des représentations visuelles pour exprimer les résultats.

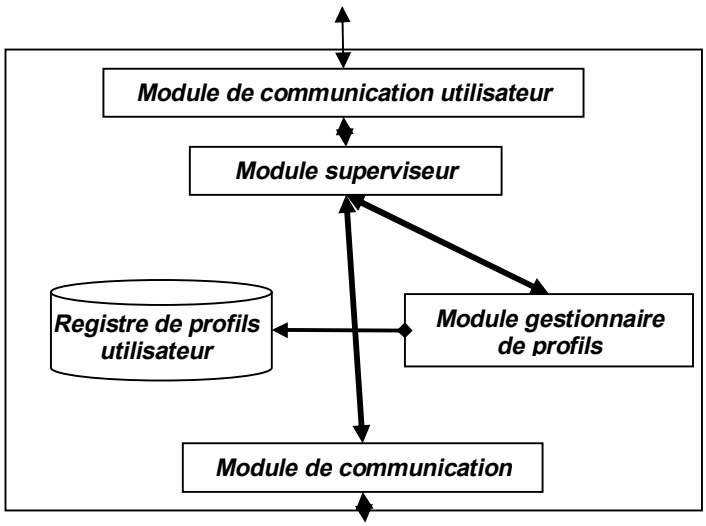


Figure 45 : Architecture de l'agent Utilisateur

5.1.1 Les Modules de l'agent Utilisateur

a) Module de communication utilisateur

Le module de communication utilisateur est responsable de la présentation des fonctionnalités du système sous forme d'une interface graphique. Il fournit à l'utilisateur le bon formulaire qui lui permettra de faire une requête facilement. Et concernant l'analyse, ce module permet la création conviviale de requêtes ; il s'agit en l'occurrence de choisir sur l'interface la mesure, l'opérateur à appliquer, ainsi que les attributs représentant les axes d'analyse.

L'exploitation du schéma de l'entrepôt permet de guider le choix de l'utilisateur sur l'interface en facilitant la création de la requête.

Ce module interagit par l'envoie des messages en réaction à des évènements utilisateurs sur l'IHM, il peut écouter les évènements utilisateurs de l'interface enveloppée, et de capter le choix de l'utilisateur et passer la requête au module superviseur. En fonction du bouton sélectionné, il publie une donnée, ou invoque une demande de requête en récupérant éventuellement les valeurs saisies dans les zones de saisie.

Ce module est chargé aussi de la présentation des résultats lorsqu'il reçoit un message contenant le résultat des requêtes qui leurs parviennent sous forme de documents XML, il affiche les données dans la zone d'affichage correspondante en se basant sur le profil de l'utilisateur et ses préférences de présentation.

b) Module superviseur

Ce module consiste au cœur de l'agent utilisateur, il a pour rôle d'analyser la requête de l'utilisateur. Ce module reçoit des données du module de communication utilisateur et détermine ensuite si toutes les informations nécessaires à la formulation de la requête sont disponibles. Dans l'affirmative, il construit un message et demande au module de communication de le transmettre à l'agent de décision. Dans la négative, il demande les informations complémentaires à l'utilisateur via un formulaire qui est transmis au module de communication utilisateur, ce dernier se chargera ensuite de le faire parvenir à l'utilisateur.

Il fait automatiquement des requêtes de recherche à travers l'intranet ou l'Internet sur le sujet d'intérêt de l'utilisateur, en analysant le registre de profils utilisateurs et avec l'agent de décision; qui stocke un cache de résultats sur les sujets trouvés dans le schéma d'entrepôt, et en temps réel quand l'utilisateur formule une requête.

Avant d'envisager d'étendre les possibilités d'analyse de l'entrepôt par l'utilisateur, il faut que l'utilisateur soit en mesure de connaître les possibilités d'analyse actuelles par la visualisation du schéma de l'entrepôt. Pour ce faire, le module superviseur demande à l'agent d'administration via l'agent de décision de générer le schéma de l'entrepôt sous forme d'un document XML en interrogeant la base de méta-données.

L'avantage de XML est de permettre aux utilisateurs de naviguer à travers les hiérarchies du modèle et de décrire correctement les possibilités d'analyse. Ainsi, ce document va permettre aux utilisateurs de les aider dans leur choix d'analyse. Cette fonctionnalité exploite donc le standard XML, évitant le recours à un outil de visualisation utilisant un format propriétaire. En effet, grâce a

ce document XML, il est possible de déterminer les faits, les dimensions, les hiérarchies, et l'ensemble des attributs qui les décrivent.

c) Module gestionnaire de profils

L'objectif est de pouvoir gérer les préférences des décideurs, pour se faire, la notion de profil est utilisée. L'intérêt de l'utilisation de profils peut être motivé à la fois : de la volumétrie des données connues dans l'entrepôt de données, et du rôle central que joue le décideur dans le processus décisionnel. En effet, ce module peut affecter à un utilisateur un profil en fonction de son poste et de disposer des différents renseignements concernant celui-ci.

Pour mieux exploiter le profil utilisateur tout en l'adaptant à nos besoins, nous le divisons en trois dimensions : La première dimension représente l'historique des interactions de l'utilisateur, la deuxième représente le centre d'intérêt de l'utilisateur, et la troisième représente les préférences de l'utilisateur. Ces profils sont ensuite utilisés par le module superviseur dans le processus de traitement de la requête. Le contenu du profil peut être utilisé de différentes façons, il peut remplacer la requête, permettre de l'enrichir (ajout de critères de sélection, et de nouveaux mots-clés), ou être utilisé pour adapter les résultats dans leur contenu (filtrage) ou bien dans leur forme de présentation.

En effet, dans le contexte de l'analyse en ligne, pour affiner la requête de l'utilisateur afin de mieux répondre à ses besoins, Il s'agit d'exprimer des préférences et de satisfaire des contraintes de visualisation dans la mesure où l'aspect visualisation est primordial. Et de permettre la personnalisation au niveau de la navigation, il s'agit de représenter les habitudes d'analyse de l'utilisateur sous forme de coefficient de préférences, pour faciliter sa navigation.

Ce module permet de stocker les préférences des utilisateurs. Il est en effet possible d'indiquer quel utilisateur courant et de maintenir à jour le nombre de fois de son choix d'une valeur donnée pour un attribut donné. En effectuant ce comptage, nous pouvons estimer une distribution de probabilité reflétant les préférences de l'utilisateur. La gestion de profils des utilisateurs correspond à un ensemble de fonctionnalités qui permettent à un utilisateur de créer, modifier, activer ou désactiver ses profils.

d) Module de communication

Il est se charge de l'envoi et de la réception et la compréhension des messages reçus. Ce module assure la communication avec les autres agents de notre système, donc il envoi les requêtes des utilisateurs et reçoit de l'agent de décision les résultats.

5.1.2 Fonctionnement de l'agent utilisateur

Premièrement le module de communication utilisateur permet la création conviviale des requêtes d'analyse, et ça c'est leur rôle principal, il capte les actions de l'utilisateur et envoie la requête au module superviseur qui va ensuite construire un message et demande au module de communication de le transmettre à l'agent de décision.

D'autre part, le module gestionnaire de profils utilisateur reçoit du module communication utilisateur les informations sur l'utilisateur et ses actions pour permettre de gérer la communication avec l'utilisateur suivant ses désires. Et permettre aussi l'affichage des résultats obtenus de la requête proprement dit et les résultats de recherche d'information externe, l'affichage se fait tout en suivant le profil utilisateurs et ses préférences du format d'affichage des résultats et sur le contenue qui l'intéresse.

5.2. Agent d'Application (AApp)

L'agent d'application peut être considéré comme l'agent utilisateur, mais l'utilisateur n'est pas un être humain, il est une autre application qui utilise le data warehouse comme source de données en vue d'aider leur utilisateur.

Cet agent est aussi de style adaptateur "wrapper" qui représente diverses applications d'analyse dans le système, agissant en tant que consommateurs d'information. Il encapsule généralement plusieurs types d'application, tels que :

- des applications OLAP de navigation dans les données multidimensionnelles qui permettent, avec des outils visuels, d'explorer un cube de données en se basant sur les opérations OLAP classiques telles que le forage vers le haut (roll-up), le forage vers le bas (drill-down), la sélection (slice) et la projection (dice).
- Des applications de datamining qui génèrent de nouvelles connaissances à partir des données de data warehouse.
- Ou bien des applications SIG qui permet à des utilisateurs de visualiser des données de la perspective géographique, basée sur des données telles que la l'altitude et la longitude. Les applications de SIG peuvent fournir la perspective géographique pour des données de météo en montrant leur zone de couverture sur la carte d'une région.

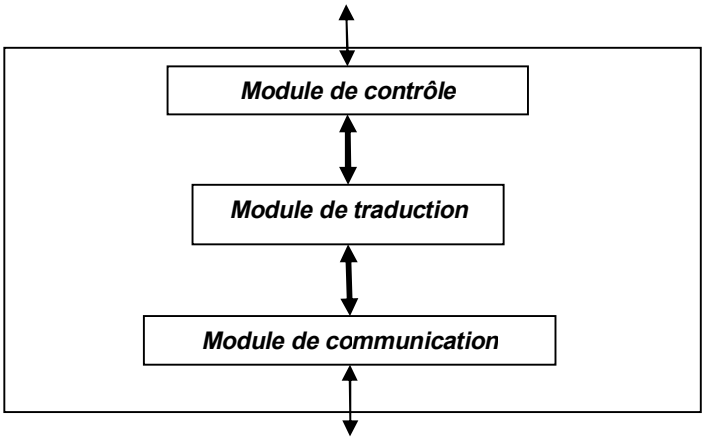


Figure 46 : Architecture de l'agent d'application

5.2.1 Les modules de l'agent d'application

Un agent d'Application est constitué de trois modules : un module de contrôle, un module de traduction, et un module de communication.

a) Module de contrôle

C'est le module de contrôle qui permet l'enveloppement du système logiciel. C'est à travers ce module que l'agent d'application va piloter et écouter le système logiciel. Il permet à l'agent d'application de se servir du système logiciel, de faire appel à des fonctions du système logiciel, de l'initialiser, de l'arrêter, de le relancer, de récupérer une donnée, … . Mais l'agent d'application doit également être capable de satisfaire les besoins de ce système logiciel. A travers ce module de contrôle, il est capable de détecter un besoin ou de lui faire parvenir une donnée attendue.

b) Module de traduction

Ce module va permettre l'interopérabilité de ce système logiciel avec notre système de data warehousig. Il fait la traduction entre les deux autres modules, il traduit les besoins du système logiciel, détectés par le module de contrôle, en messages ACL exploitables par le module de communication. Et inversement, il assure la traduction d'un message réceptionné par le module de communication à un appel de fonction qui va pouvoir être invoqué par le module de contrôle sur le système logiciel.

On peut citer les services de traduction comme suit:

- Du système logiciel au message ACL : lorsque le module de contrôle détecte, au sein du système logiciel qu'il enveloppe, un évènement devant remonter au niveau de système de data warehousing (besoin de service, publication de donnée…), il traduit cela en message ACL.

- Des messages ACL au système logiciel : lorsque l'agent d'application reçoit un message ACL devant avoir des répercussion au niveau du système logiciel qu'il enveloppe (réception d'une donnée…), il traduit le message ACL, et effectue les opérations correspondantes sur le système logiciel.
- Conversion de données : l'agent d'application doit être capable de convertir une donnée émanant de l'agent de décision au format nécessaire pour le système logiciel enveloppé, et inversement traduire des données provenant du système logiciel dans le format adapté.

c) Module de communication

Ce module va permettre au système logiciel d'interagir avec les autres agents. Ce module permet à l'agent d'application d'envoyer et recevoir des messages ACL. Il transfère, si nécessaire, au module de traduction les messages ACL reçus, et inversement, envoie au bon destinataire les messages ACL fournis par le module de traduction.

5.2.2 Fonctionnement de l'agent d'application

Il fournit plus qu'un pont fonctionnel à un système logiciel externe, il agentifie le système logiciel, et l'intègre au processus de data warehousing. Donc à travers le module de contrôle, l'agent d'application est capable de détecter un besoin ou de lui faire parvenir une donnée attendue par la formulation d'une requête pour ces données.

La traduction de la requête du modèle propre au système logiciel vers le modèle de notre système de data warehousing est le rôle du module de traduction. Ainsi le module de communication va envoyer cette requête à l'agent de décision puis attendre la réponse.

Quand une réponse reçue par le module de communication, elle doit être acheminé vers le module de traduction pour faire les traductions nécessaires à homogénéiser les données avec le modèle de données accepté par le système logiciel. Puis le module de contrôle envoie les données traduites au système logiciel contrôlé.

5.3. Agent de décision (Adecision)

L'agent de décision présente des capacités de médiation, acte en acceptant des tâches à partir des agents de la couche utilisateur, Il peut les décomposer et les envoyant à traiter pour l'exécution des requêtes dans l'agent intermédiaire, puis intégrant les résultats retournés par celui-ci, et les envoyer à l'agent utilisateur correspondant;

L'agent de décision a la connaissance au sujet du domaine de tâche, aussi bien que les capacités d'autres agents; il peut rechercher les services d'un groupe d'agents de recherche d'information semi ou non structurée provenant à partir des sources internes ou bien du Web.

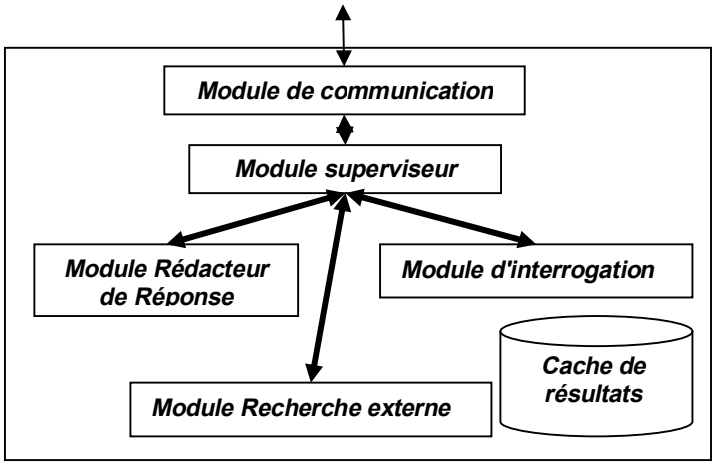


Figure 47 : Architecture de l'agent de décision

5.3.1 Les modules de l'agent de décision

a) Module Recherche externe

Ce module interagit avec l'agent Extraction d'Information par la formulation des demandes de recherche en exploitant l'ontologie; pour l'utilisation d'un maximum de termes, et en exploitant les résultats fournis par l'agent extraction; pour mettre à jour la cache de résultats qui stocke les informations extraites sur les sujets du schéma d'entrepôt.

b) Module d'interrogation

Ce module reçoit les requêtes de la couche utilisateur et permet la formulation des requêtes normalisées d'interrogation de l'entrepôt de données, on homogénéise les termes de la requête avec les termes utilisés dans la définition du schéma d'entrepôt, par l'utilisation de l'ontologie,

c) Module Rédacteur de Réponse

Il est chargé d'associer les réponses obtenues afin de formuler une seule réponse attendue par l'utilisateur à sa requête. Le rédacteur de réponse fournit à l'utilisateur une réponse au format XML, qui peut ensuite soit être utilisée par l'agent utilisateur pour l'afficher selon le profil de l'utilisateur (l'affichage peut se varier, certaines données peuvent être mises en avant ou bien au contraire occultées), ou bien être utilisée par une autre application via l'agent d'application.

d) Module superviseur

Ce module permet la supervision de fonctionnement de cet agent, il accepte les messages du module de communication puis vérifie son contenu pour l'envoie au module approprié, tel que :

- si le message est une requête d'interrogation du data warehouse, le donne au module d'interrogation et de recherche externe,
- si le message est une réponse le module rédacteur de réponse qui va suivre.

e) Module de communication

Il se charge de l'envoi et de la réception et la compréhension des messages reçus. Ce module assure la communication avec les autres agents de notre système, donc à partir les agents de la couche utilisateur, il reçoit des requêtes des utilisateurs et les renvoie au module superviseur, et aussi retourner vers celui-ci les résultats.

5.3.2 Fonctionnement de l'agent de décision:

Il reçoit les requêtes de la couche utilisateur par le module de communication qui va ensuite traiter le message et envoyer la requête au module superviseur, celui-ci, va l'envoyer au module d'interrogation pour la normalisation de la requête au format de data warehouse, ensuite il envoie la requête à l'agent intermédiaire via le module de communication et aussi envoie les termes composant la requête au module Recherche externe.

Le module Recherche externe va formuler la demande de recherche d'information externe via l'agent d'extraction d'information. Le module rédacteur de réponse et le module recherche externe tombent en état d'attente, et quand cet agent reçoit les réponses, les deux derniers modules doivent les rédiger au format XML pour l'envoyer à l'agent qui demande cette requête.

5.4. Agent d'Administration (AAdmin)

L'agent d'Administration est un agent de support au cycle de vie de notre système de data warehousing. Il fournit des interfaces d'interaction à l'administrateur, avec le système gérant l'entrepôt et avec les sources de données. Il fournit aussi des mécanismes pour construire l'entrepôt à partir de données stockées dans des sources multiples. Les activités sont réalisées de manière transparente, c'est à dire sans considérer l'implantation (relationnelle, multidimensionnelle, à objets) de l'entrepôt, ni l'hétérogénéité des sources auxquelles il faut accéder.

L'administrateur décrit un schéma multidimensionnel (formé par un ensemble de schémas de dimension et un ensemble de schémas de cubes) et c'est l'agent d'administration qui le traduit dans un schéma décrit en termes du modèle d'implantation, et ajoute des informations pertinentes (comme les attributs clés, les types d'attributs, etc.), et la compile dans une structure interne puis l'envoie à l'agent intermédiaire.

Il est la façade d'entrée et de la manipulation de méta-données de système qui enregistre toutes les parties importantes de la connaissance configurable :

- **des informations sur le système:** connaître des informations sur les agents, leurs services et états.

- **des informations sur les ressources d'information:** stocker l'information d'identité, l'information d'accès, ontologies locales et des descriptions du contenu des ressources externes.

Un autre rôle de l'agent d'administration consiste à maintenir à jour les schémas de bases de données dans celle de méta-données. Trois méthodes sont possibles pour cette mise à jour des schémas :

1. l'utilisateur demande par l'intermédiaire d'une requête prédéfinie la mise à jour des schémas de base de données. L'agent intermédiaire traite alors la requête, telle qu'elle en demandant aux agents moniteurs les nouveaux schémas de base de données;

2. l'agent d'Administration, de façon automatique et périodique, demande aux agents moniteurs, sur sa seule initiative, les nouveaux schémas de base de données;

3. un agent moniteur, suite à une mise à jour du schéma de la base de données à laquelle il est relié, envoie une notification de mise à jour à l'agent intermédiaire, ce dernier l'envoie à l'agent d'Administration.

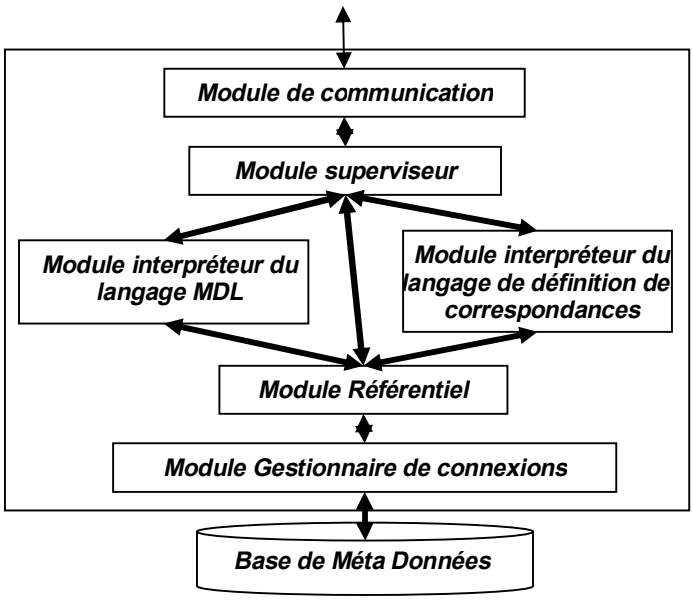


Figure 48 : Architecture de l'agent d'Administration

5.4.1 Les modules de l'agent d'Administration

a) Module de communication

Ce module assure l'intégration de cet agent au système SMA, donc il reçoit les messages d'interrogation et de modification de notre base de méta-données et aussi envoi les résultats aux agents tels que par exemple le schéma de l'entrepôt à l'agent de décision.

b) Module superviseur

Le Module superviseur assure la supervision de l'agent d'administration. Une fois que le référentiel a stocké le schéma de dimension ou le schéma de cube, ce module demande à l'agent intermédiaire de générer, à partir du schéma de la dimension ou du cube, l'expression pour créer la relation correspondante dans l'entrepôt. Et aussi quand la fin de définition des correspondances est terminée, de signaler le début de la construction de l'entrepôt.

c) Module interpréteur du langage de description de données multidimensionnel (MDL)

Il traite les expressions de création et de modification du schéma multidimensionnel, il autorise en particulier la création de schémas de dimension avec plusieurs niveaux et de schémas de cube avec une ou plusieurs mesures. Ce module traduit ces expressions en une représentation interne, Pour cela, il analyse d'abord les expressions MDL données en entrée afin de vérifier sa validité (Une expression MDL est valide si elle est conforme aux règles d'écriture du langage), si la vérification est positive, le schéma est stocké par le référentiel pour une utilisation ultérieure.

Les informations sur l'opération de la création d'un schéma de dimension ou d'un schéma de cube et ses arguments sont envoyés ensuite à l'agent intermédiaire. Cet agent traduit alors le schéma multidimensionnel en un schéma d'implantation.

d) Module interpréteur du langage de définition des correspondances

Il traite les expressions pour définir les correspondances et pour signaler le début de la construction de l'entrepôt. Ce module définit un ensemble de correspondances permettant d'associer les schémas de dimension et les schémas de cube à l'ontologie afin de récupérer les données pertinentes à leur construction. Une correspondance prend la forme d'une requête exprimée en termes de l'ontologie.

Considérons d'abord les correspondances entre un schéma de dimension et l'ontologie, dans ce cas, chaque niveau, chaque relation d'ordre entre deux niveaux, et chaque propriété est associé à l'ontologie par une correspondance spécifique :

- *Correspondance entre un niveau et l'ontologie*. Une correspondance entre un niveau *l* d'une dimension *d* et un terme de l'ontologie prend la forme d'une requête *q* exprimée en termes de l'ontologie. La requête *q* doit être spécifiée de telle façon que le résultat de son évaluation soit l'ensemble de membres du niveau *l*.
- *Correspondance entre une propriété et l'ontologie*. C'est une requête *q* sur l'ontologie. La requête *q* doit être spécifiée de telle façon que le résultat de son évaluation soit un

ensemble de paires (u,v) où u est un membre du niveau l , et v est la valeur de la propriété p pour le membre u.

- *Correspondance entre une relation d'ordre et l'ontologie.* Une correspondance entre une relation d'ordre entre deux niveaux l et l_2 d'une dimension d (tels que l_2 est père de l) et l'ontologie est une requête q. La requête q doit être spécifiée de manière à ce que le résultat de son évaluation soit un ensemble de paires (e_1,e_2) où e_1 est un membre du niveau l et e_2 est un membre du niveau l_2.

Considérons maintenant les correspondances entre un schéma de cube et l'ontologie. Rappelons qu'un schéma de cube est un n-uplet $(c, \{l_1,\ldots,l_s\}, \{m_1,\ldots,m_t\})$, où c est le nom du cube, l_i est un axe du cube et m_j est une mesure. Chacune de ces mesures est associée au schéma global par une correspondance spécifique. Une correspondance entre une mesure m d'un cube c et l'ontologie est une requête q exprimée en termes de l'ontologie. La requête q doit être spécifiée de façon à que le résultat de son évaluation soit un ensemble de n-uplets de la forme $(e_1,\ldots,e_s, v)$ où e_i est un membre de l'axe l_i et v est la valeur que la mesure m_j prend pour la combinaison de membres $e_1,\ldots,e_s$.

e) Le Module référentiel

Le référentiel effectue le stockage et la récupération des informations utilisées par notre système de data warehousing. Ces informations comprennent le schéma multidimensionnel de l'entrepôt et les correspondances entre ce schéma et l'ontologie et aussi les descriptions des schémas des sources. Ces informations sont stockées sous la forme de fichiers XML.

Cette base de méta-données contient l'ensemble des informations qui ne sont pas les données proprement dites, ces informations sont dites méta-données. Nous avons subdivisé cette base en un ensemble d'autres bases décrites dans ce qui suit :

e.1 Une ontologie de domaine partagée

Elle est considérée comme le schéma global sur lequel s'effectue la requête de l'utilisateur, et elle est un support pour la résolution des conflits sémantique inter-sources, à chaque élément des schémas sources correspond un seule élément de l'ontologie et ces correspondances sont stockées au niveau de la base de description des schémas.

C'est un support pour la génération de la requête de l'utilisateur et les requêtes de correspondance, elle est aussi consultée dans le cas d'un ajout ou d'une mise à jour d'une source de données pour effectuer les mises en correspondance nécessaires pour l'intégration de cette nouvelle source.

e.2 Base du schéma de l'entrepôt

Cette base stocke les schémas des dimensions et du Cube de l'entrepôt, cette base prend le schéma suivant:

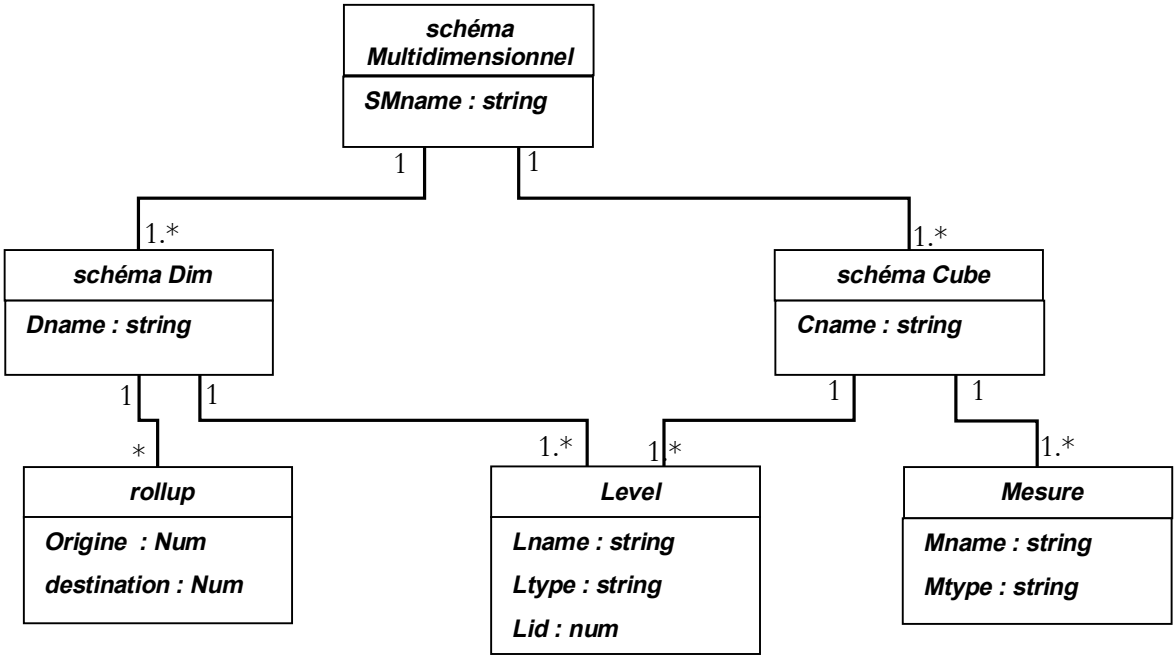


Figure 49 : Base du schéma de l'entrepôt

e.3 Bases de description des schémas des sources

C'est une base de données qui contient des informations descriptives sur les sources de données (tables, attributs, clés, etc.), et toutes les informations concernant les correspondances entre les éléments des sources et ceux de l'ontologie, et celles portant sur la localisation et l'organisation de ces éléments dans les sources de données. Pour chaque source de données, les correspondances avec l'ontologie sont effectuées dès sa mise à disposition pour l'intégration.

Elle est utilisée par l'agent superviseur afin de récupérer les correspondances nécessaires à la décomposition et réécriture de la requête globale sur les sources de données, elle est aussi mise à jour par l'agent d'administration en cas d'ajout, suppression ou mise à jour d'une source de données.

Nous décrivons cette base dans le schéma relationnel présenté dans la figure 24 :

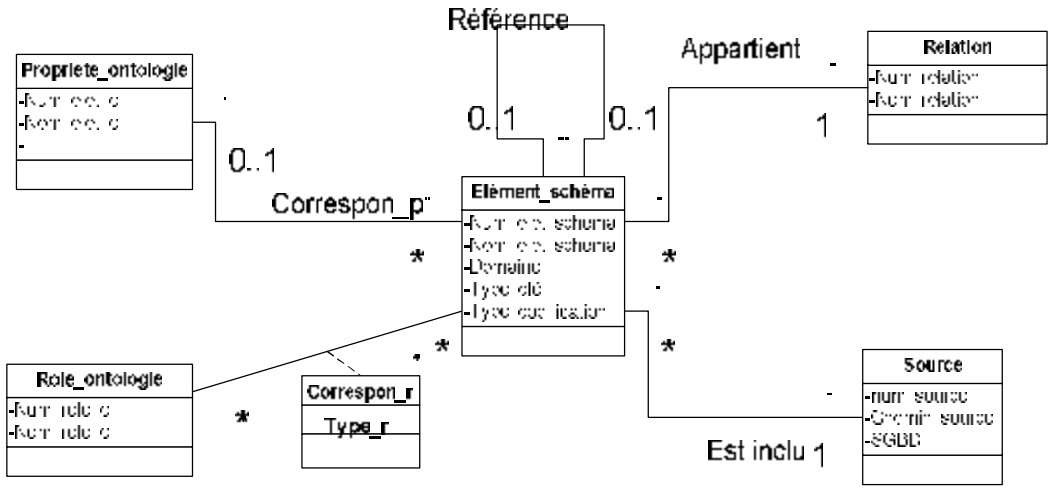


Figure 50 : Base de description des schémas

Cette base de données comporte les relations suivantes :

- **Propriete_ontologie** (Num_elet_o, Nom_elet_o)
- **Role_ontologie** (Num_role_o, Nom_role_o)
- **Source** (Num_source, Nom_source, Pilote, Chemin, Utilisateur, Mot_passe)
- **Relation** (Num_relation, Nom_relation)
- **Element_schema** (Num_elet_schema, Nom_elet_schema, Domaine, Type_cle, Type_codification, Type_role, Num_elet_o, Num_role_o, Num_source, Num_relation, Num_elet_schema1)

La mise en correspondance entre le schéma global et les schémas des bases de données est l'étape la plus importante dans les processus d'intégration car elle permet de mettre en relation la structure et la sémantique de ces différentes bases de données, dans notre cas nous avons choisi d'utiliser l'ontologie comme schéma global et par conséquent nous devrons lier les différents éléments des sources de données aux concept, rôles et propriétés de cette ontologie.

5.4.2 Fonctionnement de l'agent administration

Dans la création du schéma, L'administrateur décrit le schéma multidimensionnel de l'entrepôt à l'aide du langage MDL. Le module interpréteur du langage MDL vérifie que les expressions MDL en entrée soient valides, et une fois que cette vérification est effectuée, le schéma est envoyé vers le module référentiel pour le stocker et pour une utilisation ultérieure.

Une fois que le module référentiel a stocké le schéma de dimension ou le schéma de cube, le module superviseur demande à l'agent intermédiaire de générer, à partir du schéma de la dimension ou du cube, l'expression pour créer la relation correspondante dans l'entrepôt.

D'une même façon, l'administrateur définit les correspondances entre schémas de dimension et les schémas de cube à l'ontologie, en tant que un schéma global, sous la forme de requêtes XML exprimées en termes de l'ontologie. Puis le module référentiel les stocke dans la base de méta données.

5.5. Agent Superviseur (ASuper)

L'agent superviseur est un des principaux éléments du système d'entreposage, il est aussi l'un des plus complexes. Quant à lui, il reçoit les requêtes globales de l'agent intermédiaire, il est le responsable de l'exécution à haut niveau des requêtes, où il contient les programmes chargés d'analyser la requête global de l'utilisateur, de la décomposer en sous requêtes sur les sources pertinentes en utilisant les mises en correspondances de la base de description des schémas, ensuite il pose les requêtes appropriées à chaque source aux agents ressources afin qu'ils le répondent. En fin il est chargé d'associer les réponses obtenues afin de former une seule réponse attendue. Il passe ensuite les réponses à l'agent intermédiaire.

Pour cela, l'agent superviseur utilise une base de métadonnées. Cette dernière contient toutes les informations nécessaires à la décomposition de la requête, qu'elle va aider cet agent à déterminer:

- quelles bases de données interrogées,
- quelles données doit-on y rechercher,
- quelles données "annexes" (c'est-à-dire non spécifiées par l'utilisateur mais qui ont un lien avec certaines données recherchées, et qui par conséquent pourraient être utiles au décideur) sont à rechercher,
- si une sous-requête a besoin du résultat d'une autre sous-requête dans ses prédicats,
- dans quel ordre les sous-requêtes doivent être évaluées.

L'agent superviseur va alors élaborer :

- Les diverses sous-requêtes (une par base de données à interroger) au format XML,
- un filtre de décomposition, qui définit comment la requête principale peut être reconstituée à partir des sous-requêtes élaborées,

- un plan de séquencement qui donne l'ordre dans lequel les sous-requêtes doivent être exécutées, et si une sous-requête a besoin d'un résultat d'une autre sous-requête dans ses critères de sélection.

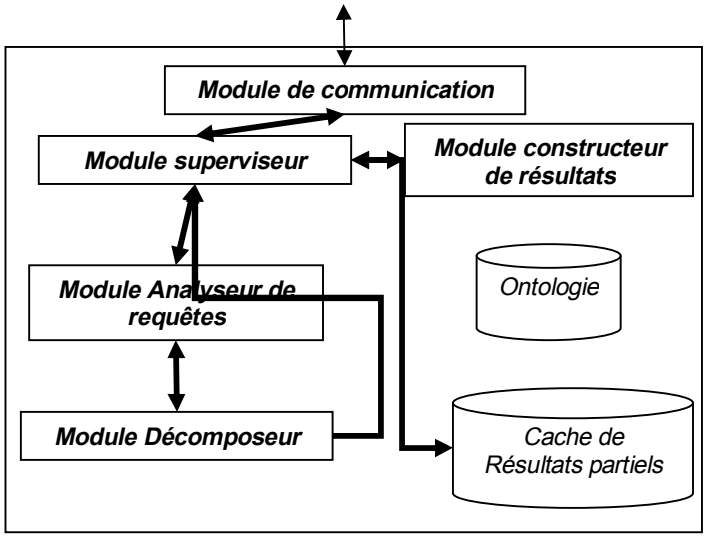


Figure 51 : Architecture de l'agent superviseur

5.5.1 Les Modules de l'agent superviseur

a) Module de communication

Ce module va permettre au système logiciel d'interagir avec les autres agents. Il permet à l'agent superviseur d'envoyer et recevoir des messages ACL. Il accepte les messages qui contient les requêtes puis les transfère au module superviseur, et inversement, envoie au bon destinataire les messages ACL fournis par le module superviseur.

b) Module Analyseur de requêtes

C'est le premier module qui va participer dans le processus de traitement de la requête qui consiste en premièrement d'analyser la requête et produire la structure de requête interne si la requête est syntaxiquement correcte et bien typée. Donc la fonction principale de ce module est la vérification de la validité de la requête globale XML. Cette analyse vise à vérifier la lexicographie et la syntaxe de la requête. Si aucune erreur n'est détectée lors de cette analyse, le résultat est une représentation interne de la requête pour faciliter sa manipulation.

Le module analyseur de requêtes peut extraire les différents composants de la requête globale qui permettront la décomposition de la requête après la détermination des éléments équivalents dans les sources de données, ainsi que la localisation des sources pertinentes sur lesquelles seront exécutées les sous-requêtes.

c) Module décomposeur de requêtes

Ce module raisonne sur les connaissances des sources contenues dans la base de métadonnées. En fonction des relations entre le schéma global et les schémas exportés, il décompose la requête globale de l'utilisateur en un ensemble de sous-requêtes applicables sur les sources locales.

Plusieurs étapes de décompositions peuvent être réalisé, on les cite comme suit :

c.1) Localisation des sources pertinentes

Cette étape qui vient après l'analyse consiste en la détermination des sources pertinentes capables de donner une réponse partielle ou totale à la requête globale. Pour réaliser cette étape, le module décomposeur de requêtes doit accéder à la base de description des schémas qui relie chaque propriété de l'ontologie à tous les attributs qui lui sont sémantiquement équivalents dans les sources locales.

Pour localiser les sources nous suivrons la procédure suivante :

- Premièrement soient les conventions, qu'on a acceptées dans le reste de ce mémoire :
  - LSi la liste des attributs de la source « Si »,
  - « Lp » l'ensemble des propriétés à projeter de la requête globale,
  - « Ld » la liste des propriétés composant les conditions de la requête globale,
  - « LAp » l'ensemble des attributs contenus dans toutes les sources sémantiquement équivalents aux propriétés de l'ensemble « Lp »,
  - « LAdi » l'ensemble des attributs équivalents aux propriétés de l'ensemble « Ld» contenus dans la source « Si ».
- On aura les cas de figures suivants :

Si $|Ld| \neq |LAdi|$ Alors la source « Si » n'est pas pertinente (Il est impossible de vérifier les conditions sur les propriétés projetées)

Sinon

1- ($LAp \cap LSi \neq \phi$) (il existe des attributs communs entre les attribut projetés et ceux de la source locale Si) :

Dans ce cas source locale Si est pertinente.

2- ($LAp \cap LSi = \phi$) (il n'existe aucun attribut commun entre les attributs projetés et ceux de la source locale Si) :

Dans ce cas la source locale Si n'est pas pertinente.

c.2) La décomposition de requêtes

Après la localisation des sources pertinentes, on procède à la décomposition de la requête globale en sous requêtes exécutables sur les bases de données et leurs réécritures d'une manière à être exécuté par chacun des agents ressources locaux.

Cette requête globale Q (Lp, Lc , Lr , Ld) doit être décomposée et traduite sur les sources locales :

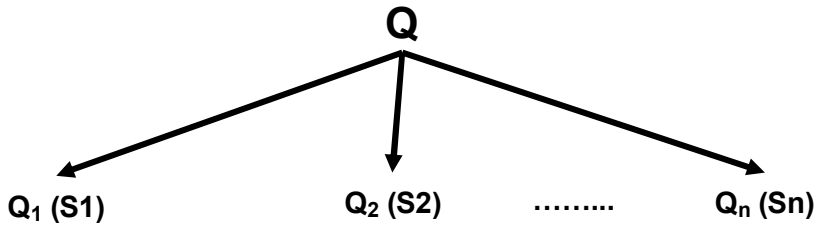


Figure 52 : Décomposition de la requête globale en sous requêtes

Q (Lp , Lc, Lr, Ld) = Q1 (La1, Lt1, Ld1 , Lj1) ® Q2 (La2, Lt2, Ld2 , Lj2) ® Q3 (La3, Lt3, Ld3,Lj3)®…

Avec

Q_i : Sous-requête correspondante à la source i.

L_{ai} : Liste des attributs sélectionnés de la source i.

L_{ti} : Liste des tables ou relations de la source i participantes à la requête Qi.

L_{di} : Liste des conditions sur les attributs.

L_{ji} : Liste des jointures à effectuer entre les tables participantes à la requête Qi.

® représente un opérateur ensembliste (jointure, union, intersection, différence, etc.).

Pour une source locale S_i donnée, sa requête doit être générée comme suit :

- L'ensemble des attributs à projeter est LAp ∩ LS_i.
- Ajouter à cet ensemble d'attributs projetés un ou plusieurs attributs nécessaires pour effectuer des jointures dans le cas où l'information recherchée est fragmentée entre plusieurs sources (voir 2ème cas).
- Pour uniformiser les résultats obtenus après l'interrogation des sources, on doit traduire dans la sous-requête les attributs sélectionnés par le nom de la propriété de l'ontologie qui leur est équivalente.
- Les tables de la source participante à la sous-requête sont celles contenant les différents attributs de LAp ∩ LSi en plus des tables contenant les attributs correspondant aux rôles de l'ontologie dans la requête globale qui participeront aux différentes jointures intra sources.

Les conditions de sélection sont celles de la requête globale en effectuant les mises en correspondances nécessaires avec les attributs locaux.

c.3) La réécriture de requêtes

On distingue trois cas dans la réécriture des sous-requêtes :

- **Cas 1 :** Les sources locales contiennent tous les attributs nécessaires à la satisfaction de la requête globale.
- **Cas 2 :** Les sources locales contiennent juste une partie des attributs nécessaires à la satisfaction de la requête globale mais peuvent être complétées par des attributs d'autres sources.
- **Cas 3 :** Les sources locales contiennent juste une partie des attributs nécessaires à la satisfaction de la requête globale et ne peuvent pas être complétées par d'autres attributs à partir d'autres sources.

1er Cas : Source ayant une réponse totale

Dans ce cas la réécriture de la sous-requête sur la source concernée comporte les étapes suivantes :

1. Remplacer toutes les propriétés projetées de la requête globale par les attributs correspondants dans la source locale mais en renommant les colonnes dans la requête par le nom des propriétés de l'ontologie.
2. Dans la partie FROM de la sous requêtes on écrit toutes les tables contenant au moins un élément des attributs correspondant à Lp.
3. On insère dans la partie WHERE de la sous requête les conditions de la requête globale dont les attributs correspondant aux propriétés la composant se trouvent dans la source concernée.
4. Dans le cas où il y a un rôle de l'ontologie dans la requête globale on a deux cas :
    - Si le rôle est équivalent à une clé étrangère (jointure entre deux tables avec clé étrangère) alors dans ce cas on doit ajouter la jointure correspondante dans la partie WHERE de la requête. Les deux clés composant cette jointure seront déterminées à partir de la base de description des schémas grâce à l'association « Référence ».
    - Si le rôle est équivalent à deux clés étrangères (jointure entre deux tables avec table d'association) alors dans ce cas on doit insérer dans la partie FROM de la requête locale la table d'association concernée par la jointure et dans sa partie WHERE les jointures correspondantes.

2ème Cas : Source avec une réponse partielle pouvant être complétée

Dans ce cas la source locale « Si » ne contient pas toutes les propriétés projetées de la requête globale mais peut être complétée par les propriétés manquantes à partir d'autres sources. Pour traiter ce cas là nous devrons concevoir **une procédure** qui détermine pour chaque source de

données les propriétés manquantes qui peuvent être complétées à partir des autres sources. Ajouter un attribut « a » à la source Si à partir de Sj n'est possible que si les hypothèses suivantes sont vérifiées :

- L'existence d'un attribut « a » sémantiquement équivalent à la propriété recherchée manquante à la source Si dans une autre source Sj.
- L'existence de deux attributs « ai » dans Si et « aj » dans Sj qui serviront de lien entre les deux sources pour importer l'attribut « a » manquant.
- Les deux attributs qui serviront de lien entre les deux sources doivent être sémantiquement équivalents, avoir le même domaine et le même type de codification on entend par là que si (ai=aj) alors il représentent la même instance de l'entité à laquelle ils appartiennent dans Si et Sj.
- L'attribut « ai » doit être soit une clé primaire soit une clé étrangère dans la source Si, quant à l'attribut « aj » il doit nécessairement être une clé primaire.

Dans ce cas la réécriture de la requête « Q_i » pour la source « S_i » génère en plus de la sous requête locale plusieurs sous-requêtes appliquées sur les différentes sources où se trouvent les propriétés manquantes à la source « S_i » pour cela cette réécriture doit suivre les étapes suivantes :

1. Pour les propriétés projetées de la requête globale « Q » ayant des attributs équivalents dans la source « S_i » on applique le même traitement que pour le **1er cas**.
2. Pour compléter les propriétés manquantes on doit suivre la procédure suivante :
   - **2.1** Pour chaque propriété « P_k » manquante on doit chercher dans la base de description des schémas les sources « S_j » qui peuvent compléter la source « S_i » avec la propriété « P_k »**.**
   - **2.2** Pour chaque triplé (S_j, a_i, a_j) obtenu par la procédure précédente on doit :
     - Dans la sous requête locale « Q_i» on doit ajouter dans la partie SELECT l'attribut « a_i ».

On génère une sous requête à deux colonnes la première étant l'attribut de liaison « a_j » et le second l'attribut de « S_j » correspondant à « P_i ».

3ème Cas : Source ayant une réponse partielle

Dans ce cas où la source interrogée ne répond qu'à une partie des propriétés de la requête globale et contrairement au deuxième cas les propriétés manquantes ne peuvent être ramenés à partir des autres sources avec des attributs de liaison inter-sources. Les résultats que pourront fournir ces sources seront pris en compte mais affichée comme étant incomplets. La génération de la sous requête locale à partir de la requête se fait selon les étapes suivantes :

1. On suit les mêmes étapes que pour la réécriture dans le premier cas pour les propriétés ayant des attributs correspondant dans la source locale.
2. Pour uniformiser le résultat obtenu à partir de cette source avec les résultats des autres sous requêtes posées sur les autres sources on ajoute dans la partie SELECT toutes les propriétés manquantes en leur attribuant la valeur « NULL ».

- Dans ce qui suit les deux algorithmes de décomposition et de réécriture que nous avons utilisé pour traiter ces trois cas de figures :

c.4) Algorithme de décomposition

Procédure Décomposition (Entrées : Lp, Lr, Ld, E;
Sorties : S_Req = {(Qi , Type_repi, E_jointurei)})

/ Lp, Lr, Ld : Liste des propriétés, rôles et conditions de la requête globale*/*
/ E : Ensemble des sources pertinentes */*
/ S_Req : Ensemble de sous requêtes générées globale */*
/ Qi : Sous requête associé à Si, elle se compose de :*
Qi.prop : Ensemble des propriétés contenu dans Qi.
Qi.roles : Ensemble des rôles contenu dans Qi.
*Qi.cond : Ensemble des conditions de la requête. */*
/ Type_reponsei : Type de réponse de la source i (1er , 2ème ou 3ème cas) */*
/ Procédure Equivp (pj, Si) : Procédure qui renvoie l'attribut de Si qui est sémantiquement équivalent à la propriété pj de l'ontologie */*
/ Procédure Equivr (pj, Si) : Procédure qui renvoie la clé étrangère de Si qui est sémantiquement équivalent au rôle rj de l'ontologie*/*
/ Procédure Lier_source(Si, pk) : Procédure présenté dans le 2ème cas */*
/ E_jointure : Ensemble des liaisons à effectuer dans le cas où le type de réponse de la source Si est du 2ème cas. */*

```
DEBUT
    S_Req ← ∅
    Pour Chaque source Si ∈ E faire
        Qi.prop ← ∅
        Qi.roles ← ∅
        Type_reponsei ← 1er cas
        E_jointurei ← ∅     /* Précise si la source Si est du 2ème cas */
        Qi.cond ← Ld
        Pour chaque pj ∈ Lp faire
            Si Equivp (pj, Si) ≠ ∅ alors      /* Attribut équivalent à pj dans Si */
              Qi.prop← Qi.prop + pj
            Sinon
              Si Lier_sources (pj, Si) = ∅ alors
                Type_reponsei← 3ème cas
              Sinon
                E_jointurei ← E_jointurei ∪ Lier_source (Si, pj)
              FSi
            FSi
        FPour
        Pour chaque rôle rj ∈ Lr faire
            Si Equivr (rj, Si) ≠ ∅ alors      /* Clé étrangère équivalente à rj dans Si */
              Qi.roles ← Qi.roles + rj
            Sinon
              Type_reponsei ← 3ème cas
            FSi
        FPour
        Si E_jointurei ≠ ∅ alors
          Type_reponsei ← 2ème cas
        FSi
        S_Req ← S_Req + (Qi , Type_reponsei, E_jointurei)
    FPour
FIN
```

Cette procédure permet de décomposer la requête globale en un ensemble de sous requêtes sur les sources locales contenant les éléments de l'ontologie pertinents à la sources, ainsi que le type de réponse de chaque source, la procédure suivante permet de la réécriture de chaque sous requête pour être exécutée au niveau des bases de données locales.

c.5) Algorithme de réécriture

```
Procédure Réécriture ( Entrées : S_Req ={ ( Qi, Type_reponsei, E_jointurei ) }
                          Sorties :  E_Req= { (Reqi, EJoini = {( Sreqj, Sj)}  } )
                    /* E_Req : Ensemble de requêtes  réécrites sur les sources */
                    /* Reqi : Requête réécrite sur la source i
                       Elle se compose des parties SELECT, FROM et WHERE          */
                    /* EJoini : Ensemble des sous requêtes inter-sources pour Si(2ème cas) */
                    /* Sreqj : Sous requête générée pour Si dans Sj */
DEBUT
    E_Req ← ∅
    Pour Chaque requête Qi dans S_Req faire
        Ejoini ← ∅
        Si (Qi.Type_reponsei = 1er cas ou 3ème cas) alors
            Pour chaque pj ∈ Lp faire                          /* pj propriété projetée */
                Si pj ∈ Qi.prop alors
                   Reqi.SELECT ←  Reqi.SELECT + Equivp (pj, Si)
                    Si Table (Equiv ( pj, Si)) ∉ Reqi.FROM alors
                     Reqi.FROM ← Reqi.FROM + Table (Equivp ( pj, Si))
                    FSi
                Sinon
                  Reqi.SELECT ← Reqi.SELECT + ' Null AS' + pj
                FSi
            FPour
        Sinon                                                  /* Cas avec liaisons inter-sources */
            Pour chaque pj ∈ Lp faire                  /* pj propriété projetée */
                Si pj ∈ Qi.prop alors
                    Reqi.SELECT ←  Reqi.SELECT + Equivp (pj, Si)
                    Si Table (Equivp ( pj, Si)) ∉ Reqi.FROM alors
                     Reqi.FROM ← Reqi.FROM + Table (Equivp ( pj, Si))
                    FSi                                    /* Table (A) ≡ table contenant A */
                Sinon
                    Si  Lier_source(Si, pj) ≠ ∅ alors
                       Générer les sous requêtes inter sources dans EJoin i /* voir cas 2*/
                    Sinon
                        Reqi.SELECT ← Reqi.SELECT + ' Null AS' + pj
                    FSi
                FSi
            FPour
        FSi
        Pour chaque dj ∈ Ld faire
            Reqi.WHERE ← Reqi.WHERE + Equivp (dj, Si)
        FPour
        Pour chaque rj ∈ Lr faire
            Reqi.WHERE ← Reqi.WHERE + jointure (rj, Si) /* Jointure intra-source */
        FPour
        E_Req← E_Req  +  (Reqi, EJoin i)
    FPour
FIN
```

d) Module superviseur

Le Module superviseur a deux fonctions principales. La première consiste à envoyer pour exécution chacune des sous-requêtes générées par le module décomposeur de requête à l'agent ressource correspondant. La deuxième fonction de ce module est de récupérer les résultats partiels (sous la forme de fichiers de données XML) produits par les différents agents ressource.

Il détermine dans quel ordre envoyer les sous-requêtes et déterminer si une sous-requête nécessite le résultat d'une autre sous-requête dans ses prédicats de recherche. Toutes les sous-requêtes ne nécessitant pas de modifications et pouvant être traitées en parallèle sont envoyées en même temps aux bases de données concernées.

Enfin, ce module récupère les résultats et les stocke temporairement dans un cache sous la forme de fichiers. Ce cache est un élément interne de cet agent, et il est mis à la disposition du constructeur de résultats lors du calcul du résultat global. L'ensemble des sous-requêtes et de leur réponse est envoyé à l'agent intermédiaire, après réception de la dernière réponse.

Si un agent ressource ne répond pas, il existe un délai de réponse qui permet de déterminer une éventuelle absence de réponse. Dans ce cas, l'agent demandant est averti du problème et les réponses déjà obtenues sont envoyées à celui-ci.

e) Module de recomposition du résultat

Après que chaque agent ressource d'une source pertinente ait exécuté localement sa requête générée par l'agent superviseur, il envoi le résultat obtenu à ce dernier qui reçoit le résultat de chaque source, ensuite il fusionne les résultats partiels obtenus par le module superviseur de requêtes en un résultat global. Chacun des trois cas de réécriture de requête vus dans les sections précédentes a un traitement particulier :

- Pour le 1er cas (Sources satisfaisant totalement la requête) on doit faire une union entre tous les résultats des sous-requêtes.
- Pour le 2ème cas les liaisons entre toutes les sous-requêtes générées pour la source locale interrogée doivent suivre l'ordre et le processus suivant :

  Avant d'expliquer le processus on adopte les notations suivantes :

  - Qi : La sous requête générée pour la source « Si » concernée par l'interrogation à laquelle il manque des propriétés pour satisfaire la requête globale. Les colonnes de la requête «Qi» contiennent tous les attributs de la source locale « Si » participants à la liaison avec d'autres éléments dans les différentes sources permettant de récupérer les propriétés manquantes.

- Qjk : Sous requête générée pour récupérer l'attribut manquant « pj » à partir de la source «Sk».
- R (Q) : Résultat de la requête « Q ».

Le processus est le suivant :

Pour chaque propriété pj manquante faire

- Pour chaque source « Sk » contenant « pj » et pouvant compléter «Si » faire

  Rjk ← R (Qi) **join** R (Qjk)

  Soustraire de Rjk la colonne de « Sk» participante à la jointure

  Fin pour

  R (Qi) ← L'« UNION » de tout les Rjk

Fpour

- Resultat_final ← R(Qi) – Les colonnes ajoutée à Qi pour faire des jointures et qui ne correspondent à aucune des propriétés projetées de la requête globale.

- Pour le ***3ème*** **cas** les résultats obtenus à partir de l'interrogation des sources locales seront incomplets. On doit aussi faire l'union entre les différents sous résultats de ce type et on aura une table dont les colonnes sont les propriétés projetées mais contenant des lignes incomplètes par rapport à une ou plusieurs propriétés.
- Le résultat final de la requête est l'union de tous les sous résultats obtenus pour les trois cas précédents sous forme d'un fichier XML stockant les données fusionnées.

5.5.2 Fonctionnement de l'agent superviseur

Cet agent de traitement des requêtes issu des agents intermédiaire ou de décision a été implémenté grâce à un ensemble de modules qui assurent l'analyse, la réécriture des sous-requêtes sur les sources de données locales, la recomposition du résultat final en vue d'être envoyé à leur destination. Il est participé dans le traitement d'une requête de médiation dans notre système et il est décomposé dans les étapes suivantes :

- Quand le module de communication reçoit un message contenant une requête, il l'envoie au module superviseur pour l'acheminer au module analyseur de requêtes qui met à sa disposition les différents concepts, propriétés et rôles de l'ontologie de domaine utilisée, présents dans la base de description des schémas.

- Après la réception de la requête analysé par le module décomposeur de requêtes, celui-ci interroge la base de description des schémas afin d'obtenir les information nécessaire à la décomposition et la réécriture de la requête sur les sources de données locales. Ces informations sont aussi nécessaires à la détermination des sources pertinentes qui seront l'objet de l'interrogation.
- Après la détermination des sources pertinentes, la décomposition et la réécriture de la requête, le module décomposeur des requêtes envoie l'ensemble des sous requêtes générées sur les sources de données au module superviseur pour le renvoyer aux agents ressources chargés d'exécuter chaque sous-requête sur chaque source de données pertinente correspondante.
- Puis le module superviseur reçoit tous les sous résultats récupérés par les agents ressources au module de traitement de la requête et les stocker dans le cache de résultats partiels, et ensuite le module de recomposition du résultat se charge d'effectuer la recomposition et la fusion des sous résultats par l'union de tous les sous résultats récupérés.

5.6. Agent Intermédiaire (AInter)

L'agent intermédiaire a pour principal rôle de faciliter l'initialisation et la maintenance des instances des schémas multidimensionnels des dimensions et des cubes de data warehouse, en calculant quelles modifications de source ont besoin d'être propagées.

Cet agent qui reçoit des notifications de changement à partir des agents moniteurs doit intégrer les changements dans l'entrepôt, et il compte un ensemble cohérent de mise à jour aux cubes enregistrés dans l'entrepôt. L'agent intermédiaire, exécute alors le traitement nécessaire pour intégrer le changement dans l'entrepôt, pendant ce traitement, il peut devoir envoyer des requêtes à l'agent superviseur de nouveau, pour obtenir des données supplémentaires de la même ou d'autres sources.

Un agent Intermédiaire est le responsable de l'intégration des données des sources dans l'entrepôt. Donc l'intégration de ces données hétérogènes dans un schéma existant est un problème difficile exigeant plusieurs étapes. D'abord, les données doivent être conformes au schéma conceptuel employé par le data warehouse, puis doivent être fusionnées avec les données déjà existants.

Quand il reçoit de l'agent Moniteur une notification de changement de schéma d'une source de données, il va l'envoyer vers l'agent administrateur pour effectuer cette tâche. Donc essentiellement, cet agent est là où:

- des différences à travers les sources d'informations sont indiquées,
- des relations entre les sources de données multiples sont définies,
- des duplications et des incohérences sont détectés, et
- on détermine comment l'information sera intégrée dans l'entrepôt.

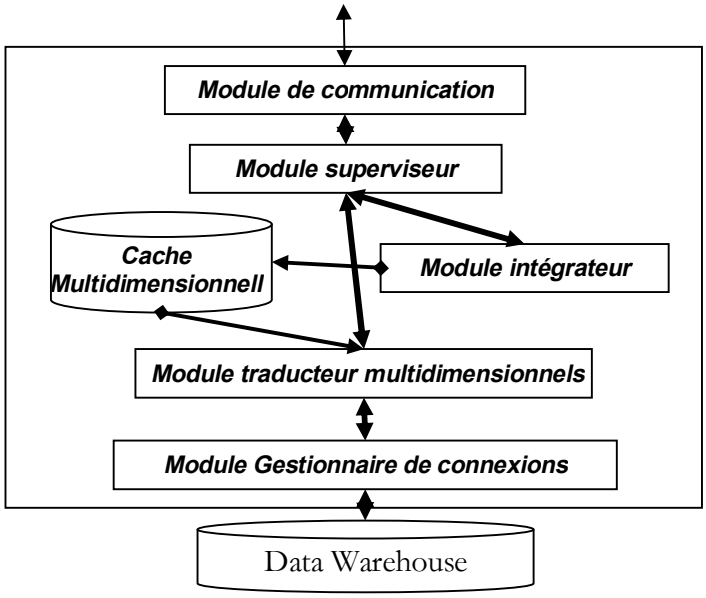


Figure 53 : Architecture de l'agent Intermédiaire

5.6.1 Les Modules de l'agent Intermédiaire

a) Module de communication

Ce module assure l'intégration de l'agent Intermédiaire dans notre système SMA, il assure la réception des messages envoyés par des autres agents, et les passe au module superviseurs pour les traiter, et vis versas.

b) Module superviseur

Une fois l'agent administrateur stocke le schéma de dimensions et le schéma de cube, le module superviseur demande au module traducteur multidimensionnel de générer, à partir du schéma de la dimension ou du cube, les expressions appropriées pour créer la relation correspondante dans l'entrepôt.

D'autre part, quand le module intégrateur signale à ce module la fin du processus d'intégration de données. Le module superviseur indique à la fois au module traducteur

multidimensionnel que les données intégrées, trouvées dans le cache des résultats, sont prêtes pour être stockées dans l'entrepôt.

Un autre activité de ce module et respectivement de l'agent intermédiaire est de fournir une interface pour l'interrogation de l'entrepôt par l'agent de décision, donc ce module accepte des requêtes et l'envoie au module traducteur multidimensionnel, puis il collecte les résultats et l'envoie via le module de communication vers l'agent de décision.

c) Module intégrateur

Le but de ce module est la création d'une instance du schéma multidimensionnel de l'entrepôt, donc il calcule les données à stocker dans l'entrepôt. Pour cela, il récupère auprès des sources les données nécessaires à la construction de l'entrepôt et leurs structures sous la forme de dimensions ou de cubes. Lorsque le début de la construction est signalé par le module superviseur, l'intégrateur construit d'abord les instances des schémas de dimension et ensuite les instances des schémas de cube. Les données nécessaires à la création de ces instances sont extraites des sources grâce aux correspondances qui associent les schémas de dimension et les schémas de cube à l'ontologie.

L'intégrateur demande alors à l'agent superviseur d'évaluer les requêtes définies dans les correspondances afin de récupérer les données nécessaires à la construction. On va illustrer le processus de construction comme suit :

- *Pour construire une dimension*, l'intégrateur calcule les membres de tous les niveaux du schéma de la dimension et forme avec eux la hiérarchie d'agrégation. L'intégrateur récupère d'abord les membres du niveau bas du schéma de la dimension, pour cela, l'intégrateur demande à l'agent superviseur d'évaluer la requête associée à ce niveau. Pour chaque membre *m* récupéré, il cherche ensuite à déterminer le membre *m'* du niveau immédiatement supérieur *l'* auquel le membre *m* est associé. Pour cela, l'intégrateur demande à l'agent superviseur d'évaluer la requête associée à la relation d'ordre entre le niveau bas et le niveau *l'*. L'intégrateur cherche ensuite à déterminer pour chaque membre *m'* du niveau *l'* le membre *m"* du niveau *l"* (supérieur à *l'*) auquel *m'* est associé. Ce processus continue jusqu'à atteindre le niveau plus haut du schéma de la dimension.
- *Pour construire un cube*, l'intégrateur calcule les cellules qui le forment. Pour cela, l'intégrateur demande à l'agent superviseur d'évaluer la requête associée à la mesure du cube. L'intégrateur récupère ensuite le résultat de l'évaluation de la requête (un ensemble de cellules) et le stocke dans le cache. Nous avons défini le schéma du cube comme un n-uplet

de la forme (c, {l_1,.. . ,l_s}, {m_1,. . . , m_t}), où *c* est le nom du cube, l_i sont les axes du cube et m_i sont les mesures. Pour construire un cube conforme à un schéma de cette forme, il faut:

i. évaluer les requêtes associées aux mesures du cube,
ii. fusionner les résultats partiels.

Le résultat d'exécution d'une requête *q* associée à une mesure *m*, est un ensemble de n-uplets de la forme (e_1, . . . , e_s, v); où e_i : un membre de l'axe l_i et *v* : la valeur que la mesure *m* prend pour la combinaison de membres e_1, . . . , e_s.

L'objectif de l'étape de fusion des résultats partiels est de créer enfin les cellules du cube de la manière suivante:

- $\{ (e_1, \ldots, e_s, v_1, \ldots, v_t) \mid (e_1, \ldots, e_s, v_1) \in r(q_1) \wedge \ldots \wedge (e_1, \ldots, e_s, v_1) \in r(q_t)\}$,
- où $r(q_i)$ représente l'ensemble de n-uplets résultat de l'exécution de la requête q_i.

- L'intégrateur stocke les dimensions et les cubes qu'il construit dans un cache multidimensionnel. Ce dernier prend la forme d'une collection de fichiers XML temporaires, que l'intégrateur supprime une fois les dimensions et les cubes sont stockés de manière définitive dans l'entrepôt.

D'autre part, quand l'agent moniteur envoie les mises à jour à l'agent intermédiaire, le module superviseur va l'acheminer vers le module intégrateur qui va faire les calculs appropriés; pour indiquer quelles modifications de sources ont besoin d'être propagées dans les cubes définissant l'entrepôt, et quelle mesures affectées par ces modifications, et ensuite générer les requêtes pour collecter les informations nécessaires au calcul de ces mesures, enfin le module intégrateur stocke les résultats dans le cache.

d) Module traducteur multidimensionnel

Une fois les dimensions et les cubes sont prêts pour être stockés dans l'entrepôt, il faut les traduire en données représentées dans le modèle d'implantation de l'entrepôt. Par exemple, si le modèle d'implantation est le modèle relationnel, il faut traduire les dimensions et les cubes en relations. Ce module récupère du cache Multidimensionnel les données intégrées et génère les expressions adéquates de langage de manipulation de données comme les expression SQL.

Pour insérer les données dans l'entrepôt. Ce module est procédé en deux modes:

- *la traduction de schémas multidimensionnels*, c'est le premier mode de ce module qui interagit avec le module gestionnaire de connexions afin de créer et modifier le schéma d'implantation en étoile de l'entrepôt. Ce module implante les règles pour traduire le schéma d'une dimension ou d'un cube en un schéma approprié du model utilisé pour la modélisation

de l'entrepôt. Par exemple, le résultat de la traduction est une expression SQL pour créer une table de dimension.

- *la traduction de données multidimensionnelles*, dans ce deuxième mode, le module interagit également avec le module gestionnaire de connexions afin de stocker dans l'entrepôt les données des dimensions et des cubes construits par le module intégrateur, trouvées dans le cache multidimensionnel. Par exemple, si l'entrepôt est implémenté suivant le model relationnel, le résultat de la traduction est un ensemble d'expressions SQL de la forme insert into r (<liste_attributs>) values (<liste_valeurs>), où *r* : le nom d'une relation, liste_attributs : la liste d'attributs de la relation *r* et liste_valeurs : les valeurs que prennent ces attributs.

e) Module Gestionnaire de connexions

Similaire à l'agent ressource et l'agent moniteur, le module gestionnaire de connexions gère l'accès à l'entrepôt, quand les expressions générées sont envoyées vers ce module pour être exécutées. Ce module est en effet un client JDBC du serveur SGBD de l'entrepôt. Il crée une connexion JDBC au serveur pour chaque demande d'exécution d'une expression générée, et ferme la connexion à la fin de cette exécution.

5.6.2 Fonctionnement de l'agent intermédiaire

Cet agent fonctionne en trois modes, qui peuvent être cités comme suit: la construction de l'entrepôt, la détection de mise à jour périodiquement, et l'interrogation de l'entrepôt.

- ***la construction de l'entrepôt***

Lorsque le début de la construction est signalé, le Module superviseur récupère de l'agent administrateur les correspondances entre le schéma multidimensionnel et l'ontologie (le schéma global). Le module intégrateur utilise ces correspondances pour récupérer à partir de sources les données nécessaires à la construction.

La fonction du module intégrateur est de calculer les données à stocker dans l'entrepôt. Pour cela, il récupère auprès des sources les données nécessaires à la construction de l'entrepôt et leurs structures sous la forme de dimensions ou de cubes. Le module intégrateur stocke les dimensions et les cubes qu'il construit dans un cache multidimensionnel (une collection de fichiers temporaires).

Puis le module traducteur multidimensionnels traduire les données trouvées dans le cache en données représentées dans le modèle d'implantation de l'entrepôt, et interagit avec le module Gestionnaire de connexions afin de stocker les données calculées dans l'entrepôt.

- ***la détection de mise à jour périodiquement***

Quand l'agent moniteur envoie les mises à jour à l'agent intermédiaire, le module superviseur va l'acheminer vers le module intégrateur qui va faire les calculs appropriés; pour indiquer quelles modifications de sources ont besoin d'être propagées dans les cubes définissant l'entrepôt, et quelle mesures affectées par ces modifications, et ensuite générer les requêtes pour collecter les informations nécessaires au calcul de ces mesures, enfin le module intégrateur stocke les résultats dans le cache.

Comme dans la construction de l'entrepôt, le module traducteur multidimensionnel traduit les données trouvées dans le cache en données représentées dans le modèle d'implantation de l'entrepôt, et interagit avec le module Gestionnaire de connexions afin de stocker les données calculées dans l'entrepôt.

- ***L'interrogation de l'entrepôt***

Une autre activité de cet agent est de fournir une interface pour l'interrogation de l'entrepôt par l'agent de décision, pour cela, le module superviseur accepte des requêtes et les envoie au module traducteur multidimensionnel, puis il collecte les résultats et les envoie via le module de communication vers l'agent de décision.

5.7. Agent Ressource (AR)

Afin de préparer l'intégration des données dans l'entrepôt, il est nécessaire de convertir les données à envoyer dans un format plus ou moins commun. Pour ce fait, Ces agents sont responsables de traduire différents formats des sources de données au modèle commun de données du système de data warehousing.

Le besoin fonctionnel auquel doit répondre un Agent ressource est de standardiser la syntaxe et la sémantique d'une source de données locale en les convertissant au format de la médiation. Ce service offre une solution de diffusion de l'information, permettant de conserver l'autonomie locale des sources de données. Ils permettent la traduction de chaque sous requête dans le langage local de la source concernée, l'interrogation de celle-ci par l'exécution de la requête, la récupération du résultat des sous requêtes et sa traduction dans le format commun du niveau médiateur.

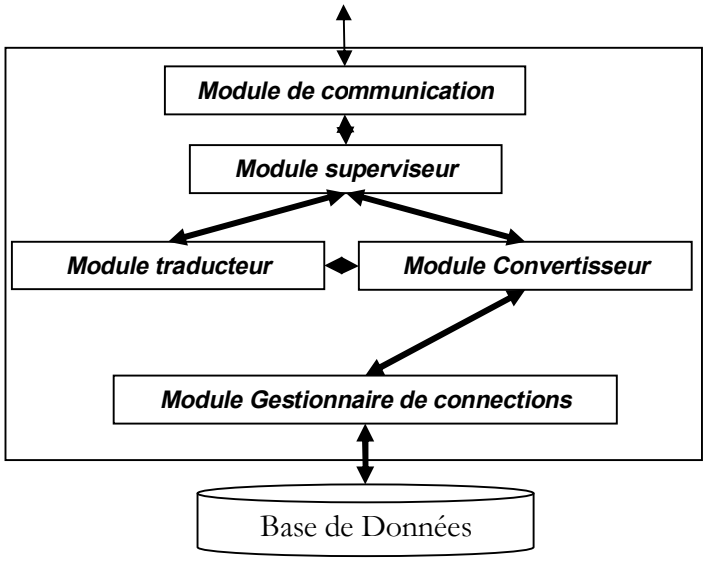


Figure 54 : Architecture de l'agent Ressource

6.7.1 Les Modules de l'agent Ressource

a) Module de communication

Il est, comme ses équivalents dans les autres agents, responsable du traitement des messages entrant, et de l'élaboration des messages sortant. C'est l'interface de l'agent.

Il reçoit des autres agents les requêtes sous la forme d'un message et suite à cela, il appelle le module superviseur. Il reçoit également des demandes de transmission de messages du module de traitement. Ces demandes de transmissions constituent des réponses aux requêtes reçues.

b) Module superviseur:

Le Module superviseur assure la supervision de l'agent Ressource. Plusieurs rôles lui sont dévolus. Tout d'abord, il est là pour recevoir les requêtes en provenance du Module de communication et il est chargé de faire suivre la sous-requête au bon module, ce que soit le module Convertisseur, traducteur, ou bien de l'envoie à l'agent moniteur lorsque les schémas des bases de données doivent être mis à jour.

Il construit ensuite une réponse sous forme de message qu'il donne au module de communication pour être expédiée.

c) Module gestionnaire de connections

Ce module a deux fonctions principales. La première est l'acheminement de la requête générée par le module convertisseur vers la source pour son exécution. La deuxième fonction est la récupération du résultat de l'exécution de la requête. C'est un Connecteur aux différentes sources de données (ODBC, JDBC, Native...), il doit être fourni par le constructeur de source d'informations,

ou soit codé par le programmeur dans le pire de cas; C'est le responsable de rechercher des données demandées par d'autres modules.

Pour construire ce pilote, nous avons utilisé un pont JDBC (Java Data Base Connectivity) / ODBC (Open Database Connectivity). Il crée une connexion JDBC à la source pour chaque demande d'exécution d'une requête et la ferme à la fin de cette exécution. Une fois que la connexion a été établie, ce module envoie la requête à la source pour son exécution et reste en attente des résultats. une fois que le résultat de l'exécution de la requête est disponible, Il récupère l'ensemble de n-uplets,. Ce résultat est ensuite mis à la disposition du module convertisseur.

d) Module Convertisseur

Le principal rôle du module convertisseur dans notre approche est la conversion schématique et syntaxique des données au format du système data warehousing et de résoudre les conflits dans la représentation de données. Par exemple, nous pouvons utiliser différentes représentations des mêmes données de type DATE dans les bases de données, comme "1975-03-19", ainsi le module prend la responsabilité de la transformation en même modèle que celui du data warehousing, par exemple, 19-mar-75.

Si le système utilise un même schéma que celui de la source d'information, le module superviseur doit directement communiquer avec le module convertisseur sans invoquer le module traducteur.

Quand une requête est arrivée au module convertisseur, une analyse est effectuée pour trouver les conflits entre différents formats de données et pour les convertir en celui de la source d'informations. Alors la requête traduite sera envoyée au module gestionnaire de connexion. Une fois que le résultat est trouvé et retourné par le module gestionnaire de connexion, le module convertisseur convertira la représentation de données.

e) Module Traducteur

C'est le module qui caractérise l'agent ressource. il a comme premier rôle de traduire un langage de manipulation de données à un autre langage compréhensible par la base de données à laquelle l'agent ressource est relié, qui peut être du relationnel, objet, réseaux, … pour ne citer que quelques-uns. La sous-requête traduite est alors envoyée au module convertisseur de la base de données à interroger.

Le second rôle est La réception de la réponse renvoyée au module traducteur qui va effectuer l'opération inverse de traduction. C'est donc une transformation de données de type interne en celui que le système de data warehouse peut identifier (document XML). La réponse traduite,

que le module superviseur reçoit, est donc au format d'un document XML, Il la fait suivre à l'agent superviseur, qui peut alors la traiter, sous format du message ACL via le module de communication.

6.7.2 Fonctionnement de l'agent ressource

Lors de l'interrogation d'une source de données, l'agent superviseur envoie à l'agent ressource la sous-requête qui lui est destinée. Le module superviseur de ce dernier informe d'abord l'agent moniteur de détecter les mises à jour. L'agent moniteur détecte les modifications de la source d'informations et envoie les mises à jour à l'agent intermédiaire. Le module superviseur de l'agent ressource n'envoie pas la requête au module traducteur jusqu'à ce que la détection de mise à jour se termine. Quand l'agent moniteur termine sa détection, il informera l'agent ressource de continuer l'exécution de la sous-requête.

Le module superviseur est chargé de faire suivre la sous-requête au bon module, en l'occurrence ici au module traducteur si le langage d'interrogation de la base de données est celui de notre système qui est XML. Ce dernier va traduire la sous-requête en un langage compréhensible (SQL, par exemple, pour le relationnel) par la base de données à laquelle l'agent ressource est relié. La sous-requête est alors envoyée au module convertisseur qui va détecter les conflits dans la représentation des données, puis il va transmettre la sous-requête convertie à la base de données via le Module gestionnaire de connexions. La réponse de ce dernier est renvoyée au convertisseur, et au traducteur pour effectuer l'opération inverse de la conversion et traduction. C'est donc une réponse au format XML que le module superviseur de l'agent ressource reçoit, il la fait suivre à l'agent superviseur, qui peut alors la traiter, via le module de communication.

5.8. Agent Moniteur (AM)

L'agent moniteur est responsable de la détection des modifications des données sources et de leur notification à l'agent intermédiaire. Il existe un agent moniteur pour chaque source. Il faut être capable d'interroger périodiquement chaque base locale ou son journal afin de récupérer les mises à jour effectuées durant la dernière période.

Le rôle du Moniteur est donc de détecter les mises à jour des sources locales et de réaliser l'extraction des données afin de les préparer pour l'émission.

Il se charge aussi pour l'ajout d'un schéma d'une nouvelle base de données, et lorsque les schémas des bases de données doivent être mis à jour. Cet agent récupère le schéma de sa base de données, au format interne de la base de données, et le traduit en XML. Ce schéma est enrichi notamment par le nom de la base de données traitée ainsi que son emplacement, puis il

homogénéise ce schéma. Il peut ensuite envoyer le schéma de la base de données au format XML à l'agent intermédiaire.

Cet agent gère le stock des informations concernant la source (méta-données de source), comme le chemin complet de la table de données (adresse de réseau, nom de base de données, protocole de connexion) et les identités de relation.

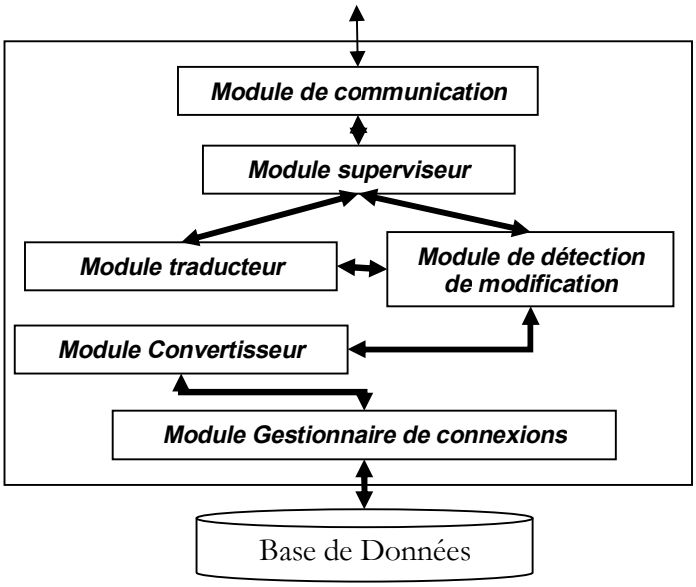


Figure 55 : Architecture de l'agent Moniteur

5.8.1 Les modules de l'agent Moniteur

Les modules de l'agent moniteur est similaire aux celui de l'agent ressource parce que tous les deux sont reliés à la même ressource, on va détailler les modules comme suit:

a) Module de communication

Il est, comme ses équivalents aux autres agents, responsable du traitement des messages entrant et de l'élaboration des messages sortant. C'est l'interface de l'agent avec le SMA.

b) Module superviseur

Le Module superviseur assure la supervision de l'agent Moniteur que celui de l'agent Ressource. Plusieurs rôles lui sont dévolus. Tout d'abord, il est là pour recevoir les messages en provenance du module de communication et il est chargé de faire suivre la-requête au bon module, ou bien de renvoie les schémas à l'agent intermédiaire lorsque les schémas des bases de données doivent être mis à jour.

Il construit ensuite une réponse sous forme de message ACL qu'il donne au module de communication pour être expédiée.

c) Module Gestionnaire de connexions

C'est le même module que celui de l'agent ressource, il permet la connexion avec la source d'information.

d) Module de détection de modification

N'importe quel changement sur la source d'informations doit être détecté par le module de modification. Ce module peut récupère les données à partir du module convertisseur, filtrer les données intéressants, puis demander au module traducteur de propager ces données à l'agent intermédiaire via le module superviseur.

Si la source d'informations fournit suffisamment de supports, il réduira beaucoup de temps pour développer ce module. Quand le module fonctionne avec une source d'informations non coopérative, il peut employer d'autres méthodes pour réaliser la détection de mise à jour.

Il existe trois approches générales de mise à jour:

- *Mise à jour immédiate*. La mise à jour qui s'est produite dans la source d'informations sera propagée immédiatement.
- *Mise à jour retardée*. La mise à jour qui est propagée quand une requête est arrivée.
- *Mise à jour périodique*. La mise à jour qui est propagée périodiquement.

Dans notre approche, nous combinons la mise à jour retardée et la mise à jour périodique pour mettre à jour l'uniformité de vues. Il signifie que nous détectons des mises à jour périodiquement ou quand une requête est envoyée à l'agent ressource. La période peut être ajustée par l'administrateur du système.

Pour des sources d'informations coopératives, telles que Oracle, nous pouvons employer des déclencheurs (triggers) pour détecter la modification dans la source. il enregistre ces informations dans une autre table, et recherche périodiquement des données de cette table. Ainsi, il peut utiliser ces informations pour informer l'agent intermédiaire.

Pour des sources d'informations non coopératives, telles que PostgreSQL ou XML, nous pouvons utiliser un algorithme de capture d'image instantanée de la source pour trouver les modifications. Nous dupliquons une copie de la table originale, et comparons périodiquement les deux tables. Ce dernier algorithme est sans aucun doute inefficace, mais dans certains cas on n'a pas un autre choix.

D'autre part, puisque le moniteur devrait envoyer la méta-données à l'agent intermédiaire, nous devons obtenir l'information à partir de la source d'informations automatiquement, ou de

l'administrateur de l'entrepôt de données. La méta-données contient le nom de la relation, de clés, d'attributs et type de données.

e) Module Convertisseur et le Module Traducteur

Ces deux modules servent non seulement l'agent ressource mais également l'agent moniteur, parce que les besoins des informations détectées et de la méta-données à renvoyer, devront aussi être transformés au modèle d'intégration du système de data warehouse.

5.8.2 Fonctionnement de l'agent Moniteur

Cet agent fonctionne en trois modes, qui peuvent être cités comme suit: Mise à jour périodique, l'interrogation de la source de données, et Mise à jour des schémas de bases de données.

- ***Détection de mise à jour périodiquement***

Dans notre approche, des mises à jour seront enregistrées jusqu'à ce que le module de détection de modification déclenche la période déterminée par l'administrateur de système de data warehousing. Chaque mise à jour sera enregistrée dans une table avant qu'elle soit envoyée.

À l'intervalle indiqué, l'agent moniteur envoie les mises à jour à l'agent intermédiaire et nettoie la table. Quand une autre mise à jour s'est effectuée sur la relation, la mise à jour sera enregistrée dans la table, et sera envoyée à l'intégrateur ultérieurement au temps prédéterminé.

- ***Lors de l'interrogation de la source de données,***

Quand l'agent ressource reçoit la sous-requête qui lui est destinée, il informe d'abord l'agent moniteur de détecter les mises à jour. Donc le module superviseur indique au module de détection de modification la collection des modifications de la source d'informations, et envoie les mises à jour au module traducteur, qui va traiter ces données en les rendant conformes au format du système d'entreposage, puis l'envoyer au module superviseur. Ce dernier module va ensuite traduire les résultats sous forme de message ACL, pour l'émission de mises à jour à l'agent intermédiaire, et envoyer une autre message à l'agent ressource en l'indiquant que la détection de mise à jour a été terminée et de continuer l'exécution de la sous-requête.

- ***Mise à jour des schémas de bases de données***

Celle-ci est effectuée lors d'une détection d'un changement dans les schémas de la source de données de façon automatique ou bien par l'administrateur du site, et aussi lors d'une demande de mise à jour à cause d'une requête type envoyée par l'agent intermédiaire à l'agent moniteur. C'est le module superviseur qui reçoit tous les messages via le module de communication, et les envoie au module détection de changements. Tous les messages concernant le schéma de la base de données

auquel cet agent est relié passent par ce dernier module, qui est chargé de récupérer le nouveau schéma (pour la mise à jour).

Le module de détection de changements envoie les mises à jour au module traducteur qui est chargé de traduire le schéma en XML. Le module superviseur va donc recevoir en retour le schéma XML de la base de données du module traducteur. Ce schéma contient notamment le nom de la base de données traitée ainsi que son emplacement, ajoutés par l'agent moniteur. Le schéma est envoyé à l'agent intermédiaire.

5.9. Agent Recherche d'Information (ARI)

L'agent de recherches d'information est chargé de l'interrogation des moteurs de recherche externes sur le Web (comme Google) afin d'obtenir des pages Web qui seront traitées par l'agent d'Extraction d'information, ainsi il est employé pour rechercher les pages HTML utiles, ou les documents utiles sur intranet.

Le **ARI** peut être travaillé comme un méta robot de recherche, recherchant des pages Web, en interrogeant des moteurs de recherche existants comme Google, Altavista ou d'autres. Ces pages sont ensuite transmises au l'agent d'extraction.

Il est responsable de l'acquisition des informations de diverses sources et il est en outre spécialisé dans la recherche autonome d'information disponible sur l'Internet sous différente forme (journal, avis de forum, états de la bourse etc...) au sujet du domaine choisi. Les ressources où il a recherché sont également identifiées automatiquement pour assurer un niveau plus élevé d'autonomie étant possible, ou bien l'administrateur doit l'orienter vers les associations ou les sites qui offrent à l'entreprise des rapports périodique sur l'état du marché, ou lorsqu'on a un abonnement le donnant à l'agent leur adresse et les clefs (par exemple username et password) pour lui permettre un accès.

D'autre part, cet agent est orienté vers la recherche des données utiles dans les documents locaux (intranet), y compris les spécifications, les contrats etc… sous différents formats (Doc, Pdf, Txt etc…).

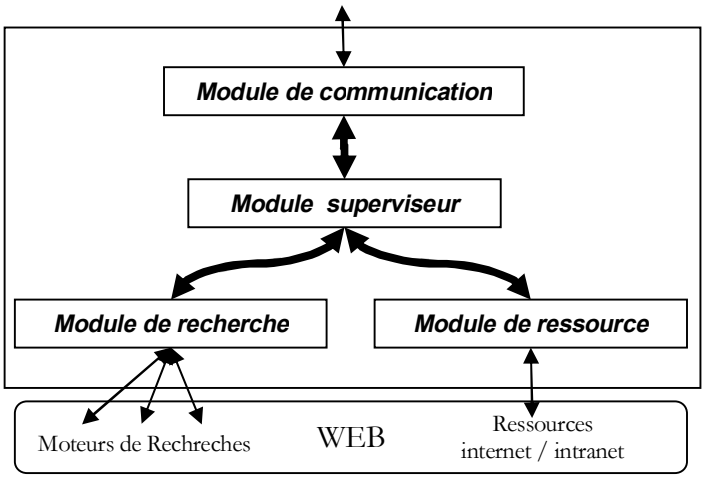


Figure 56 : Architecture de l'Agent de Recherche

5.9.1 Les modules de l'agent de recherche

a) Module de recherche

Le module de recherche transmet aux moteurs de recherche traditionnels (Google, Yahoo, Altavista, …) les requêtes qui lui proviennent de l'agent d'extraction. Après avoir fusionné les résultats obtenus par les différents moteurs sollicités, cet agent retourne les résultats à l'agent d'extraction concerné.

b) module de ressource

Le module de ressource est relativement similaire au module de recherche, cependant il transmet les requêtes issues du l'agent extraction vers des ressources spécialisées du Web ou bien sur l'intranet.

c) Module superviseur

Le Module superviseur assure la supervision de l'agent de recherche. Plusieurs rôles lui sont dévolus. Tout d'abord, il est là pour recevoir les requêtes en provenance du module de communication, et il va gérer l'allocation des requêtes au module de recherche et de ressources.

d) Module de Communication

Il se charge de l'envoi, la réception et la compréhension des messages reçus. Donc il est là pour recevoir les requêtes en provenance de l'agent extraction d'information, et offrir les résultats à celui-ci.

5.9.2 Fonctionnement de l'agent de recherche

L'agent de recherche reçoit par le module de communication les requêtes provenant de l'agent d'extraction d'information. Ensuite ces requêtes doivent être envoyées au module superviseur pour l'envoyer, soit:

- Au module de recherche pour permettre, via des moteurs de recherche, de récupérer des pages Web en HTML.
- Au module de ressources pour récupérer les résultats provenant des ressources spécialisées du Web ou bien de l'intranet.

Les résultats sont retournés vers le module superviseur qui s'occupe ensuite de les transmettre à l'agent d'extraction via le module de communication.

5.10. Agent Extraction d'Information (AEI)

L'agent d'Extraction d'information est spécialisé dans le traitement de pages Web sur un domaine spécifique. Un agent extraction demande une recherche pour des pages particulières à l'agent de recherches d'information. L'information extraite est transmise à l'agent de décision, afin d'être stockée dans une base de données spécifique, et qui sera accessible pour interrogation par les utilisateurs.

Le **AEI** génère tout d'abord les requêtes de pages Web envoyées à l'agent de recherche d'information, puis, et c'est sa tâche majeure, traite les résultats de ces requêtes provenant du **ARI**. Cette tâche est composée de plusieurs sous tâches: validation, classification fonctionnelle et extraction d'information.

Le rôle de l'agent d'extraction est d'utiliser des méthodes d'intelligence artificielle et des méthodes analytiques pour l'identification des faits importants et des configurations dans des données reçues à partir de l'agent de recherche d'information.

Les informations dérivées sont donc directement employées pour aider les utilisateurs dans leur prise de décision en les retournant à l'agent de décision.

L'agent d'Extraction d'information est employé pour extraire les informations dans les pages HTML; par le processus de transformation des résultats retournés par l'agent de recherche d'information à des sources de données XML.

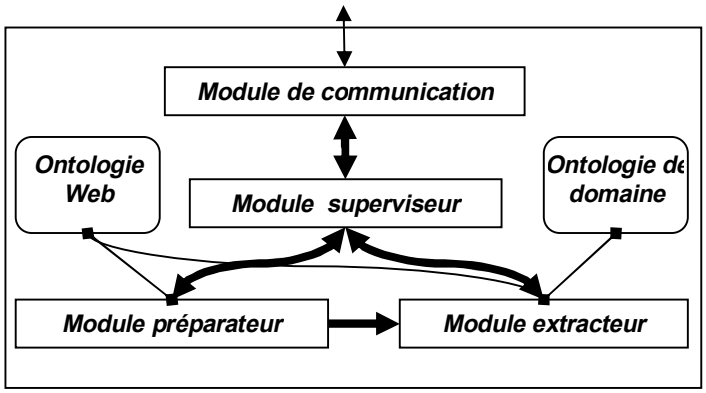


Figure 57 : Architecture de l'Agent extraction d'information

5.10.1 Les modules de l'agent d'extraction d'information

a) Module de communication

C'est comme celui des autres agents, c'est le point où l'agent communique avec SMA. Il accepte les messages qui contiennent les requêtes de l'agent de décision et les résultats de l'agent de Recherche, et envoie aussi les requêtes à l'agent de recherche et des pages traitées à l'agent de décision.

b) Module préparateur

Le module préparateur reçoit les pages Web en provenance ARI. Ce module réalise un premier traitement des pages Web reçues facilitant par la suite, le traitement d'extraction d'information. Ce traitement préliminaire consiste en la séquence des tâches suivantes :

- Une validation consistant à vérifier si les pages Web sont valides. C'est-à-dire si elles sont au format HTML et si elles sont accessibles. Les pages ne vérifiant pas ces conditions ne seront pas traitées par la suite.
- Un pre-processing identifiant le contenu, le titre, les liens et les emails disponibles dans la page.
- Une classification fonctionnelle. Cette tâche, à base de connaissances, utilise une ontologie définissant des concepts liés au Web. En utilisant le module préparateur, il y aura classification des pages Web selon un aspect fonctionnel. Les catégories fonctionnelles ainsi obtenues sont les suivantes : message, liste (de liens, de personnes, d'ouvrages), les pages choisies pour l'extraction et enfin les pages non valides.

c) Module extracteur

La tâche spécifique de ce module est d'effectuer une classification sémantique des pages reçues, ainsi qu'une extraction des informations qu'elles contiennent. Ce module est associé à un concept particulier de l'ontologie de domaine associé à l'agent d'extraction considéré.

Cet agent va utiliser les mots clefs pour la classification sémantique de ces pages et l'extraction des informations qui y sont continues. L'information extraite est ensuite transmise au module superviseur sous forme de document XML.

d) Module superviseur

Le Module superviseur de l'agent d'extraction possède des fonctionnalités similaires à celui de l'agent de recherche d'information. Il supervise l'activité des modules.

Il reçoit les résultats des requêtes envoyées par l'agent de recherche. Et les renvoie au module préparateur, et de même façon, il reçoit les informations extraites et les envoie à l'agent de décision via le module de communication.

En plus et indépendamment des autres tâches, le module superviseur a la possibilité d'interroger le Web en envoyant une requête à l'agent ARI.

5.10.2 Fonctionnement de l'agent d'extraction d'information

Le module préparateur reçoit du module superviseur les pages HTML brutes (sans traitement préalable), commence alors la tâche de validation, de pre-processing et de classification fonctionnelle de ces pages. Cette tâche produit des pages prêtes pour la phase d'extraction et de classification, et ces dernières sont envoyées au module extracteur.

Le Module extracteur reçoit les pages préparées et effectue la classification sémantique et l'extraction d'information des pages, ensuite, sont envoyées au module superviseur, qui va les envoyer à l'agent de décision via le module de communication.

Indépendamment des autres tâches, l'agent extraction d'information a la possibilité d'interroger le Web en envoyant une requête à l'agent ARI.

6. ACTEURS ET CAS D'UTILISATION DE NOTRE SYSTEME

La démarche préconisée pour l'utilisation d'UML est de commencer par analyser les fonctions que doit fournir le système. Cela revient à déterminer les cas d'utilisations du système. Pour chaque cas d'utilisation, correspond à une séquence d'objets mettant en œuvre des scénarios implémentant des cas d'utilisation. Un acteur représente un rôle joué par une personne ou une chose qui interagit avec le système.

Comme acteurs qui interagissent avec notre architecture nous avons :

1. L'utilisateur : demandeur de service du système de data warehousing qui accède à une interface via un agent de la couche utilisateur qui lui permet de générer ses requêtes à partir des éléments du schéma de l'entrepôt et de l'ontologie, et de récupérer ses résultats après les traitements effectués par le système.

2. Les administrateurs des sources : ce sont les gestionnaires des sources locales à intégrer, ils interviennent dans le système en cas d'ajout, suppression ou mise à jour de leurs sources de données, ils ont accès au système à travers une interface de mise à jour.

3. L'administrateur de l'entrepôt : c'est la personne responsable de la supervision du système, et de définition de schéma de l'entrepôt et celle des correspondances entre ce dernier schéma et l'ontologie.

Après avoir défini les acteurs qui interagissent avec notre architecture, nous allons déterminer pour chacun, ses cas d'utilisation. Le résultat de cette étape est le diagramme de cas d'utilisation représenté sur la figure (Fig.IV.5).

Acteur	Cas d'utilisation
Utilisateur	- Lancer la recherche pour l'analyse.
Administrateur de la BD	- Authentifier. - Ajouter un nouveau schéma d'une BD. - Modifier un schéma d'une BD.
Administrateur de l'entrepôt	- Authentifier. - Définit les schémas de l'entrepôt. - Lancer les agents. - Superviser les agents de l'architecture.

Tableau 5 : Les cas d'utilisations de l'architecture.

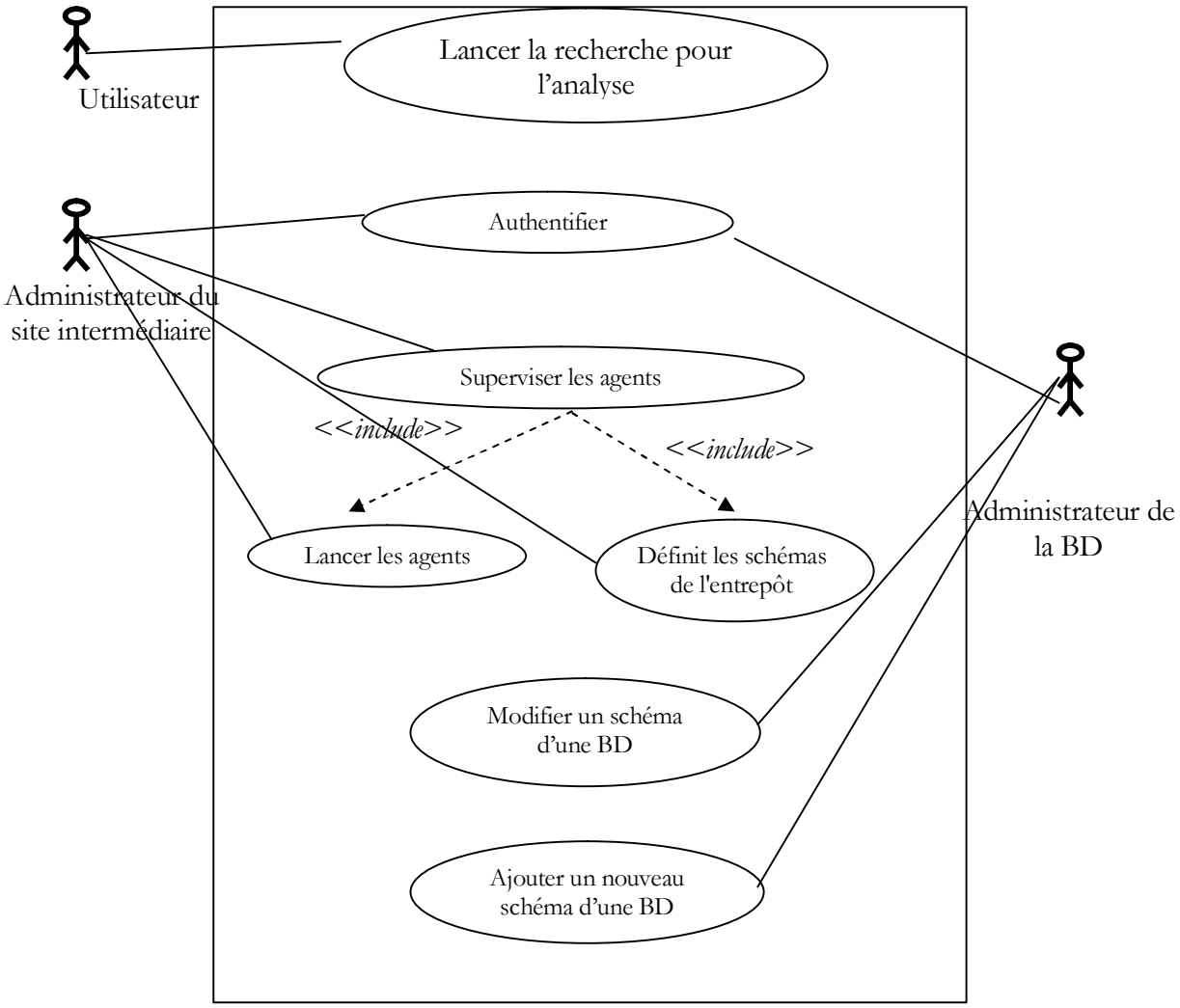


Figure 58 : Diagramme de cas d'utilisation de l'architecture

7. QUELQUES DIAGRAMMES DE FONCTIONNEMENT

Le fonctionnement de notre système s'articule autour de trois couches et dix agents distribués dans les couches qui se chargent de l'assistance dans le processus de l'entreposage de données par intégration de données complexes dans le data warehouse, tout en facilitant aussi l'accès au Data warehouse par les utilisateurs humains ou logiciels.

Nous présentons dans cette section quelques diagrammes de fonctionnement, réalisés à l'aide de la plateforme open source StarUML, selon le formalisme UML. Donc suivant les modes de fonctionnement nous avons clarifié celui-ci par deux types de diagramme: de collaboration et de séquencement.

7.1 Initialisation du système

Tout d'abord dans l'initialisation de notre système, l'administrateur du système par l'assistance de l'agent d'administration doit définir le schéma de l'entrepôt, et les administrateurs locaux des sites qui contiennent les sources d'informations doivent introduire les parties de ces bases de données autorisées dans le processus de data warehousing.

Lorsque le début de la construction est signalé, l'agent intermédiaire récupère de l'agent administrateur les correspondances entre le schéma multidimensionnel et l'ontologie. Des requêtes sont envoyées à l'agent superviseur pour les utiliser afin de récupérer, à partir des sources, les données nécessaires à la construction.

L'agent superviseur assure l'analyse, la décomposition, la réécriture et l'envoie des sous requêtes vers les agents ressource des sources de données locales,

Puis chaque agent ressource doit convertir et exécuter la requête sur sa source d'information, c'est donc une réponse au format XML que l'agent ressource la fait suivre à l'agent superviseur pour la recomposition du résultat final en vue d'être envoyé à l'agent intermédiaire.

L'agent intermédiaire calcule les données à stocker dans l'entrepôt. Pour cela, il récupère auprès l'agent superviseur les données nécessaires à la construction de l'entrepôt, qui sont les résultats des requêtes, et il les structure sous la forme de dimensions ou de cubes, et les stocke dans un cache multidimensionnel, qui est une collection de fichiers temporaires. Et en fin la traduction et le stockage des données se trouvent dans le cache en données représentées dans le modèle d'implantation de l'entrepôt.

Le diagramme de la **Figure 59** illustre le mécanisme d'initialisation de notre système, le diagramme de séquencement étant illustré dans la **Figure 60**.

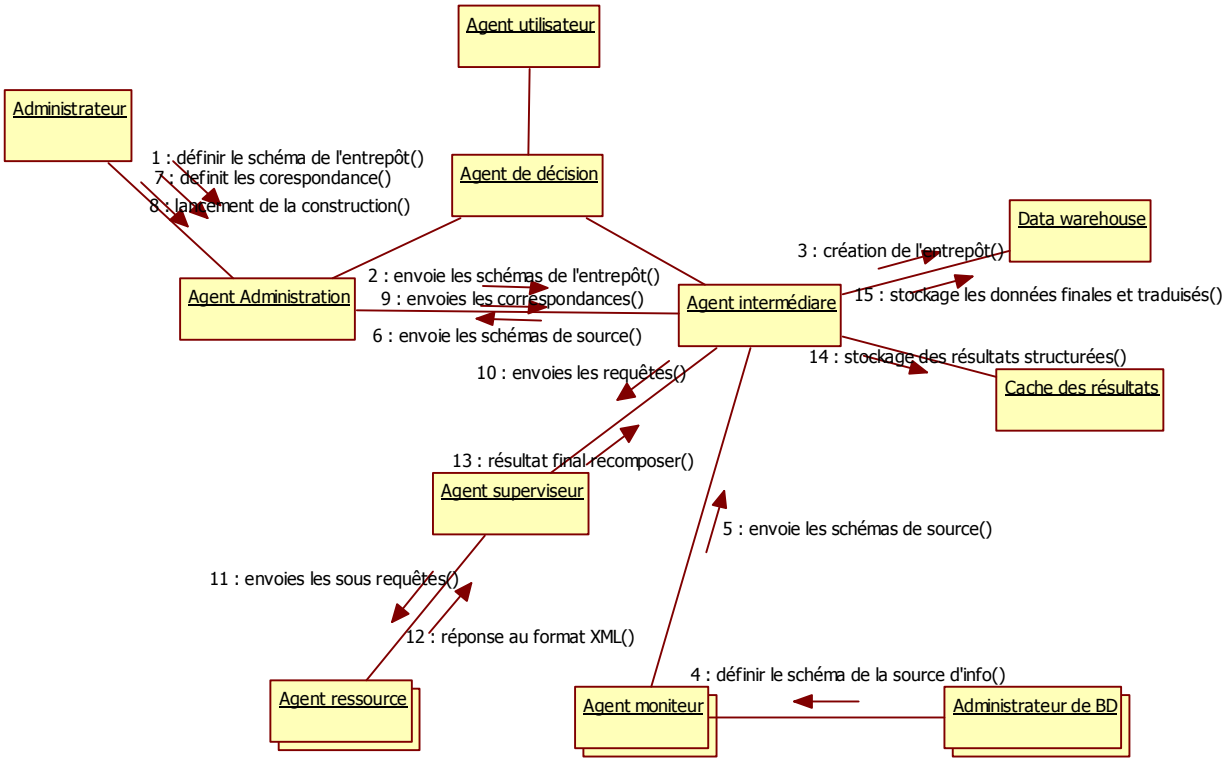


Figure 59 : Diagramme de Collaboration (UML) de l'initialisation

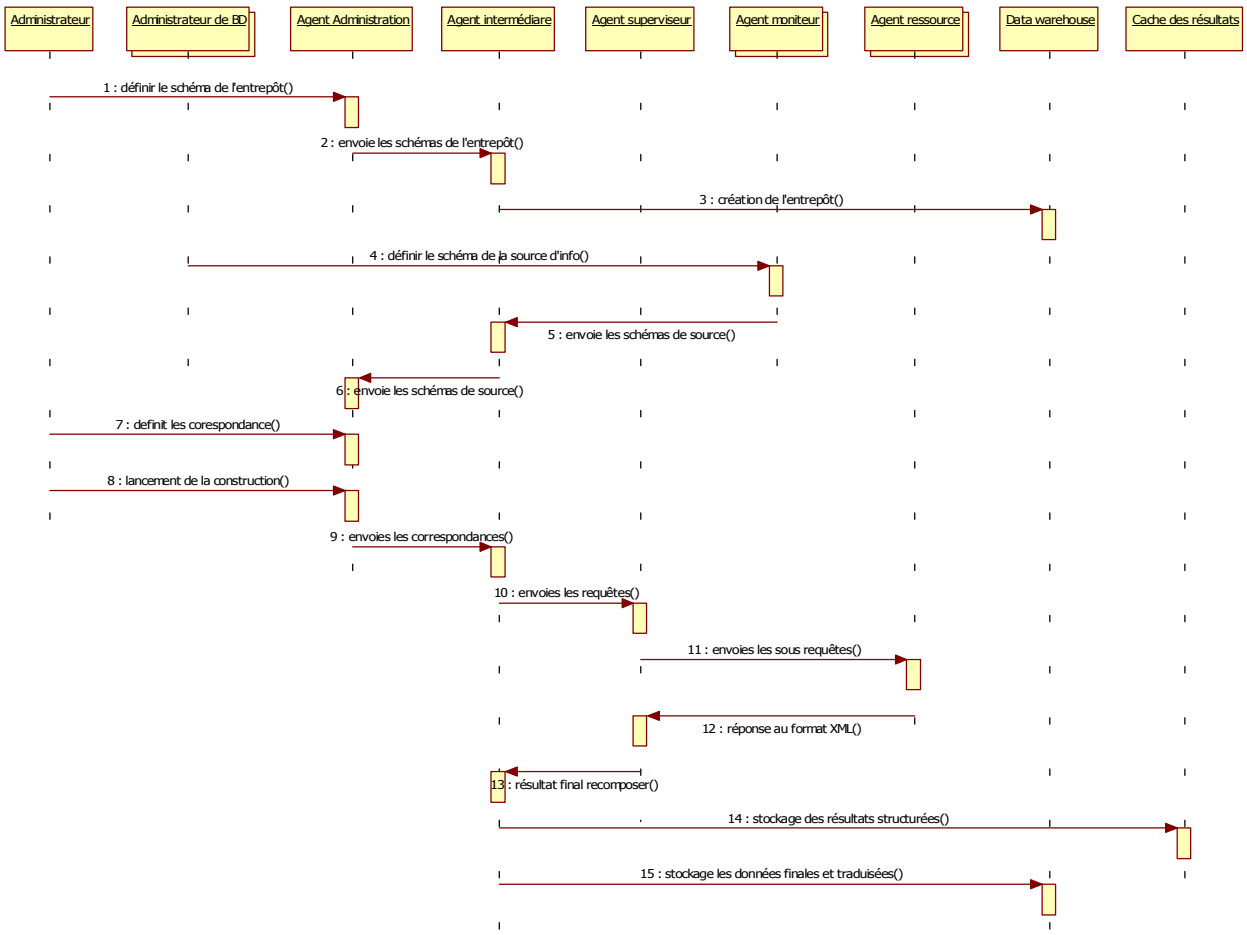


Figure 60 : Diagramme de Séquencement (UML) de l'initialisation

Ce schéma illustre de façon séquentielle le processus d'initialisation du système de data warehousing. Nous pouvons remarquer que dans notre exemple, cette opération n'est terminée que lorsque l'entrepôt est alimenté par la version initiale. Il faut préciser que ceci n'est qu'une vue du processus. En réalité, le système gère de multiples demandes et il prend aussi en compte le fait qu'une des bases interrogées peut ne pas répondre. De même, les problèmes de transmission des données ne sont pas illustrés ici et dans les modes de fonctionnement suivantes. Ces schémas n'ont ici qu'à titre indicatif, pour clarifier la vue des processus de fonctionnement.

7.2 Détection de mise à jour périodiquement

Dans notre approche, des mises à jour seront enregistrées, dans une table, jusqu'à une période prédéfinit par l'administrateur de l'entrepôt de données. C'est l'agent moniteur qui envoie les mises à jour à l'agent intermédiaire pour faire des calculs afin d'indiquer quelles modifications de sources ont besoin d'être propagées dans les cubes définissant l'entrepôt et quelles mesures affectées par ces modifications et ensuite générer les requêtes pour collecter les informations nécessaires au calcul de cette mesure,

Comme dans le mode de fonctionnement précédant, l'agent intermédiaire envoie ces requêtes à l'agent superviseur pour les décomposer aux différents agents ressources pour la collection des données demandées dans les requêtes, et donc retourner les résultats à l'agent intermédiaire, puis ce dernier stocke les résultats dans le cache afin de traduire les données trouvées dans le cache en données représentées dans le modèle d'implantation de l'entrepôt.

Le diagramme de la **Figure 61** illustre le mécanisme de détection de mise à jour périodiquement, le diagramme de séquencement étant illustré dans la **figure 62**.

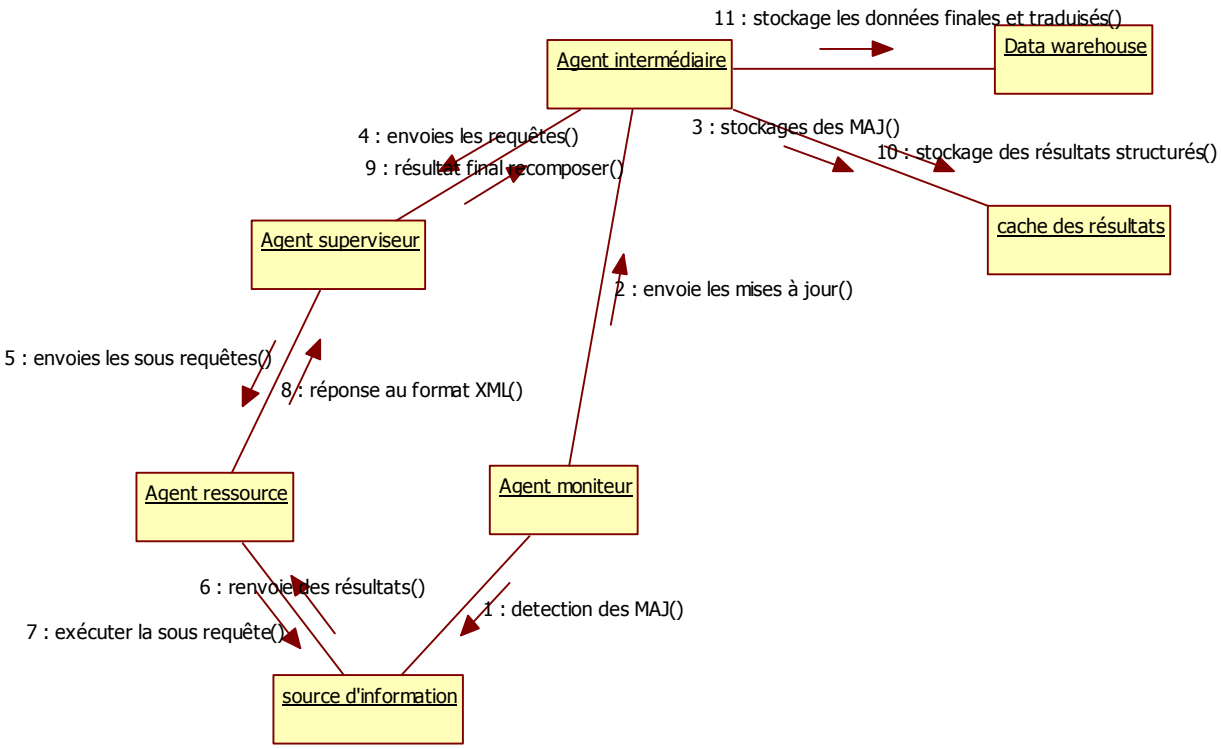


Figure 61 : Diagramme de Collaboration (UML) de détection de mise à jour périodiquement

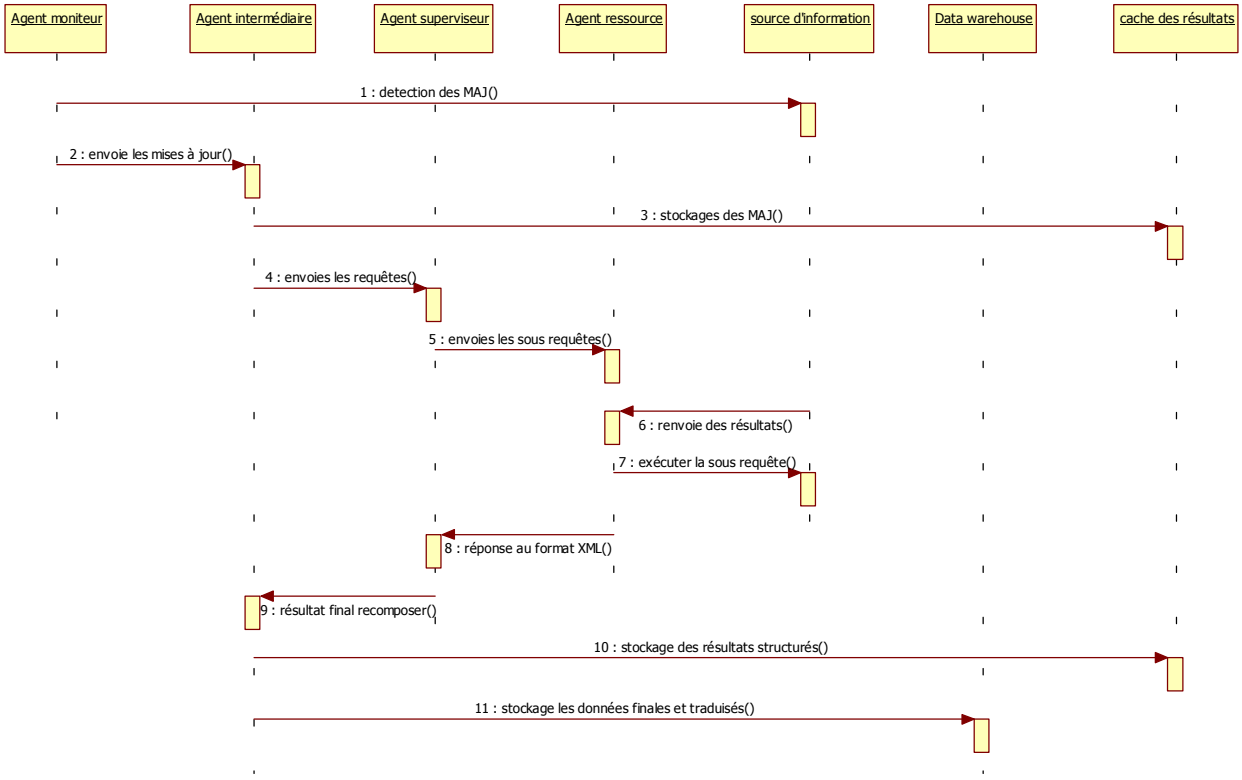


Figure 62 : Diagramme de Séquencement (UML) de détection de mise à jour périodiquement

7.3 Interrogation de l'entrepôt

Premièrement L'agent utilisateur permet la création conviviale des requêtes d'analyse, et ça c'est leur rôle principal, qui va ensuite construire un message et le transmettre à l'agent de décision.

D'autre part ces requêtes peuvent provenir des agents d'application. Ces agents doivent agentifier un système logiciel externe, et l'intègrer au processus de data warehousing. Donc les agents d'application permettent de détecter les désirs des applications par la formulation des requêtes. Puis il va envoyer ces requêtes à l'agent de décision et attend les réponses.

L'agent de décision reçoit les requêtes de la couche utilisateur qui va ensuite les traiter et les envoyer à l'agent intermédiaire, et aussi il va formuler une demande de recherche d'information externe en termes des concepts composants la requête, puis il l'envoie vers l'agent d'extraction d'information.

Donc maintenant l'agent intermédiaire permet de fournir une interface pour l'interrogation de l'entrepôt par l'agent de décision, pour cela il accepte les requêtes et les exécuter par le système de gestion de l'entrepôt de données, puis il collecte les résultats et les envoie vers l'agent de décision.

Parallèlement, l'agent d'extraction d'information a la possibilité d'interroger le Web suite aux demandes de l'agent de décision en envoyant une requête à l'agent **ARI**.

L'agent de recherche reçoit les requêtes. Ensuite ces requêtes doivent permettre, via des moteurs de recherche, de récupérer des pages Web en HTML, ou bien récupérer les résultats provenant des ressources spécialisées du Web ou bien sur l'intranet. Les résultats sont retournés vers l'agent d'extraction.

L'agent d'extraction d'information reçoit les pages HTML brutes, puis il commence alors la tâche de validation, et de classification fonctionnelle de ces pages pour la *classification sémantique* et *l'extraction d'information* des pages, ensuite envoyées à l'agent de décision.

Quand l'agent de décision recueille les réponses soit de l'agent intermédiaire ou bien de l'agent d'extraction d'information, il doit les rédiger au format XML pour les envoyer aux agents qui demandent ces informations.

Quand une réponse reçue par l'agent d'application, cette réponse doit être acheminée vers le système logiciel après les traductions nécessaires pour homogénéiser les données avec le modèle de données accepté par le système logiciel.

L'agent utilisateur permet aussi l'affichage des résultats obtenus suivant le profil utilisateurs et ses préférences du format d'affichage des résultats et sur le contenue qui l'intéresse tel que de la requête proprement dit et les résultats de recherche d'information externe.

Le diagramme de la **Figure 63** illustre le mécanisme de L'interrogation de l'entrepôt, le diagramme de séquencement étant illustré dans la **figure 64**.

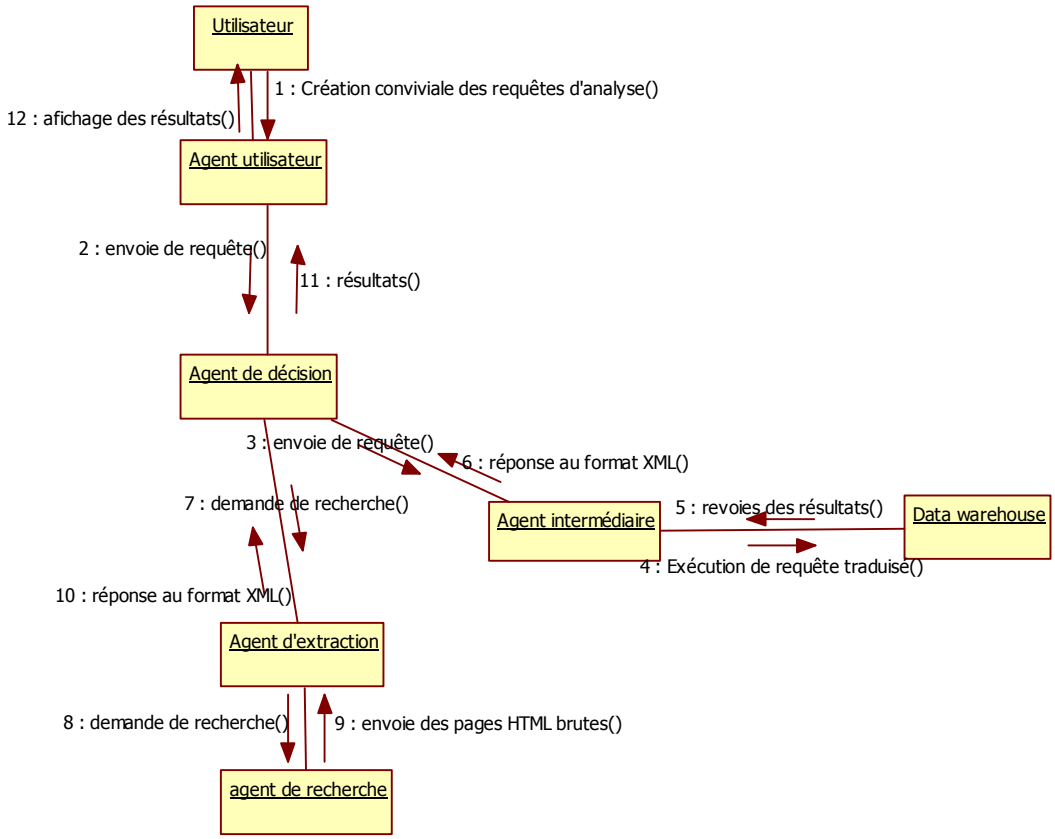


Figure 63 : Diagramme de Collaboration (UML) d'Interrogation de l'entrepôt

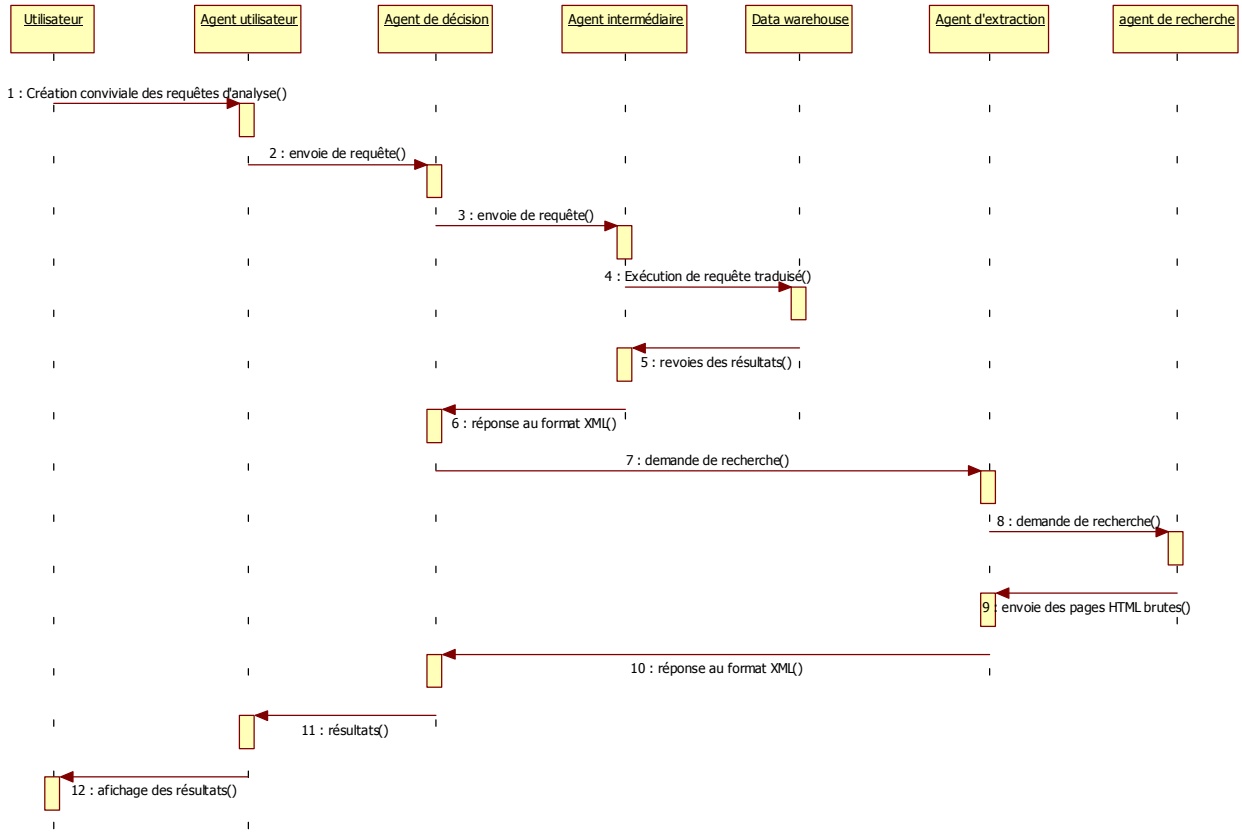


Figure 64 : Diagramme de Séquencement (UML) d'Interrogation de l'entrepôt

8. CONCLUSION

Après avoir présenté les différents aspects théoriques de Data Warehousing et d'Agents à réaliser dans la première partie, nous avons consacré cette deuxième partie (La conception) aux aspects pratiques de ce mémoire. Tout au long de cette partie nous avons présenté et conçu les différents composants d'une architecture destinée à l'entreposage de données reposant sur l'interaction et la coopération d'un ensemble d'agents.

Dans le chapitre suivant nous allons présente l'implémentation de notre architecture.

CHAPITRE V 5

Implémentation

CHAPITRE V: IMPLÉMENTATION

1. INTRODUCTION

Après avoir présenté l'étude conceptuelle de notre approche, nous présenterons dans cette partie la mise en œuvre du prototype conçu pour valider la solution.

Nous essayerons d'élaborer un prototype de système de data warehousing à base d'agent capable de supporter de manière quasi-automatique l'évolutivité du nombre de sources de données tout en mettant en œuvre les procédures nécessaires pour la traduction d'une requête globale sur le système.

L'objectif de ce présent chapitre est de mettre en œuvre notre architecture sur des bases de données réparties. Tout d'abord, nous commençons par une brève présentation de l'environnement de développement, puis nous présentons les interfaces de notre application.

2. EXPERIMENTATION

Le prototype que nous avons mis en œuvre pour valider notre conception comporte les composants suivants :

2.1 L'ontologie

Notre prototype inclut une ontologie de domaine OWL qui représentera le schéma global et sera donc le support de la génération des requêtes globales de l'utilisateur, une ontologie OWL sera exploitée à travers ses concepts, rôle et propriétés. Dans notre travail nous avons utilisé une ontologie locale qui a été conçue avec l'utilitaire PROTéGé 2000. Cette ontologie sera stockée au niveau de site intermédiaire.

2.2 Base de description des schémas

Notre base de description des schémas est une collection de documents XML qui contient les différentes caractéristiques de chaque élément des sources à intégrer ainsi que les différentes correspondances avec les propriétés et rôles de l'ontologie, elle est également stockée au niveau de site intermédiaire. La description de cette base de données a été donnée dans la partie conception.

2.3 Le Data Warehouse

L'objectif de nous expériences est de montrer l'utilisation de notre prototype pour construire et faire évoluer un entrepôt de données. Nous avons choisi une application typique: les ventes des produits dans une chaîne de magasins.

La figure 65 montre le schéma multidimensionnel de notre application. Il est formé par les schémas des dimensions Produit, Magasin et Temps, et par le schéma du cube Ventes:

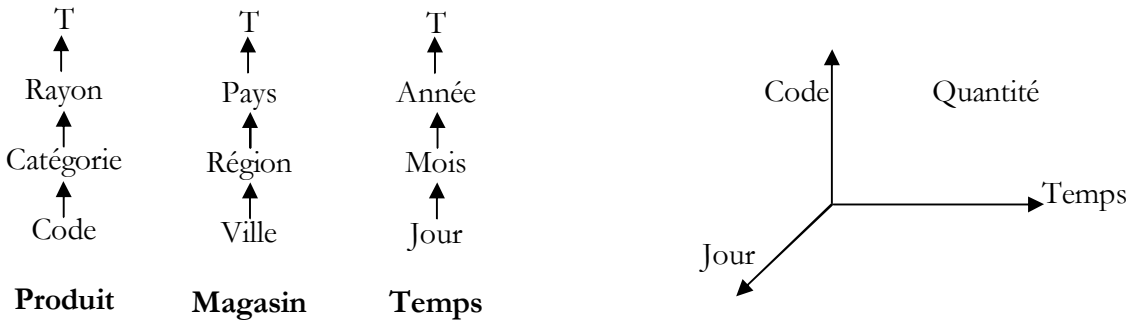


Figure 65 : schéma multidimensionnel de l'application exemple

La figure 66 montre les schémas des dimensions et de cube représentés sous format XML.

```
<dimension>
    <name> Magasin </>
    <level lid='1' type='char(40)'> ville </>
    <level lid='2' type='char(40)'> region </>
    <level lid='3' type='char(40)'> pays <I>
    <level lid='4' type='none'> Ail </>
    <rollup origin='1' target='2' />
    <rollup origin='2' target='3' />
    <rollup origin='3' target='4' />
</>
```

```
<dimension>
    <name> temps </>
    <level lid='1' type='char(8)'> jour </>
    <level lid='2' type='char(10)'> mois </>
    <level lid='3' type='char(4)'> année <I>
    <level lid='4' type='none'> Ail </>
    <rollup origin='1' target='2' />
    <rollup origin='2' target='3' />
    <rollup origin='3' target='4' />
</>
```

```
<dimension>
    <name> Produit  </>
    <level lid='1' type='char(10)'>  code  </>
     <level lid='2' type='char(40)'>  categorie  </>
    <level lid='3' type='char(40)'>  rayon  </>
    <level lid='4' type='none'> Ail </>
    <rollup origin='1' target='2' />
    <rollup origin='2' target='3' />
    <rollup origin='3' target='4' />
</>
```

```
<cube>
    <name> ventes </>
    <axis> code </>
    <axis> ville </>
    <axis> jour </>
    <mesure type='integer'> quantité </>
</cube>
```

Figure 66 : Représentation XML du schéma multidimensionnel

2.4 Les sources de données

Notre entrepôt est construit à partir du schéma XML montré par la figure 66. Rappelons que cette vue intégrée par cinq sources.

- **La source catalogue :** fournit des informations sur les produits: code, nom, catégorie, rayon. Voici le schéma XML de cette source :

```
<ELEMENT catalogue (produit*)>
<ELEMENT produit (pid, nom, cat, rayon)>
<ELEMENT pid (#PCDATA)>
<ELEMENT nom (#PCDATA)>
<ELEMENT cat (#PCDATA)>
<ELEMENT rayon (#PCDATA)>
```

- **La source magasins** : fournit des informations sur chacun des magasins, telles que son numéro d'identification, son adresse et son numéro de téléphone.

```
<ELEMENT magasins (magasin*)>
<ELEMENT magasin (mid, adresse, tel, ville)>
<ELEMENT mid (#PCDATA)>
<ELEMENT adresse (#PCDATA)>
<ELEMENT tel (#PCDATA)>
<ELEMENT ville (#PCDATA)>
```

- **La source *geog*** : fournit des données géographiques.

```
<ELEMENT geog (pays*)>
<ELEMENT pays (nomp, region*)>
<ELEMENT region (nomr, ville*)>
<ELEMENT nomp (#PCDATA)>
<ELEMENT nomr (#PCDATA)>
<ELEMENT ville (#PCDATA)>
```

- Une source décrit les ventes journalières réalisées par chacun des magasins localisés à Biskra, Batna et Constantine.

```
<ELEMENT ventes (vente*)>
<ELEMENT vente(pid,unites, jour)>
<ELEMENT pid (#PCDATA)>
<ELEMENT unites
                (#PCDATA)>
<ELEMENT jour(#PCDATA)>
```

Biskra

```
<ELEMENT ventes (vente*)>
<ELEMENT vente (code, Quantité,
 date)>
<ELEMENTcode (#PCDATA)>
<ELEMENT Quantité
                (#PCDATA)>
<ELEMENTdate (#PCDATA)>
```

Batna

```
<ELEMENT ventes (vente*)>
<ELEMENT vente (pid, QT,
 date)>
<ELEMENT pid (#PCDATA)>
<ELEMENT QT
                (#PCDATA)>
<ELEMEN date (#PCDATA)>
```

Constantine

3. ENVIRONNEMENT DE DÉVELOPPEMENT

Nous avons développé notre prototype sous le système d'exploitation Windows XP, avec des outils nécessaires à sa mise en œuvre :

3.1 La plateforme JADE

JADE « Java Agent Development Environment » est une bibliothèque développée en Java, Elle autorise l'utilisation de n'importe quelle plate-forme et des outils associés, en particulier le fonctionnement dans un environnement ouvert et dynamique, fournit une classe de base Agent qui traite toutes les tâches liées aux fonctionnement des agents, telles que l'enregistrement, la configuration, la gestion à distance, etc., et un ensemble de méthodes qui peuvent être appelées pour implémenter les tâches spécifiques à l'agent, comme l'envoi des messages, utilisation des protocoles d'interaction standards, etc., et gérant tous les aspects de contrôle des comportements. JADE propose également des classes de bases pour des types de comportements spécifiques (CyclicBehaviour, ParallelBehaviour, ReceiverBehaviour, SenderBehaviour, SequentialBehaviour...).

L'objectif de JADE est de simplifier le développement de systèmes multiagents en restant conforme aux normes par un jeu complet de services et d'agents systèmes respectant les spécifications de la FIPA : services « pages-blanches » et « pages-jaunes », service de transport et d'analyse syntaxique des messages, et une bibliothèque de protocoles d'interaction FIPA.

La plateforme multi-agents JADE a été choisie pour sa conformité aux normes de la FIPA et pour l'ensemble d'outils inclus facilitant les différentes phases de développement de systèmes multi-agents, il fournit ainsi une infrastructure agent, gérant la communication entre agents et la gestion de leurs cycles de vie, et sert de support pour le développement d'agent au comportement simple.

3.2. Langage de programmation

Nous avons utilisé l'environnement de programmation JBuilder9 de Borland basé sur le langage Java. Ce langage donne un fort potentiel pour la mobilité des codes, il est basé sur l'approche objet, il intègre des paquetages qui facilitent son utilisation et rendent le code plus lisible et facile à comprendre. Le JBuilder9 est un environnement très puissant et très évolué, il facilite la conception des applications et rend la programmation non fastidieuse.

3.3. L'API JENA

L'accès aux éléments de l'ontologie est effectué avec l'utilisation de l'API (Application Programming Interface) JENA. Cette librairie JAVA offre un ensemble de méthodes pour la manipulation des ontologies. Cette API est issue d'un projet open-source de « HP Labs Semantic web program ».

4. PRESENTATION ET TEST DE L'ARCHITECTURE PROPOSEE

L'architecture que nous avons conçue est un outil pour l'entreposage de données. Elle utilise la technologie des agents, par conséquent il faut installer sur chacun des postes le JDK (Java Developer Kit) fourni par Sun Microsystem (java.sun.com) et les classes JADE. En effet, ces classes permettent de gérer les agents.

Notre application est composée de trois interfaces principales où chacune représente une couche de l'architecture proposée :

4.1. Interface pour la couche utilisateur

4.1.1. Le site client

Cette interface permet à l'utilisateur d'introduire sa requête. Elle est constituée de deux éléments à savoir un menu principal et un espace de travail, comme le montre la figure 67.

Figure 67 : Interface pour la couche utilisateur

4.1.2. Lancement d'une recherche

a. Concept central

Dans cette étape, l'interface de la figure 67 permet à l'utilisateur d'introduire sa requête. Avant de lancer la recherche, l'utilisateur doit choisir l'un des sujets proposé par le data warehouse. Dans notre exemple on qu'un seul sujet possibles le « Ventes ». Il correspond à une table de faits dans la modélisation dimensionnelle conceptuelle présenté à la section précédente.

b. Informations à afficher

Les informations à afficher correspondent à une projection : tous les éléments sélectionnés apparaîtront dans l'extraction finale. On peut remarquer que les informations à afficher peuvent provenir autant de la table de faits que des tables dimensions. Si l'on transpose en langage SQL, cet écran permet la constitution de la clause « SELECT ».

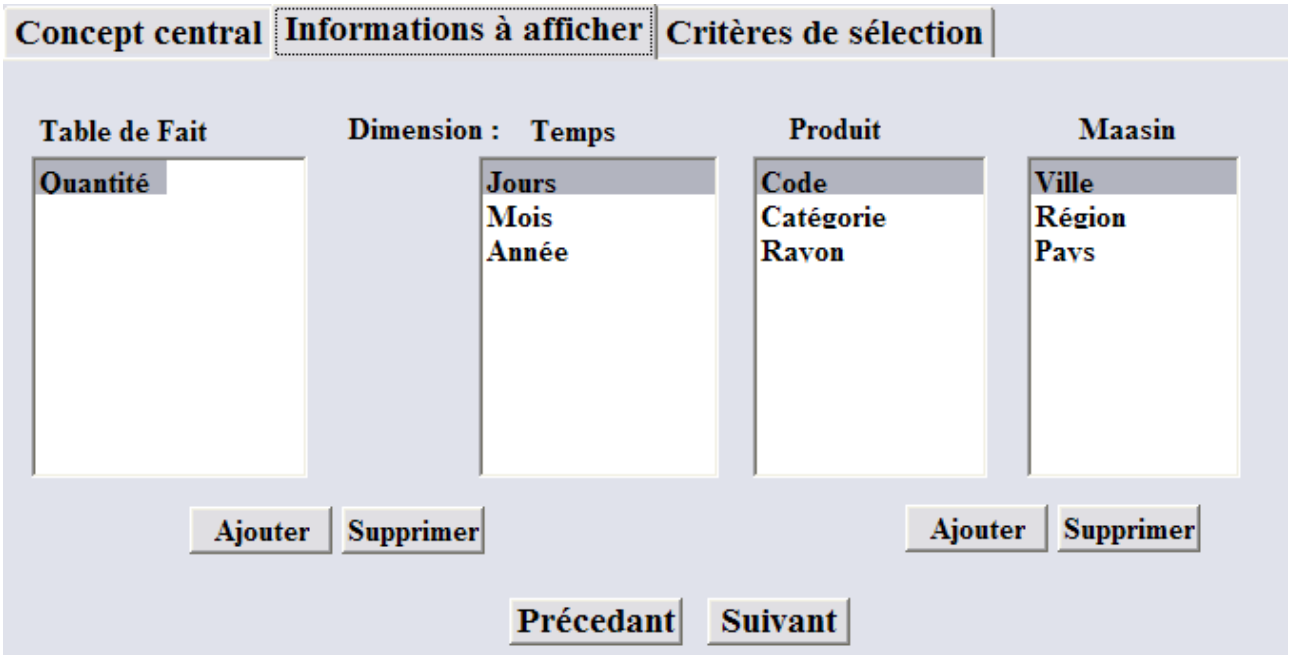


Figure 68: Interface pour la Sélection des information à afficher

c. Critères de sélection

Cet écran présente les critères de sélection que l'utilisateur peut appliquer. Ces critères correspondent en SQL à la clause « WHERE ». Dans ce prototype, toutes les clauses sont supposées liées entre elles par l'opérateur « AND » pour éviter toute complication inutile.

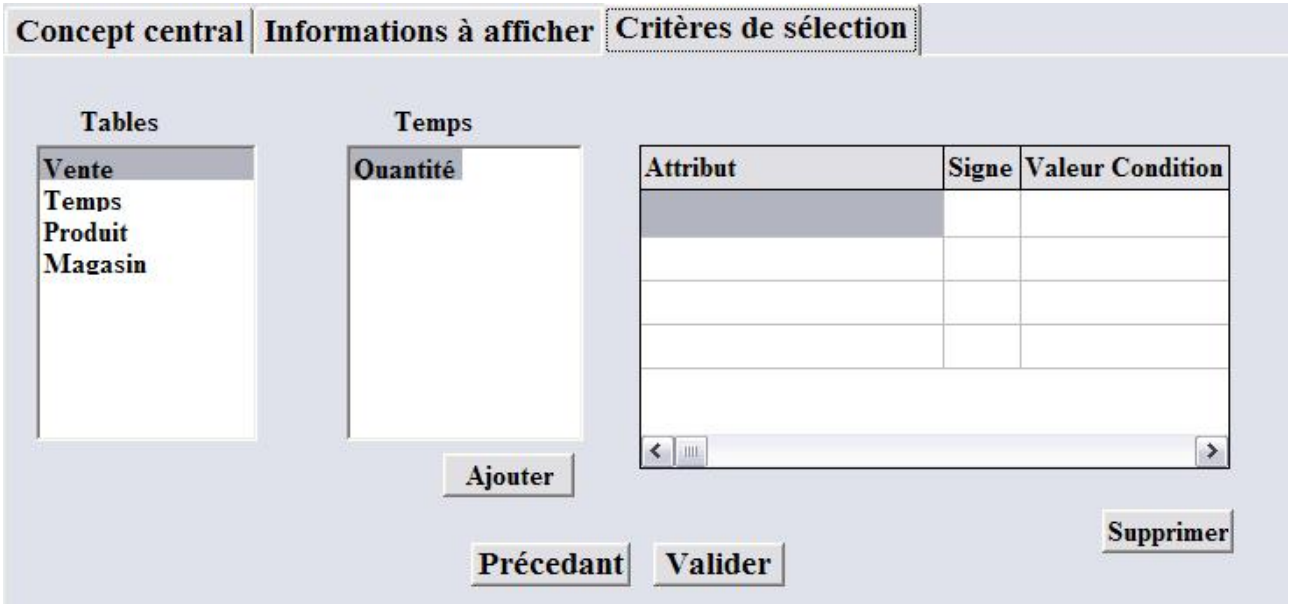


Figure 69: Interface pour la Sélection des Critères de sélection

4.2. Interface pour la couche ressource de données

4.2.1. Le site ressources de données

Le site ressources de données est une interface facilitant à l'administrateur la création et la modification d'un schéma d'une base de données, ceci fait de lui une ressource critique pour le maintien de la cohérence entre la base et son schéma. L'accès au site de ressources est conditionné par une authentification intègre.

Figure 70 : Interface pour la couche ressources de données

4.2.2 Intégration d'une nouvelle source

- La première fenêtre de cette interface (Figure 70) offre la possibilité à l'utilisateur d'introduire les paramètres de connexion de sa source (pilote, Chemin, utilisateur, mot de passe) et de tester cette connexion. A la validation des paramètres, le système vérifie la connexion à la base de données et le passage à la prochaine étape ne se fera que dans le cas de la réussite de cette connexion.
- Après la validation de la connexion à la base de données, on doit offrir à l'administrateur la possibilité d'introduire les différentes informations constituant le schéma de la base (liste des tables et liste des attributs). La liste des tables s'affiche pour qu'il sélectionne celles où la recherche est autorisée. Pour chaque table sélectionnée, seuls les attributs qu'il indique auront le droit d'être affichés.

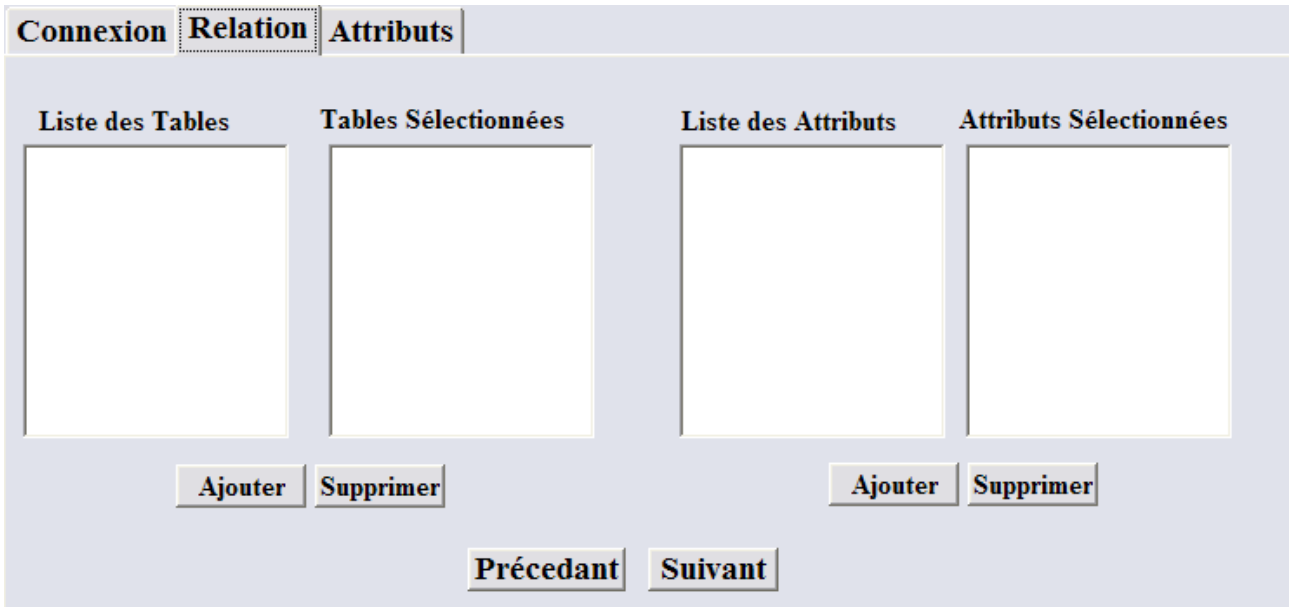


Figure 71. Saisie des tables et attributs de la nouvelle source.

- La dernière étape permet à l'utilisateur de remplir les caractéristiques des attributs des tables qu'il souhaite ajouter, ainsi que les concepts et propriétés de l'ontologie auxquels ils appartiennent. La validation de cette étape permet la mise à disposition effective de la nouvelle source dans le système.

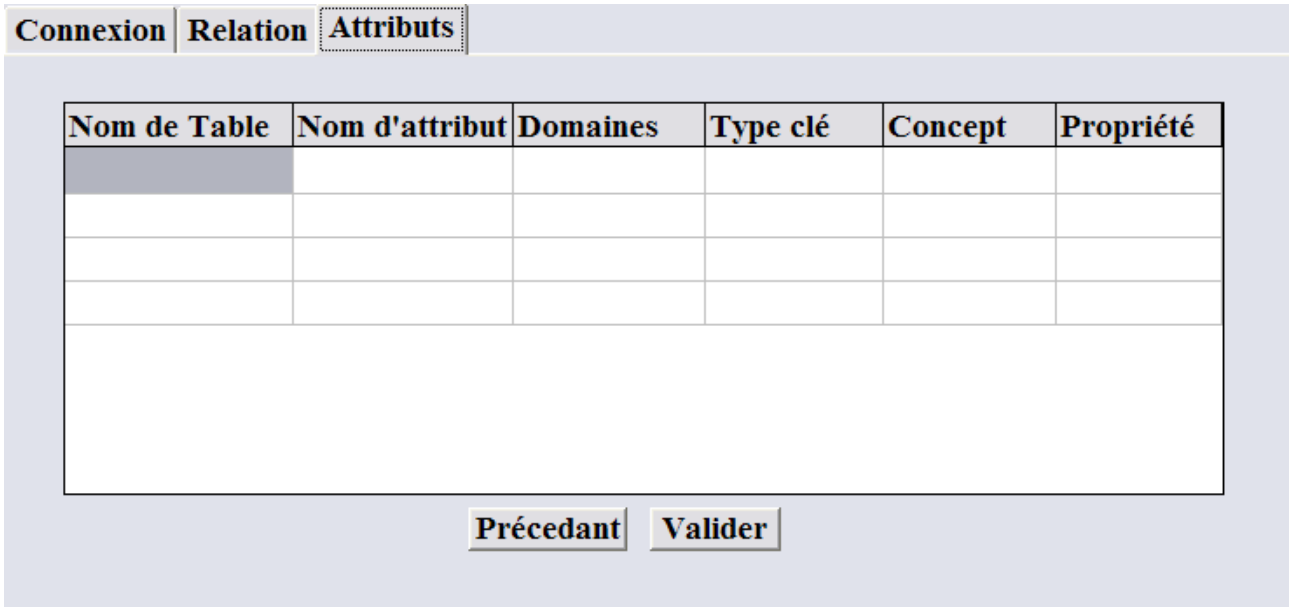


Figure 72 : Mise en correspondance des attributs avec les propriétés de l'ontologie.

4.3.. Interface pour la couche intermédiaire

Constituée essentiellement de deux éléments (Un menu principal et une barre d'outils), elle permet à l'administrateur d'autoriser la recherche d'informations en créant les agents spécifiques et appropriés.

Figure 73 : Interface pour la couche intermédiaire

5. Conclusion

Au cours de cette dernière étape de notre travail nous avons réalisé un prototype de système d'entreposage de données, nous avons présenté son architecture logicielle et ses différents composants et les différents outils utilisés pour son développement. Nous avons implémenté l'approche qui permet un accès au système en développant notre application en trois tiers. Nous avons aussi tenu compte de l'évolutivité du nombre de sources de données à intégrer en permettant un ajout automatique d'une nouvelle source de données.

Conclusion et Perspectives

Conclusion générale

Toutes les entreprises disposent de données, elles proviennent soit de leurs systèmes opérationnels, soit de l'extérieur. Ces entreprises doivent posséder des informations nécessaires pour prendre de bonnes décisions. Pour cela, il est fondamental de mettre en place une nouvelle informatique décisionnelle pour obtenir une meilleure compréhension des informations disponibles. Le but espéré par ces systèmes est de perfectionner le processus de prise de décision.

Le travail, qui nous a été confié, était de contribuer à la conception et au déploiement d'un système de data warehousing, pour garantir une certaine efficacité dans le traitement de ce très grand nombre d'informations, et d'optimiser les différents calculs nécessaires pour accomplir l'ensemble des tâches de data warehousing. Un système multi-agents, avec des agents qui s'exécutent indépendamment les uns des autres, pourra assurer parfaitement ce rôle.

Notre travail consistait alors à proposer une architecture de data warehousing basée sur l'interaction/coopération des agents, cette architecture s'articule autour de trois couches (couche utilisateur, couche intermédiaire et couche ressource) et dix agents distribuées dans ces couches qui se chargent de l'assistance dans le processus de l'entreposage de données par l'intégration de données complexes dans le data warehouse, tout en facilitant aussi l'accès à cette base par les utilisateurs humains ou logiciels.

Cette architecture est dédiée plus particulièrement au data warehousing, pour sa mise en œuvre, nous avons opté pour un environnement fondé sur une architecture modulaire d'entreposage de données proposée par le projet WHIPS. Elle ensuite fusionnée avec l'architecture du système multi-agents NetSA qui est destinée aux environnements riches en informations hétérogènes comme celle de notre ces.

Cette combinaison entre la technologie des systèmes multi-agents et celle de data warehousing a montré une vision qui parait aujourd'hui comme une solution séduisante pour parvenir à maîtriser la distribution des ressources de données et la complexité de mise en œuvre.

Perspectives

Parmi les perspectives qui restent à explorer, nous pouvons citer en particulier les listes de développement futur suivant :

- L'ajout de la caractéristique de mobilité aux agents de notre architecture tel que, les agents de la couche utilisateur qui peuvent se transporter à la couche intermédiaire où trouve les agents stationnaires, et aussi les agents moniteur qui peuvent aller à la couche intermédiaire quand une modification de schéma de la source
- Développement d'un Agent de Data mining pour la bonne exploitation des données du Data Warehouse.
- L'application de l'architecture sur un cas d'entreprise réel, pour vérifier sa validité.

BIBLIOGRAPHIES

[1]. Inmon W.Bill "Building the Data Warehouse, Fourth Edition", Wiley Publishing., Indianapolis, Indiana, 2005

[2]. Intelligence économique et décision, http5://www.urfist.cict.fr/pub9.html1

[3]. Besson B., Possin J.-C. « Du renseignement à l'intelligence économique Détecter les menaces et les opportunités pour l'entreprise », Dunod, 1996.

[4]. Martre H., Levet J.-L., et al. Rapport dit «Martre» : Intelligence économique et stratégie des entreprises, Rapport du Commissariat Général au Plan, Paris, La Documentation Française, 1994.

[5]. Revelli C., « Intelligence stratégique sur Internet, Comment développer des activités de veille et d'intelligence économique sur le web », Éditions Dunod, Paris, 2000.

[6]. Peguiron F., « Application de l'Intelligence Economique dans un Système d'Information Stratégique universitaire : les apports de la modélisation des acteurs », Doctorat de l'Université Nancy 2, 2006.

[7]. Amos D., « L'Intelligence Économique et les Systèmes d'Informations : Problématiques et approches de solutions », Colloque sur la veille stratégique en entreprise, Algérie Télécom, Équipe SITE-LORIA, 2005.

[8]. Nakache D., « Data warehouse et data mining », Conservatoire National des arts et métiers de Lille, 1998.

[9]. Paulraj P., « Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals », John Wiley & Sons, 2001.

[10]. MARHOUMI F.-E, « Entrepôts de données XML : Développement d'un outil Extraction Transformation Load (ETL) » Mémoire de fin d'études d'Ingénieur Civil, Université Libre de Bruxelles, 2006.

[11]. Le Grand Dictionnaire terminologique de l'Office québécois de la langue française, http://www.granddictionnaire.com.

[12]. Chaudhuri S., « Data Warehouse and OLAP for decision support », Microsoft research, Redmond: SIGMOD Record, USA, 1997.

[13]. Georges G., « Internet/ intranet et bases de données (data web, data media, data warehouse, data mining) », Eyrolles, 2000.

[14]. Chuck B., Dirk H., Don Schau, et al. « Data Modeling Techniques for Data Warehousing », IBM International Technical Support Organization, 1998.

[15]. Bolognini N.-R, « Etude pour la création d'un entrepôt de données dans le cadre de l'assurance vie et transformation des données en informations utiles en vue d'une prise de décision », Mémoire de Diplôme postgrade, Université de Lausanne, 2002.

[16]. Olivier T., « Modélisation et manipulation d'entrepôts de données complexes et historisées », thèse de Doctorat de l'universite paul sabatier, 2000.

[17]. Irtishad A., Salman A., « Development of a decision support system using data warehousing to assist builders/developers in site selection », Automation in Construction 13, Elsevier, 2004.

[18]. Jenifer W., « Lettre from the special issue editor », Bulletin of the technical committee on Data Engineering,Vol. 18 No. 2, IEEE Computer Society, 1995.

[19]. Elvire S., « Le Système d'Information Décisionnel ou Comment piloter l'entreprise grâce au data warehouse », Thèse Professionnelle Mastère MSIT HEC, Ecole des Mines, 2004.

[20]. Kimball R., « The data warehouse toolkit: the complete guide to dimensional modeling », 2nd edition, Wiley Computer Publishing, 2002.

[21]. Franco J. M, « Le Data Wharehouse : objectifs, définitions, architectures », Eyrolles, 1997.

[22]. Joachim H., Markus S., « Data Warehousing at the Crossroads », Manifesto of a Dagstuhl, Perspectives Seminar,2005.

[23]. MESSAOUDI M., AFRIT M.-A « Contribution à la conception et au déploiement du data warehouse, Volet: suivi d'activation des puces et de rechargement des cartes des clients pré-payés », Mémoire de fin d'études, INI, Alger, 2007.

[24]. BOULIL K., ADOUANE A., « Conception et réalisation d'un système interactif d'aide à la conception des Entrepôts de Données XML », Mémoire de fin d'études, INI, Alger, 2007.

[25]. Ladjel B., « La conception physique des data warehouses », laboratoire LISI/ENSMA, France, 2006.

[26]. Mc Hugh, R., F. Bilodeau, et al. « Annexe sur JMap Admin SOLAP », Département des sciences géomatiques, Université Laval, Montréal, Cananda. 2006.

[27]. Riadh B.-M « Couplage de l'analyse en ligne et de la fouille de données pour l'exploration, l'agrégation et l'explication des données complexes », Doctorat de l'université Lumière Lyon 2, novembre 2006.

[28]. Chloé F., Marie C. « étude comparative des différents outils d'ETL (Extract, Transform, Load) du marché », IUP MIS, Université de Bretagne-Sud, 2005.

[29]. Salvatore T.-M, Alan R.-H, « Integrated decision support systems: A data warehousing perspective », Decision Support Systems Journal 11159, Elsevier, 2005.

[30]. « DW Facile, ETL, Modélisation dimensionnelle et Data Warehousing ». www.systemeetl.com.

[31]. Matthieu G., « Comment mettre en place un reporting efficace pour piloter l'activité d'une entreprise? L'exemple de GEFCO », Master de l'Institut Supérieur du Commerce de Paris, 2007. www.m-grimaud.com/memoire-business-intelligence.

[32]. Labidi S., Lejouad W. « De l'intelligence Artificielle Distribuée aux Systèmes Multi-Agents». Rapport de Recherche, N°2004. INRIA., 1994.

[33]. Romaric C., « Des agents intelligents dans un environnement de communication multimédia : vers la conception de services adaptatifs », Doctorat de l'Université Henri Poincaré - Nancy 1. décembre 2003.

[34]. Arsène S., « Systèmes Multi-Agents: Une Analyse Comparative Des Méthodologies De Développement », Mémoire de Maîtrise de l'Université du Québec à Trois-Rivières, Octobre 2001.

[35]. Ferber J. « Les Systèmes Multi-Agents Vers une intelligence collective ». Interéditions, 1995.

[36]. Roda C., Jennings N.-R., et al « Cooperation Framework for Industrial Process Control ». October 1990.

[37]. Guillaume C., « Dialogue Entre Agents Naturels Et Agents Artificiels : Une Application Aux Communautes Virtuelles », Doctorat de l'INPG, Institut National Polytechnique de Grenoble, Décembre 2002.

[38]. Briot J.-P, Yves D., « Principes et architecture des SMA », Hermès, 2001.

[39] Kolski C., Emmanuelle G.-S, « Conception des systèmes multi-agents : pistes de réflexion en vue de futures coopérations entre ergonomes et informaticiens », LAMIH-UMR CNRS, Université de Valenciennes et du Hainaut-Cambrésis, 2004.

[40]. Ronan Q., « Les Systèmes Multi-Agents pour les Environnements Virtuels de Formation. Application à la sécurité civile », Doctorat de l'Université de Bretagne Occidentale, octobre 2000.

[41]. Imed J. et Brahim C.-D, « Aperçu sur les systèmes multi-agents », Série Scientifique, CIRANO, Montréal, Juillet 2002.

[42]. Philippe P., Brahim C-D, « Modèles Des Dialogues Entre Agents : Un Etat De L'art », Cahiers Romans de Sciences Cognitives, in Cognito VOL. 1 No. 4, 2004.

[43]. Olivier L., « Modélisation et simulation orientées agents de chaînes logistiques dans un contexte de personnalisation de masse » doctorat de l'université Paul CÉZANNE, 2006.

[44]. Serment J., « Une infrastructure d'intégration à base d'agents logiciels pour l'élaboration de systèmes d'aide à la décision environnementale : Application à la gestion hydraulique camarguaise », doctorat de l'université Paul CEZANNE, 2007.

[45]. Marc C., « NetSA : une architecture multi-agents et son application aux services financiers » Mémoire de maître ès sciences de l'Université Laval, Avril 1999.

[46]. Marc C., Brahim C-D, Nader T., « NetSA : une architecture multiagent réutilisable pour les environnements riches en informations », Département d'informatique, Université Laval, Canada, 2001.

[47]. Widom J., « Research problems in data warehousing », Proceedings of the 4 International Conference on Information and Knowledge Management - ACM CIKM'95, Baltimore Maryland, USA. 1995.

[48]. Wiener J.-L., Gupta H., et al., « A System Prototype for Warehouse View Maintenance », In Proceedings of the ACM Workshop on Materialized Views: Techniques and Applications, pp. 26-33, Montreal, Canada, 1996.

[49]. Hammer J., Garcia-Molina H., et al., « The Stanford Data Warehousing Project» , IEEE Data Engineering Bulletin, 1995.

Résumé:

Le processus de prise de décision est une mission critique dans les entreprises, qui dépend à des systèmes d'aide à la décision pour extraire, analyser et interpréter les informations à partir de multiple sources de données hétérogènes et distribuées. Il est paru que le data warehouse est nécessaire pour optimiser l'accessibilité et l'utilisation de données.

L'objectif de notre travail est de pouvoir impliquer les agents pour la conception et l'implémentation d'une architecture du data warehousing capable d'intégrer les informations des bases de données réparties et hétérogènes. Pour cela nous avons proposé une architecture de data warehousing basée sur l'interaction/coopération des agents, cet architecture articule autour de trois couches et de dix agents distribuées dans ces couches, qui se chargent de l'assistance dans le processus de l'entreposage de données par l'intégration de données complexes dans le data warehouse, tout en facilitant aussi l'accès à cette base par les utilisateurs humains ou logiciels.

Mots clés: Système d'aide à la décision, Data Warehouse, Agents, système multi-agents.

Abstract

The process of decision-making is a critical mission in enterprises, which depends on decision support systems for extracting, analyzing and interpreting information from multiple heterogeneous, distributed data sources. It is assumed that data warehouses (DW) are required for optimized data accessibility and use.

The objective of our work is to allow the utilization of agents in the design and implementation of architecture of data warehousing able to integrate information from distributed and heterogeneous data bases. For that we was have proposed an architecture of data warehousing based on the interaction / cooperation between agents, this architecture articulates around three layers and ten agents distributed in these layers, witch charge to the assistance in the process of data warehousing by integration of complex data in the data warehouse, while also we facilitated the access to this base by humans or software users.

Keywords*:* Decision Support Systems, Data warehouse, Agents, Multi-agent system.

خلاصة

عملية صنع القرار هي مهمة حاسمة في المؤسسات التي تعتمد على نظم دعم القرارات لاستخراج وتحليل وتفسير المعلومات من عدة مصادر غير متجانسة وموزعة. ومن الواضح أهمية مستودعات البيانات (DW)من أجل تحسين إلى أقصى حد أمكانيه الوصول إلى البيانات واستخدامها.

الهدف من عملنا هو إتاحة الفرصة لاستخدام الوكلاء في مجال التصميم والتنفيذ قادرة على دمج المعلومات وتوزيعها من قواعد البيانات غير متجانسة. من أجل كل هذا قد اقترحنا تصميم مستودعات البيانات على أساس التفاعل/التعاون بين الوكلاء ، هذا التصميم يدور حول ثلاث طبقات وعشرة وكلاء موزعون في هذه الطبقات ، المسئول عن المساعدة في عملية تخزين البيانات عن طريق إدماج البيانات المعقدة في مستودع البيانات ، في حين أننا أيضا نسهل الوصول إلى هذه القاعدة من قبل المستعملين البشر أو البرمجيات.

الكلمات الرئيسية: نظم دعم القرارات ، مستودع البيانات ، الوكلاء ، نظام متعدد الوكلاء.