N^{0}d'ordre :

Université * Mohamed BOUDIAF* de M'sila

Faculté des Sciences et Sciences de l'ingénieur

Département d'informatique

MEMOIRE

Présenté pour l'obtention du diplôme de :

Magister

Spécialité : Informatique
Option : Informatique industrielle

Par :

KADRI Said

Thème :

INTEROPERABILITE DES BASES DE DONNEES HETEROGENES ET REPARTIES

Soutenu publiquement le : …. /…./ ……. devant le jury composé de :

KHELLADI Abdelkader	Pr. USTHB	***Président***
AHMED Nacer Mohamed	Pr. USTHB	***Rapporteur***
MEZIANE Aider	Pr. USTHB	***Examinateur***
BOUDERAH Brahim	Mc. Université de M'sila	***Examinateur***
MOUSSAOUI Adel	C.c. Université de M'sila	***Invité***

Promotion 2006

REMERCIEMENTS

Je tiens à remercier :

- En premier lieu le directeur du mémoire **Pr.AHMED Nacer Mohamed** d'avoir accepter l'encadrement de ce travail, il n'a ménagé ni son temps ni sa peine pour mener à bien le travail, je lui en suis très reconnaissant et lui adresse ma profonde gratitude et mes sincères remerciements.

- Le directeur des recherches du laboratoire LIRIS à l'université Claude Bernard de Lyon **Dr. Djamel BENSLIMANE** pour sa précieuse assistance et surtout pour ses articles très intéressants qui m'avaient bien éclairé les choses.

- L'ex-chef du département informatique et notre enseignant du module analyse numérique en post-graduation **Mc. BOUDERRAH Brahim** pour m'avoir orienté durant la réalisation de ce mémoire, je le remercie sincèrement.

- Le chef du département informatique, **Mr. MAHDJOUBI Rosafi**, ainsi que le président du conseil scientifique **Mr. GASMI Abdelkader** pour leur soutien scientifique et moral.

- Mes remerciements s'adressent également au chef du service hydraulique agricole de la direction de l'hydraulique de de M'SILA **Mr. ZEGHLACHE Mustapha** qui était très compréhensible, généreux, et tolérant, et qui n'a pas hésité de m'aider du début jusqu'à la fin du mémoire, je lui en très reconnaissant.

- L'ensemble des enseignants de post-graduation à noiter :
  Pr. KHALLADI A/K – CERIST. Alger, **Pr. BENHOCINE A/H** – Arabie Saoudie, **Pr. BELOUADAH. H** Dept. Sciences commerciales, **Mc. MIHOUBI . D** - Dépt. Maths, **Mme. FERNINI** – Institut des langues.

- Mes collègues les enseignants de l'institut d'informatique
- Mes collègues du service hydraulique agricole – DHW . M'sila

- Les membres de jury qui m'ont fait un grand honneur en acceptant la valorisation de ce modeste travail.

- Tous ceux qui m'ont aidé de près ou de loin dans l'élaboration de ce modeste travail.

KADRI Said

SOMMAIRE

CHAPITRE II : LES DIFFERENTES APPROCHES D'INTEROPERABILITE DES BASES DE DONNEES HETEROGENES ET REPARETIES **27 - 108**

INTRODUCTION

1. Avant propos :

Le domaine des bases de données s'est profondément transformé ces quinze dernières années sous l'influence de l'évolution globale des logiciels (objets, composants, webs, …) et du matériel (augmentation vertigineuse de la puissance des machines, …)

Parallèlement, de nouvelles applications ont généré de nouveaux besoins de modélisation et de manipulation des données.

Le problème d'interopérabilité des bases de données et des SGBD prend aujourd'hui une très large place avec l'interconnexion massive des systèmes d'informations via internet et intranet .

2. Problématique :

La mise en place de systèmes d'informations distribués déroulant dans un environnement réseau tel que le réseau mondial internet ou même un réseau local intranet à l'aide de middlewares relationnels ou objets de type RDA (Remote Data Acces) ou CORBA (Common Object Request Broker Architecture) nécessite l'accès à de multiple sources de données distribuées et hétérogènes telles que : les bases de données relationnelles, les bases de données orientées objet, les documents multimédia et les fichiers semi-structurés. **[GAR et ALL 2002]**

Plusieurs problèmes nouveaux posés par la réalisation de tels systèmes, parmi ces problèmes on peut citer :

- ☒ La modélisation et l'interrogation de multiple sources de données distribuées et hétérogènes .
- ☒ La gestion des transactions réparties sur plusieurs sites distants.
- ☒ L'évolution et l'optimisation des performances des middlewares utilisés .
- ☒ L'extraction efficace de connaissances depuis de grandes bases de données.
- ☒ L'interopérabilité des bases de données hétérogènes et distribuées.

Ces problèmes deviennent encore plus complexes lorsque les sources de données contiennent des données multimédia.

Dans les domaines décisionnels et documentaires, les interrogations multibases sont difficiles à traiter à cause de :

- L'hétérogénéité des sources.
- La complexité des requêtes .
- La multiplicité des sites.
- La diversité des parcours de chemins .
- Le manque de méta-connaissances sur les sources de données.
- L'absence de modèle de coût global.

Dans les domaines techniques tels que :

- Les systèmes de contrôle et de supervision .
- La gestion des réseaux électriques ou réseaux de télécommunication.

Et les domaines de transaction tels que :

- Le commerce électronique.
- Les opérations bancaires.

Les mises à jour multibases sont des opérations particulièrement importantes. Et La cohérence de l'un de ces systèmes repose essentiellement sur le respect des propriétés transactionnelles (atomicité, cohérence, isolation, durabilité).

Malheureusement la garantie de ces propriétés par les transactions globales accédant à plusieurs sites en même temps est très difficile du fait de l'autonomie et de l'hétérogénéité des sources de données locales.

Notre travail est consacré à étudier l'un de ces problèmes fastidieux qui est le problème de l'interopérabilité des bases de données hétérogènes et réparties .

Une simple définition de l'interopérabilité entre systèmes d'informations dits interopérables, est la possibilité de ces systèmes à échanger des services et des fonctions, et de se collaborer entre eux, même s'ils ont des natures très différentes afin de réaliser des fonctionnalités communes.

Une forme encore très simple de l'interopérabilité est l'échange de données entre deux systèmes, ou l'un envoie périodiquement des données vers l'autre.

3. Objectifs visés :

1 – Comprendre le problème de l'interopérabilité entre bases de données hétérogènes et réparties, et définir sa position et son importance dans le domaine de communication et d'échange de données entre systèmes d'informations.
2 – Acquérir des compétences dans la consolidation d'entités persistantes hétérogènes.
3 – Présenter les différentes architectures existantes qui permettent un haut niveau d'interopérabilité de bases de données hétérogènes et réparties.
4 – Permettre un accès intégré et homogène à des bases de données hétérogènes distribuées entre des sites distants en tenant compte des profils des utilisateurs et des différents contextes d'exécution.
5 – Conserver l'autonomie des bases de données existantes :

- Conserver le système de gestion des données.
- Conserver la structure des données .
- Conserver les données elles mêmes .

4. Organisation du travail :

Pour mieux cerner le problème de l'interopérabilité des bases de données hétérogènes (et/ou) réparties, nous avons adopté une méthodologie bien progressive et claire. Cette méthodologie est envisagée par :

- Une introduction générale dans laquelle on a exposé le problème traité « ***le problème d'interopérabilité des bases de données hétérogènes et réparties*** » dans son cadre général, ainsi que les nouvelles exigences nécessitant cette interopérabilité.

- Chapitre I : intitulé : « ***Etat de l'art*** » consacré à présenter toutes les notions de base, les concepts, les principes fondamentaux utilisés au cours du travail. Dans ce même chapitre seront exposées d'une façon explicite les différentes approches et tendances développées par les spécialistes pour résoudre le problème d'interopérabilité.

- Chapitre II : intitulé : « ***Les différentes approches d'interopérabilité des bases de données hétérogènes et réparties*** » expose en détail les différentes approches en concluant chaque approche par les avantages apportés, ainsi que les inconvénients détectés. Cela peut donner aux chercheurs et aux intéressés une sorte de comparaison entre les différentes solutions envisagées.

- Chapitre III : consacré à spécifier une approche qu'on a jugé plus utile et efficace, c'est « ***l'approche XML*** ». Le choix de cette approche n'était pas arbitraire ni au hasard, mais basé sur une étude approfondie des différentes approches exposées, ainsi qu'un suivi méthodologique des dernières recherches publiées dans le domaine d'interopérabilité des systèmes.

- Chapitre IV : dans lequel, nous avons exposé notre propre approche, c'est une nouvelle approche qui résulte de la combinaison de trois approches et qui adopte une architecture multi-agents. Cette approche pourra être considérée comme une solution améliorée au problème de l'interopérabilité

- Le travail se termine par une conclusion générale portant nos propres observations, et surtout nos propositions relatives au sujet traité. Cette conclusion sera sans doute un point de départ pour d'autres études avec plus d'appréciations dans le but de concevoir une solution meilleure et plus optimale au problème traité.

Etat de l'art

I.1. Que signifie le terme « interopérabilité » ?

L'interopérabilité est devenue une nécessité pour répondre aux besoins d'échange d'informations entre systèmes d'informations hétérogènes, elle traduit la capacité d'un système d'informations à se collaborer avec d'autres systèmes de natures par fois très différentes **[BEN 1999]**.

Le terme interopérabilité signifie en clair l'accès uniforme et transparent à des bases de données hétérogènes et géographiquement distribuées à l'aide de mécanismes assurant un fonctionnement coordonné et coopéré entre les différents systèmes applicatifs dans un contexte ou peuvent varier les plates-formes logicielles et matérielles .

I.2. Motivation :

1. De vaste collections de données sont disponibles .
2. Les données sont coûteuses à acquérir et à mettre à jour.
3. Les nouvelles applications requièrent des accès coordonnés à plusieurs sources de données .

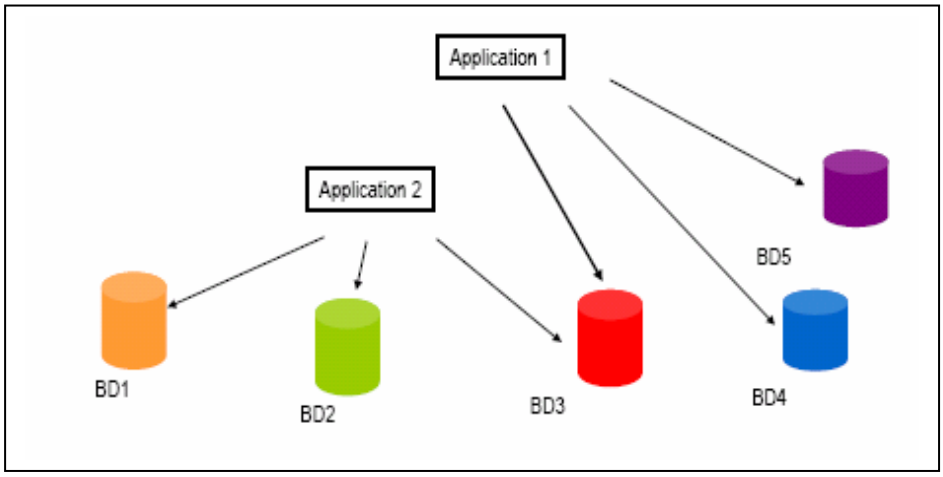


Fig I.1. Exemple d'applications accèdant à plusieurs bases de données

I.3. Systèmes interopérables :

Deux systèmes sont dits interopérables lorsque :

1. Ils aient une compréhension mutuelle des éléments qu'ils partagent .
2. Capables de découvrir d'une manière dynamique les différentes sources de données.
3. Ils peuvent échanger des messages et des requêtes.
4. Fonctionnent comme unité unique pour les taches communes.
5. Utilisent des fonctions des uns et des autres.
6. Fonctionnent comme des clients et des serveurs.
7. Capables de communiquer même avec des composants internes incompatibles
8. Approximation des requêtes multisources.

I.4. Propriétés fondamentales des systèmes interopérables :

1 - Autonomie : Une source participant dans un système interopérable doit fonctionner comme avant sa participation (garde son indépendance).

2 - Distribution : les données générées par un système interopérable proviennent de plusieurs sources de données de façon que chaque source met une partie de ses propres données à la disposition des autres sources participants au système .

3 - Hétérogénéité : Chaque source choisie du système est conçue indépendamment des autres sources (du coté matériel, SE, communication, périphériques, langages, schémas, …).

4 – Transparence d'accès aux données : Un système interopérable accède à des sources hétérogènes d'une façon uniforme et efficace.

5 – Extensibilité/Evolutivité : Possibilité d'ajouter de nouvelles sources de données sans perturber le fonctionnement du système .

I.5. Objectifs de l'interopérabilité des bases de données :

1. Permettre à l'utilisateur d'utiliser de façon transparente des données issues d'un ensemble de systèmes d'informations autonomes, répartis, et hétérogènes.
2. Développer des architectures et des outils pour le partage, l'échange et le contrôle des données.

I.6. Quand se pose le problème d'interopérabilité entre bases de données ?

Le problème de l'interopérabilité entre base de données se pose fortement dans des environnements hétérogènes (et/ou) répartis à cause : **[GAR 2001]**

1. La diversité des sources de données
2. La diversité des données.
3. La diversité des bases de données.
4. La diversité des modes d'accès aux données.
5. La diversité des méthodes d'accès.
6. Les problèmes d'échange .

Un exemple très simple qui envisage la nécessité d'interopérabilité entre bases de données hétérogènes (et/ou) réparties est celui d'un estivant qui cherche ou il doit passer ses vacances .

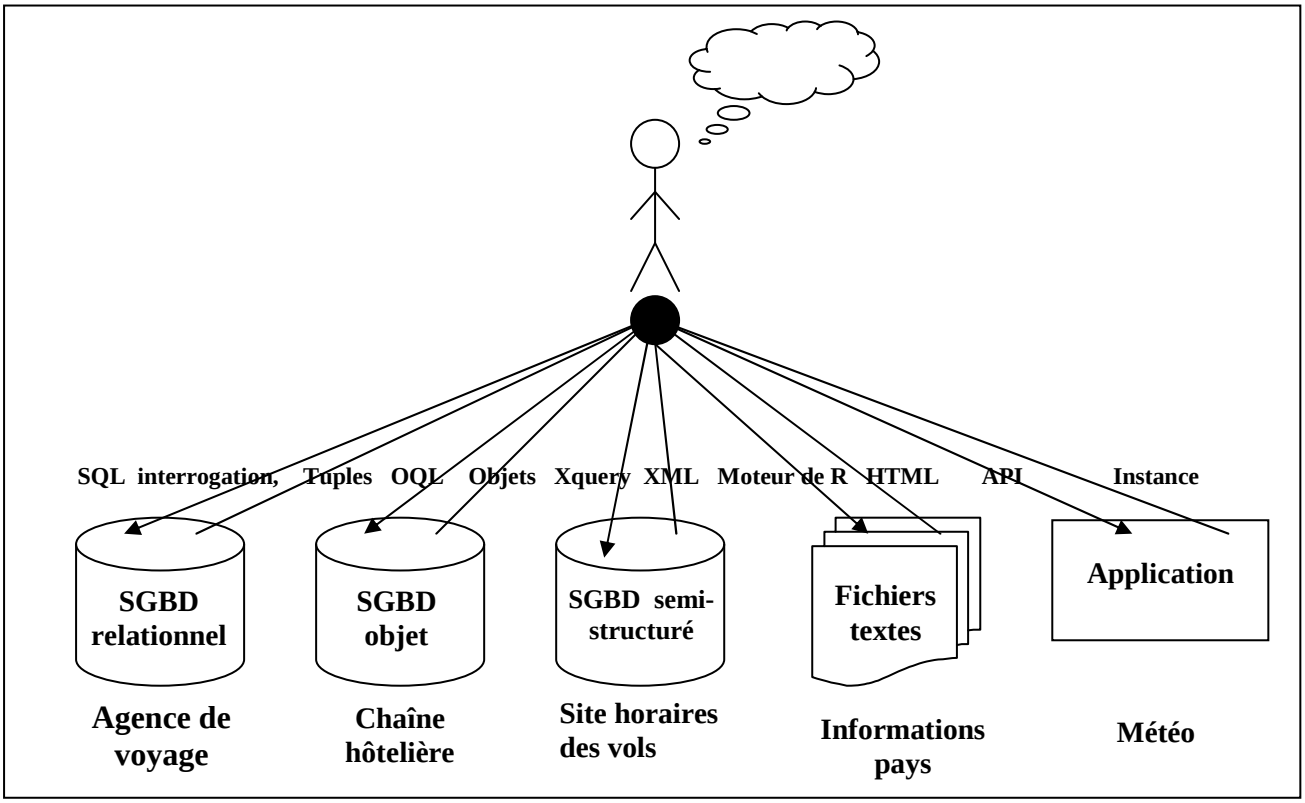


Fig I.2. Un premier exemple sur l'interopérabilité des bases de données

I.6.1.Diversité des sources de données :

Une source de données peut être :

- ☒ Un SGBD relationnel .
- ☒ Un SGBD orienté objets
- ☒ Un page web
- ☒ Un tableur
- ☒ Un système classique de fichiers (SGF).
- ☒ Une Application.

Il existe plusieurs façons pour décrire une source de données qui peut être une parmi plusieurs dans un système réparti

1. ***Par localisation :*** on peut localiser directement la source de données suivant sa nature :

   - ☒ Référence de site (URL, IP : Port, Annuaire LDAP, …).
   - ☒ Protocole de communication (TCP/IP, IPX, Apple Talk, …).
   - ☒ Moyen d'accès (pilote ODBC, pilote JDBC, interface API)
   - ☒ Support (Pages web, SGBD, Document)

2. ***Par type de données qu'elle peut gérer :***

   - ☒ Données structurées (SGBD-R, SGBD-OO, …)
   - ☒ Données semi-structurées (XML, HTML, OEM, …).
   - ☒ Données non structurées (Images, textes, multimédia, …)

3. ***Sa possibilité d'interrogation des données :***

   (SQL , OQL, XQUERY, Moteur de recherche web, API, …).

4. ***Par le format de résultats qu'elle peut délivrer*** **:**

(Doc XML, Doc HTML, Tuples, Textes, OEM, …)

I.6.2. Diversité des données (Différentes *Formes de données*) :

Généralement, les données manipulées par l'utilisateur ont plusieurs formes :

I.6.2.1. ***Données structurées :*** les données structurées sont des données bien organisées et stockées sous un format spécifique tel que :

- ☒ Des fichiers classiques .
- ☒ Des bases de données relationnelles .
- ☒ Des bases de données orientées objet .

I.6.2.2. ***Données sem-structurées :*** telles que :

- ☒ Les documents HTML.
- ☒ Les documents XML.

I.6.2.3. ***données non structurées :*** telles que :
les images, les textes, les données multimédia, …etc

I.6.3. Diversité des bases de données :

I.6.3.1. ***Bases de données autonomes (centralisées) :*** une base de données centralisée est une base de données gérée par un seul SGBD et stockée dans sa totalité à un emplacement physique unique. Ses divers traitements sont confiés à une seule unité de traitement .

Elle est caractérisée par :

a) - Au niveau de la conception :

- Son propre modèle de données (SGBD-R, SGBD-OO, XML, …).
- Son propre langage d'interrogation (SQL, OQL, XQUERY, …).
- Sa propre interprétation sémantique de données, des contraintes, et des fonctions

b) - Au niveau de communication :

- Une base de données centralisée décide quand et comment répondre aux différentes requêtes.

c) - Au niveau d'exécution :

- Une base de données centralisée n'a pas besoin d'informer d'autres systèmes concernant l'ordre d'exécution des transactions locales ou des opérations externes
- Ne distingue pas entre des opérations locales et globales .

d) - Au niveau d'association :

- Une base de donnée centralisée décide quand se connecter ou se déconnecter du système global.
- Elle décide également quelles données et fonctions à partager et avec quels type d'utilisateurs.

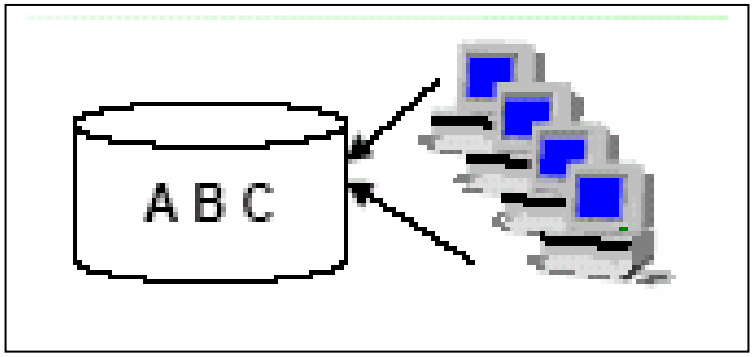


Fig I.3. Une base de données centralisée (autonome)

☒ ***Avantages :***

- Egalité d'accès .
- Facilité de gestion.

☒ ***Inconvénients :***

- Contentions sur la base de données.
- Intolérance aux pannes : lorsque le système centralisé tombe en panne, les données deviennent inaccessibles par les utilisateurs.

I.6.3.2. ***Bases de données distribuées (réparties) :*** se sont des bases de données dont les données sont stockées sur des postes (sites) géographiquement distants, reliées logiquement par un réseau intranet ou extranet ou même le réseau mondial internet.

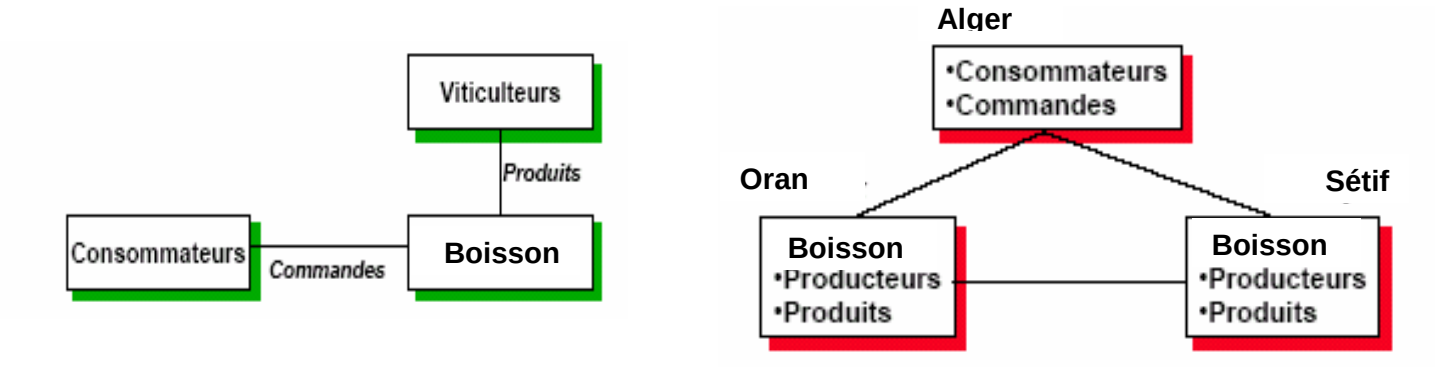


Fig I.4. Une base de données distribuée (Répartie)

Chaque site possède une machine avec un SGBD local, et une partie de la base de données répartie, et pourrait effectuer certains traitements locaux en accèdant aux données par l'intermédiaire du schéma global et non par l'intermédiaire du schéma local du site.

Le SGBD réparti décharge les utilisateurs de tous les problèmes de concurrence, de fiabilité, d'optimisation des requêtes et des transactions relatifs aux données gérées par les SGBD des différents sites. **[NAD 2005]**

Exemple : prenons l'exemple d'une banque qui centralise les données à propos les comptes tiers au niveau de son siège central, et une autre banque qui utilise une base de données répartie pour distribuer les données relatives aux comptes sur les différentes agences interconnectées entièrement ou partiellement via un réseau.

☒ ***Avantages des bases de données réparties :***

- Les données peuvent être localisées plus proche du site demandeur, ce qui permet de réduire le coût de communication par rapport au système centralisé, d'ou la rapidité d'accès aux données .
- Certaines données demandées par un utilisateur et qui sont stockées au niveau du site peuvent être accédées directement par le site (Autonomie local du site).
- Accès toujours possible aux autres sites.
- Les groupes locaux peuvent exercer un plus grand contrôle sur les données et assure l'intégrité des données (contrôle local).
- Il est plus facile et économique d'ajouter un nouveau poste et l'intégrer dans le système existant à travers un réseau.
- Maintenir une vision unique sur les bases de données malgré sa répartition.

☒ ***Inconvénients :***

- La gestion globale de la base de données répartie est plus complexe qu'une base de données centralisée.

Généralement, on distingue deux stratégies de répartition :

1 – Duplication (Réplication) des données : qui consiste à créer plusieurs copies des mêmes données sur plusieurs sites, cette duplication pourra être totale, C.à.d qui touche tous les sites, ou partielle qui vise certains entre eux.

☒ ***Avantages :***
- Disponibilité des données tout le temps et sur tous les sites .
- Rapidité d'accès aux données locales.
- Parallélisme de traitement, car des requêtes ou des transactions relatives à un ensemble de données peuvent être lancées en même temps.

☒ ***Inconvénients :***
- Augmentation du volume global de la base de données à cause la duplication.
- Difficulté de coordination des mises à jour qui doivent être appliquées sur les différentes copies en même temps.

2 – Répartition : fragmentation d'une relation R verticalement (par projection) ou horizontalement (par sélection).

☒ ***Avantages :***
- Rapidité d'accès aux données locales .
- Autonomie de chaque site local.
- Accès possible aux autres sites

☒ ***Inconvénients :***
- La gestion globale de la base de données est un peu difficile.

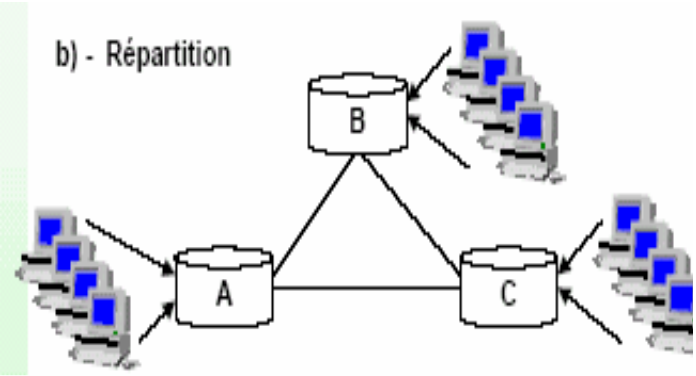


Fig I.5. Une base de données répartie par fragmentation

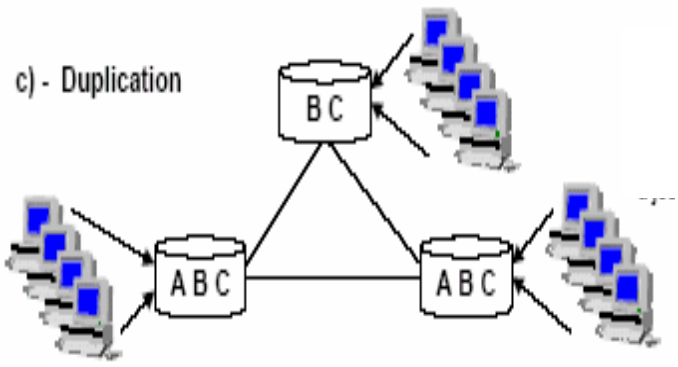


Fig I.6. Une base de données répartie par duplication

☒ ***Conception d'une base de données répartie :***

Pour concevoir une base de données répartie, on utilise une approche descendante comme il est montré sur la figure ci-après.
Cette approche est basée sur :

- La maîtrise de la complexité de la répartition (fragmentation, duplication, placement).
- Définition des schémas locaux.

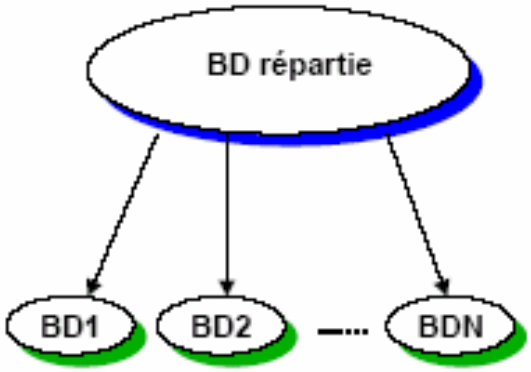


Fig I.7. Conception d'une base de données répartie – Approche descendante

☒ ***Principe de fonctionnement d'une base de données répartie :***

Dans chaque site on trouve un SGBD local qui gère les données stockées sur ce site, plus une copie du SGBD distribué, un dictionnaire de données directives (DD/D) qui contient des informations sur la localisation des données sur le réseau.

Si une requête est émise par un utilisateur ou un programme d'application, cette dernière doit passer par le SGBD distribué qui détermine si la transaction est locale avec des données situées sur un seul site, si c'est le cas, la transaction sera orientée vers le SGBD du site concerné.

Dans le cas contraire, c.à.d la transaction est globale avec des données stockées sur plusieurs sites, elle sera exécutée par le SGBD global en collaboration avec les SGBD locaux. **[NAD 2005]**

I.6.3.3. Multi-bases : Plusieurs bases de données (hétérogènes ou non) capables d'interopérer sans une vue commune (Absence du modèle commun comme avec les BDF). Un système multi-bases permet l'interopérabilité de plusieurs SGBD locaux.

I.6.3.4. ***Bases de données hétérogènes :*** se sont des bases de données différentes en : sources, schémas de modélisation, structures de représentation, langage d'interrogation, … etc.

• ***Systèmes homogènes et systèmes hétérogènes :***

Le terme hétérogène est l'opposé du terme homogène, généralement on parle de bases de données ou plus général de systèmes homogènes si toutes les données ont le même format, on parle aussi de modèle ou de schéma de représentation de données, et le logiciels qui manipulent ces données est le même pour toutes les sources de données, d'une manière générale des systèmes sont dits homogènes si :

☒ Il ont le même univers de discours .
☒ Il ont le même modèle de données (modèle R , modèle réseau, modèle OO, …).
☒ Le même logiciel qui gère toutes les données sur tous les sites .

Par contre des bases de données sont dites hétérogènes si elles ont :
Des formats multiples, des formalismes (schémas), des structures, des types, des médias, … etc.

L'hétérogénéité est donc indépendante de la distribution physique des données à travers un réseau, car on peut avoir des données distribuées à travers des postes différents d'un réseau mais qui sont homogènes (c.à.d un système distribué mais homogène). De même on peut avoir des données stockées sur le même poste mais qui sont hétérogène (système centralisé mais hétérogène).

D'une manière générale, des systèmes sont dits hétérogènes si :

- Ils appartiennent à des univers de discours différents.
- Il ont des modèles de données différents
- Il ont des types de données et de formats différents.
- Différents moyens de stockage (fichiers, BDD, …).
- Différents logiciels gérant les données sur les différents sites.
- Différents SGBDs.
- Différents langages de programmation pour les applications associées.
- Différents langages d'interrogation .
- Différentes capacités de requêtes.

• ***Classes d'hétérogénéité :*** **[BEN 2002] [CHR 1996]**

1- Hétérogénéité syntaxique : Utilisation des modèles différents dans la conception des bases de données, ce qui nécessite une traduction de tous les modèles vers tous les modèles lors de coopération ou vers un modèle commun appelé modèle canonique ou modèle pivot (l'approche de fédération des bases de données).

2- Hétérogénéité structurelle : Différentes représentations des mêmes concepts dans des bases de données différentes (utilisation des noms différents pour le même concept, différents types de données, différents attributs, différentes unités, …etc) .

3- Hétérogénéité sémantique : Résulte des conceptions indépendantes des bases de données, ce qui donne une signification, interprétation ou même utilisation différente de la même donnée ou objet du monde réel d'une base à une autre.

Afin d'éviter les conflits dus à l'hétérogénéité sémantique, et pour aboutir une intégration cohérente des bases de données, l'approche de fédération propose une solution adéquate consistant à effectuer une analyse sémantique comparée des données entre les différentes bases de données, préalable à la fédération, et souvent groupée avec la phase de traduction.

Ces mêmes classes peuvent être distribuées sur trois types comme suit :

1 – Hétérogénéité des données : les données manipulées par un tel système hétérogène sont différentes en : **[GAR 2001]**

- ***Typage :*** par exemple l'information « adresse » à comme type :

  varchar2(64) sous oracle.
  String sous postgreSQL.

- ***Structure :*** par exemple l'entité personne a comme attributs :

  Nom, prénom, age, … dans une source de données
  Nom, adresse, N°SS d ans une autre source .

- ***Un même nom*** peut être utilisé pour désigner des entités différentes entre plusieurs sources (les homonymes).

- ***Utilisation des noms différents*** par les différentes sources pour désigner la même entité (les synonymes).

2 – Hétérogénéité de schémas ou de modèles : chaque source de données utilise son propre schéma ou modèle pour représenter ses données .

Source 1 : SGBD relationnel (Description par tables)

BUVEURS

Nom	DateN	Pays	Type

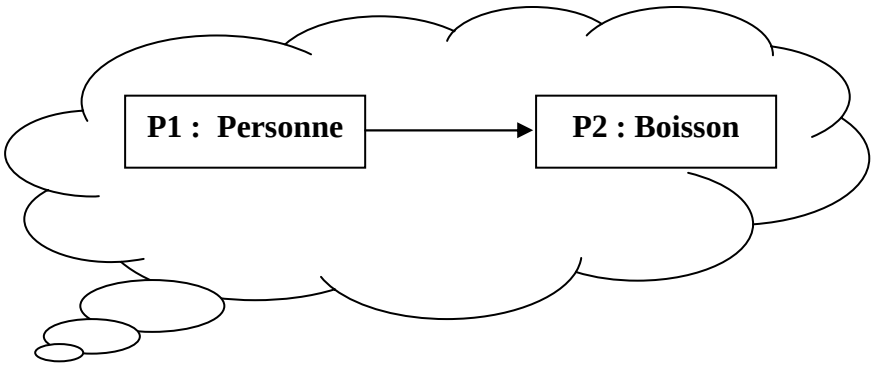
BOISSONS

Nb	Cru	Mill	Degré

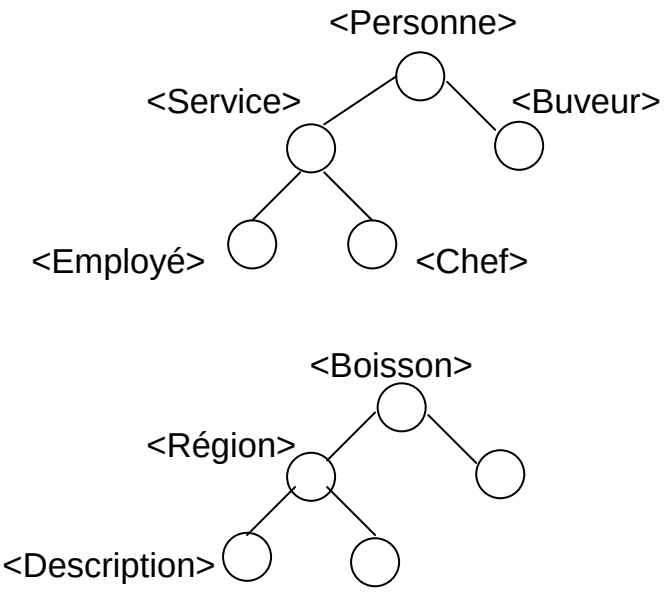
Source 2 : Repository XML (Description par balises)

< !ELEMENT Boissons(cru,degré,description+)>
< !ATTLIST Boissons Nb, CDATA # IMPLIED>
< !ELEMENT Buveurs(Nom, place, date, type) >
< !ATTLIST Buveurs Nb, CDATA # IMPLIED>
< !ELEMENT Catalogue(boisson, offre, publicité ?) +>

Source 3 : Site Web (Sous forme de pages)



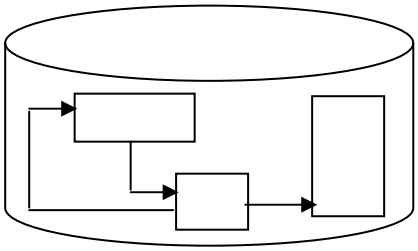
Source 4 : LDAP (Sous forme d'un arbre hiérarchique)



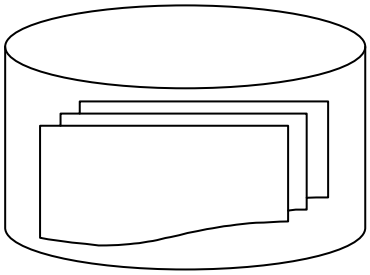
3 – Hétérogénéité des langages :

Qu'il s'agit de langages utilisés pour formuler des requêtes (SQL, OQL, CGI, BIN, …) ou même les méthodes de réponses à ces requêtes ou bien de restitution des résultats (tuples, objets, pages web, instances, …)

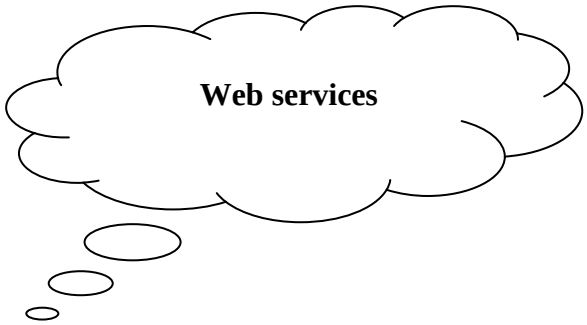
Source 1 : SGBD-R ➔ ODBC / JDBC / SQL / …



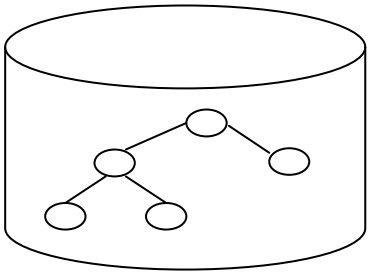
Source 2 : XML Repository ➔ SOAP/XQUERY



Source 3 : Site Web ➔ Google text queries



Source 4 : LDAP ➔ LDAP Query



- ***Applications des bases de données hétérogènes :***
  - Commerce électronique.
  - Télé médecine.
  - Economie.
  - Environnement.
  - Génome.

I.6.3.5. ***Bases de données fédérées (Federated Data Base) :*** une collection de bases de données hétérogènes (obligatoirement hétérogènes) capables d'interopérer via une vue commune (modèle commun ou pivot). Une caractéristique importante des bases de données fédérées est la préservation de l'autonomie au niveau local.

Le logiciel chargé de fournir le contrôle, la manipulation, et la coordination des différents composants d'une base de données fédérée est appelé un système de gestion de base de données fédérée.

☒ ***Objectifs de la fédération :***

- Donner aux utilisateurs une vue unique des données implémentées sur plusieurs systèmes à priori hétérogènes (en plate-forme et SGBD).
- Faire cohabiter les différents systèmes d'une entreprise tout en leur permettant d'interopérer.

☒ ***Fédération/Intégration des bases de données :***

Pour concevoir une base de données fédérée, on utilise une approche ascendante comme il est montré sur la figure ci-après.
Cette approche est basée sur :

- La maîtrise de l'hétérogénéité sémantique et syntaxique des sources de données.
- La maîtrise de l'intégration des schémas locaux pour créer un schéma global (schéma fédéré).

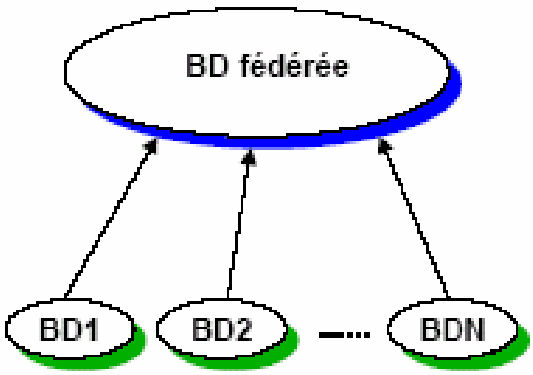


Fig I.8. Fédération des bases de données – Approche Ascendante

I.6.4. Diversité de mode d'accès (ou de consultation) :

☒ Différentes façons d'interroger les données :
Par exemple formuler des requêtes sous un langage de requêtes spécifique (SQL, XQUERY, …).

☒ Différentes manières pour la source de répondre ou délivrer un résultat suite à une requête .

- Les SGBD-R seront interrogés par requêtes SQL et donnent comme résultat des tuples .
- Les sites web seront interrogés par requêtes XQUERY pour avoir une réponse sous forme de documents XML ou HTML.

I.6.5. Diversité de méthodes d'accès :

L'utilisateur peut interagir avec la source de données par des méthodes d'accès différentes telles que :

- ☒ Des protocoles de communication (ODBC, JDBC, HOP, HTTP, SOAP, …).
- ☒ Différentes interfaces d'accès (Interfaces de programmation, interface graphique, invite, ….)

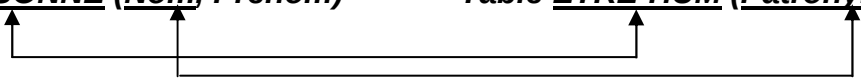
I.6.6. Problèmes d'échange : [Mim 2001]

La plupart des problèmes d'échange dus à l'hétérogénéité des sources de données, ainsi que les bases de données manipulées, et parmi ces problèmes on cite les suivants :

1 – Diversité des dénomination des concepts et des objets utilisés par les différentes sources

Exemple :

Table PERSONNE (Nom, Prénom) ***Table ETRE-HUM (Patronyme, Prénom)***



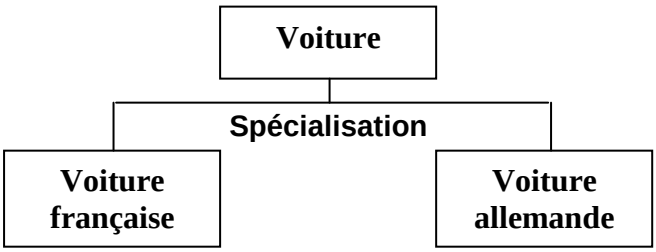
On constate que le même concept porte deux noms différents entre les différentes sources

Personne ➔ être-hum
Nom ➔ patronyme

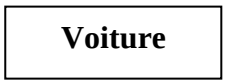
2 – Diversité de modélisation conceptuelle :

Par exemple l'objet voiture peut être modélisé différemment par deux sources de données comme suit :

Dans SGBD 1



Dans SGBD 2



Pas de spécification

I.7. Différentes classes de conflits : [BEN 2002] [CHR 1996]

I.7.1. Conflits de modèles de données : sont dus à la diversité des modèles utilisés dans la conception des différentes bases (relationnel, relationnel étendu, objet, relationnel objet, …).

I.7.2. Conflits de schémas : et regroupent :

a) – Conflits structurels : apparaissent lorsque le même élément du monde réel est perçu avec des structures différentes d'une base à une autre.

b) – Conflits de types de données : apparaissent lorsque deux bases de données utilisent deux types différents pour le même concept du monde réel.
Une solution pour ce type de conflits consiste à effectuer des conversions de type.

c) – Conflits d'organisation et de description des données : sous forme de classes et de relations.

d) – Conflits de généralisation/spécialisation : base sur la notion d'une relation généralisée ou spécialisée

e) – Conflits de représentation des formes géométriques : (au niveau des SIG)

I.7.3. Conflits sémantiques : sont dus au fait que des données présentes sur plusieurs systèmes admettent des interprétation différentes en fonction du contexte local d'utilisation, ils regroupent :

a) – Conflits de nommage : touchent essentiellement des entités et des attributs, et consiste à affecter des noms différents pour le même concept (problème des synonymes), ou affecter le même nom à des concepts différents (problème des homonymes).
Généralement on résout ce type de conflits par renommage.

b) – Conflits de codification de valeurs : Différence d'échelles utilisées, d'unités de mesure, d'intervalles de valeurs affectées aux différents attributs, … etc.

c) – Conflits de localisation des objets spatiaux (les SIG) : différents systèmes de projection, type de positionnement direct ou relatif, précision de localisation, … etc.

I.8. Exemples de conflits de données :

Exemple 01 : Soient les deux bases de données : UNIV-A, UNIV-C définies comme suit :

UNIV-A :

Info-Math

Catégorie	Dépt	Salaire
Prof	Computer science (CS)	65000
Prof-associé	CS	50000
Technicien	CS	45000
prof	Math	60000
Prof-associé	Math	55000
Technicien	Math	45000

UNIV-C

CS (Computer science)

Catégorie	Salaire
Prof	60000
Prof-associé	55000
Technicien	42000

Math

Catégorie	Salaire
Prof	70000
Prof-associé	60000
Technicien	46000

On constate des conflits de schémas entre UNIV-A , UNIV-C
Il s'agit exactement de conflits d'organisation et de description des données.

Exemple 02 : Soient les deux bases de données : UNIV-B, UNIV-D définies comme suit :

UNIV-B :

Info-Math

Catégorie	CS	MATH
prof	55000	65000
Prof-associé	50000	55000
Technicien	43000	44000

UNIV-D .

Info-Math

Dept	prof	Prof-associé	Technicien
CS	75000	60000	40000
MATH	60000	45000	38000

Comme l'exemple précédent, on constate des conflits de schémas entre UNIV-B, UNIV-D. Il s'agit exactement de conflits d'organisation et de description des données.

Exemple 03 : Soient les deux sites S1, S2

S1 définit un seul concept soit « Route » pour représenter tous les types de routes.
S2 utilise 02 concepts pour distinguer les entités {Autoroute, Route départementale} liés à un concept plus général « route ».

Dans cet exemple on a un conflit de schéma (de généralisation/spécification)

I.9. Différents niveaux d'interopérabilité : [BEN 2002]

On classifie l'interopérabilité des systèmes d'informations en plusieurs niveaux :

I.9.1. Niveau plate-forme (le niveau faible d'interopérabilité) : un système d'informations est interopérable au niveau plate-forme s'il fournit un accès aux fichiers de données stockées sur différents systèmes (possibilité d'accès aux fichiers d'informations quelques soient les plates-formes utilisées).

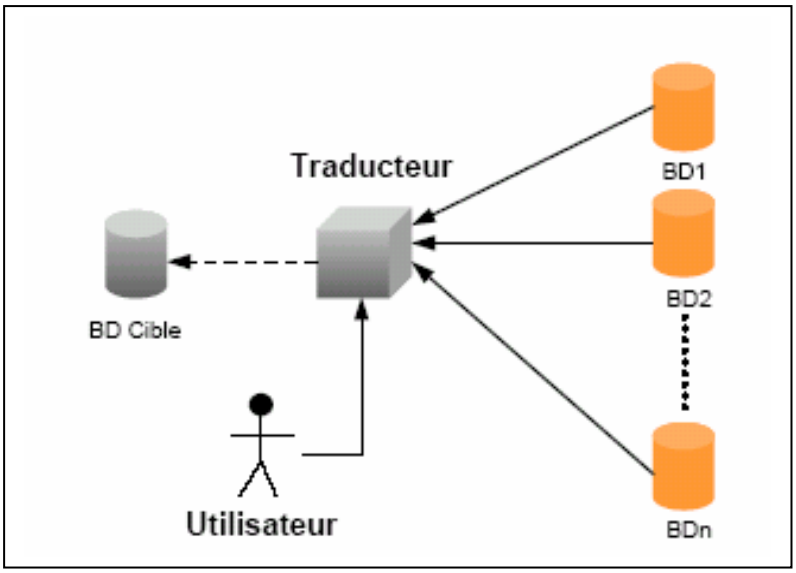


Fig I.9. Interopérabilité de niveau plate-forme

Anomalies :

- Les outils de traduction sont complexes et nécessite un spécialiste .
- Les accès aux sources des données ne sont pas transparents.
- Les conflits de données sont à la charge de l'utilisateur.

Vu aux anomalies pré-citées, le niveau d'interopérabilité dit de plate-forme ne traite qu'une partie des problèmes d'interopérabilité.

I.9.2. Niveau syntaxique : Ce niveau d'interopérabilité permet de fournir un accès uniforme à plusieurs sources de données en utilisant un modèle et un langage communs.

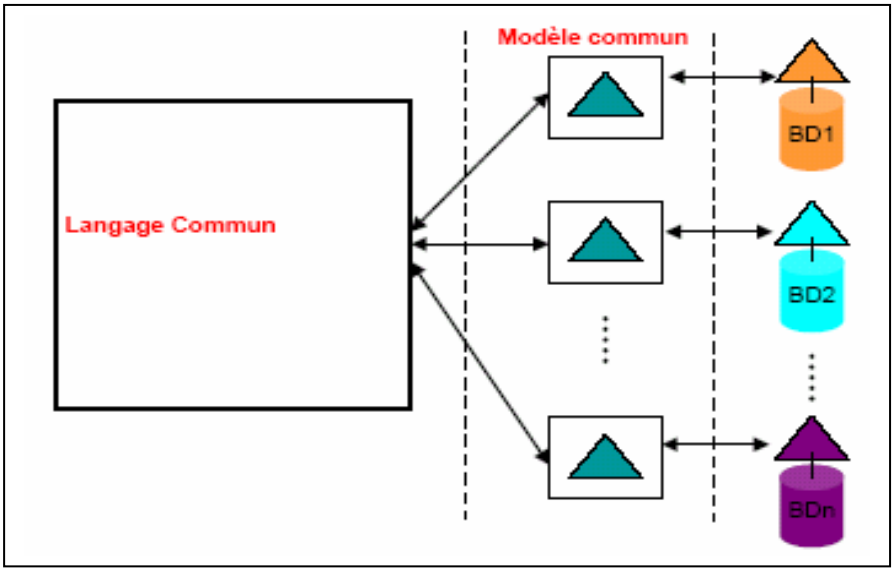


Fig I.10. Interopérabilité de niveau syntaxique

Caractéristiques et anomalies :

- L'approche est extensible.
- Les accès sont transparents, car les modèles de données utilisés par les différents sources de données sont unifiés par un modèle commun.
- La localisation des sources n'est pas transparente.
- Les conflits de schéma et les conflits sémantiques restent toujours à la charge de l'utilisateur (l'intervention de l'administrateur de bases de données est indispensable).

I.9.3. Niveau application/sémantique : Fournir un accès transparent à plusieurs sources de données .

a) - Solution basée sur fédération :

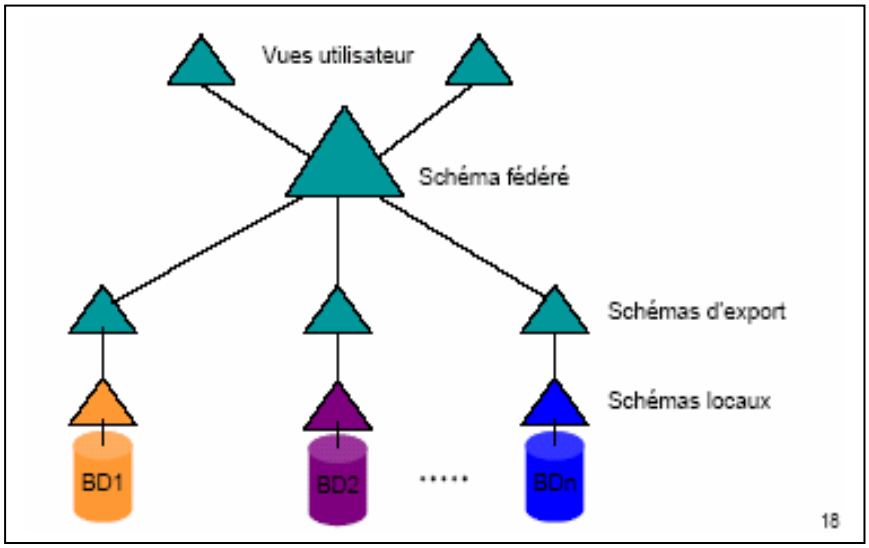


Fig I.11. Interopérabilité de niveau application/sémantique – approche fédération

b) - Solution basée sur médiation (approche LAV) :

Cette solution base essentiellement sur deux composants :

1 – Un composant médiateur (Mediator) : caractérisé par :
- Une vue globale
- Traitement des requêtes multi-sources.

2 – Un composant adaptateur (Wrapper) : caractérisé par :
- Vue mono-source
- Traitement des requêtes mono-source

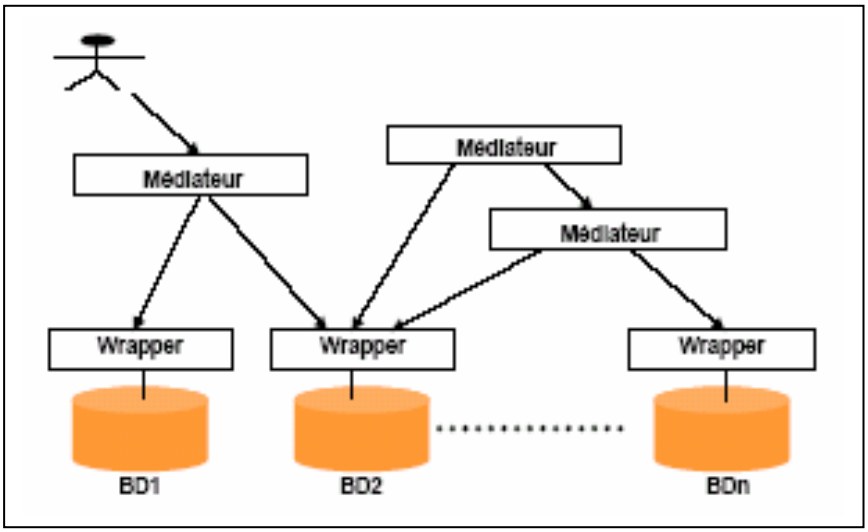


Fig I.12. Interopérabilité de niveau application/sémantique – approche de médiation LAV

c) - Solution basée sur médiation (approche GAV) :

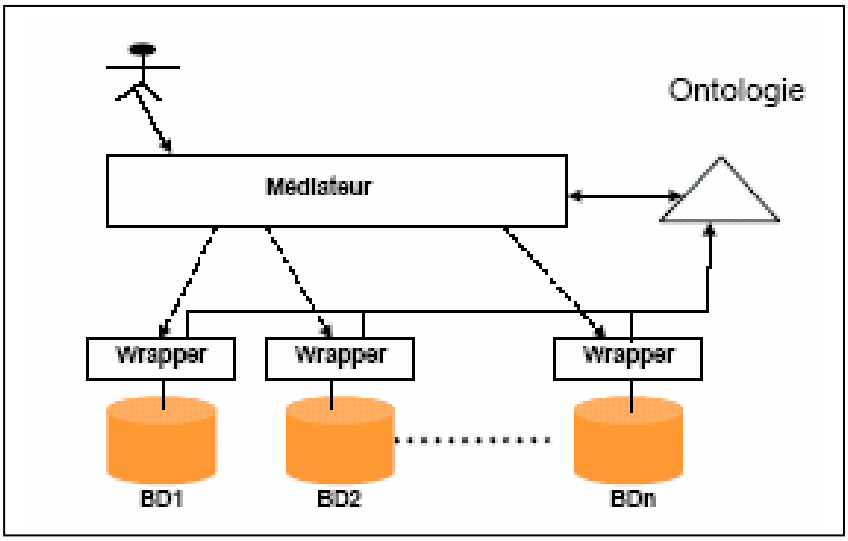


Fig I.13. Interopérabilité de niveau application/sémantique – approche de médiation GAV

Conclusion :

- Les approches de fédération et de médiation LAV sont peut adoptées aux environnements dynamiques car elles relient statiquement les schémas exportés par les sources de données .
- L'approche de médiation GAV paraît convenable mais pose des problèmes d'évolutivité du système.

d) - Solution basée sur médiation de contexte :

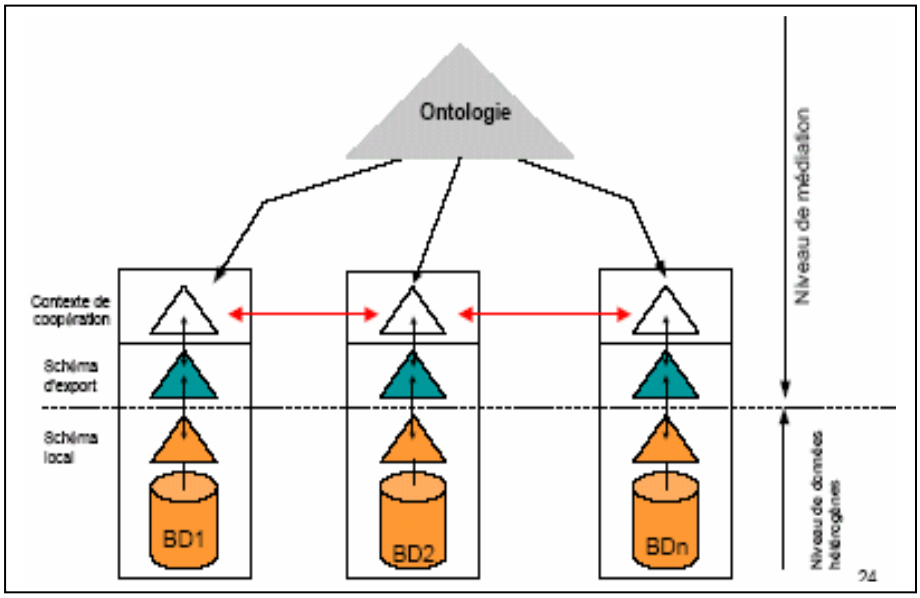


Fig I.14. Interopérabilité de niveau application/sémantique – approche de médiation de contexte

I.10. *Intégration des bases de données :* [GAR 2001] [BEN 2002][CHR 1996]

I.10.1. Définition : Par définition les bases de données contiennent des représentations d'objets d'un monde réel avec leurs propriétés et leurs liens entre eux. Ces représentations peuvent être diverses d'une base de données à une autre (problème de l'hétérogénéité).

L'intégration des données consiste tout simplement à permettre un accès cohérent à des données : d'origine, de structuration, et de représentation différentes.

Autrement dit, l'intégration des bases de données dépasse les représentations pour considérer en premier lieu ce qui est représenté (les objets) plutôt que comment il est représenté.

Le processus d'intégration des bases de données s'articule autour les taches principales suivantes :

1 – Traitement de l'hétérogénéité sémantique : en effectuant une analyse sémantique comparée des données, préalablement à l'intégration. Cette analyse est souvent groupée avec la phase de traduction.

2 –Traduction des schémas : pour résoudre le problème de l'hétérogénéité syntaxique en transformant tous les modèles vers un modèle unique qui est le modèle pivot ou le modèle canonique. Cela exige que :

- chaque source possède un traducteur local/canonique.
- Chaque traducteur d'une source locale doit pouvoir réaliser trois conversions :
  - Schéma local ⟶ Schéma équivalent en modèle canonique (le schéma intégré).
  - Données locales⟶ Données équivalentes en modèle canonique.
  - Requêtes en langage du modèle canonique⟶ Requêtes équivalentes en modèle local.

3 – Intégration des schémas : s'articule à son tour autour :

- identification des éléments de base qui sont liés (Définir les correspondances entre les éléments des différentes bases candidates).
- Choisir la représentation la plus adéquate qui peut être soit un modèle EA, un modèle relationnel, un modèle objet, ou aussi un modèle mixte objet-relationnel (choix du modèle pivot).

I.10.2. *Objectifs de l'intégration :*

- Produire un schéma global (intégré) à partir de plusieurs schémas locaux.
- Représentation des méta-données qui envisagent les différentes correspondance entre le schéma global (intégré) et les schémas locaux en terme données et requêtes.

I.10.3. *Niveaux d'intégration :*

I.10.3.1. *Intégration des schémas :*

Une opération permettant d'intégrer des schémas conceptuellement différents .

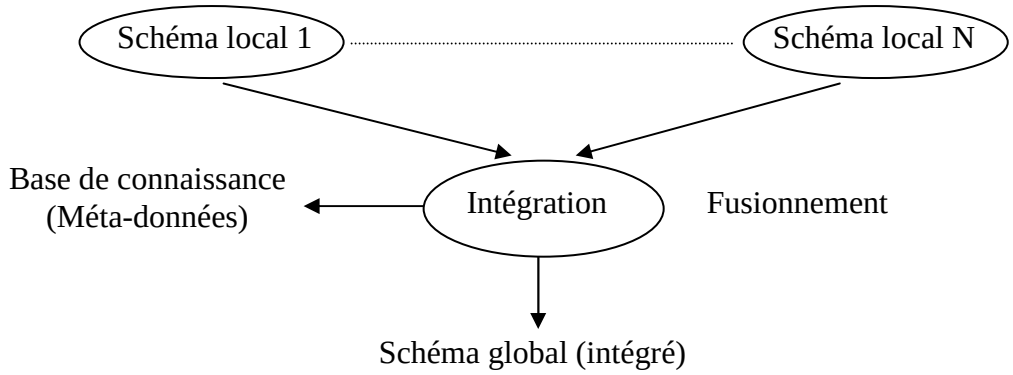


Fig I.15. Intégration des schémas

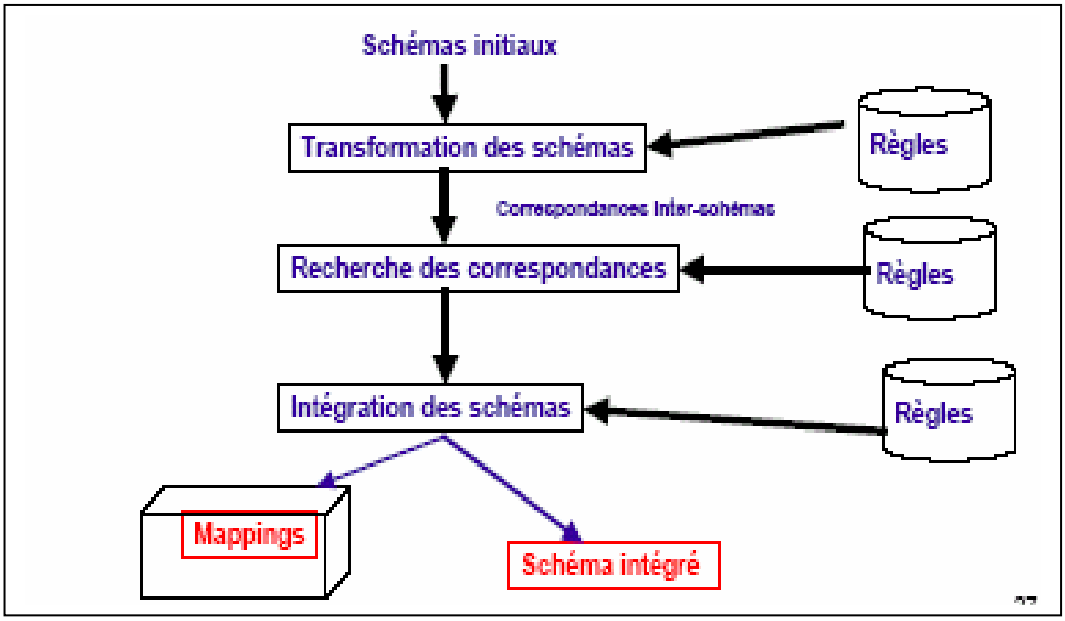


Fig I.16. Processus d'intégration des schémas (Différentes phases)

I.10.3.2. *Intégration des données :*

Processus semi-automatique permettant d'intégrer des données structurellement et sémantiquement hétérogènes.

I.10.4. *Problèmes rencontrés lors de l'intégration :*

1 – Conflits de différents types (d'abstraction, de classes, de types, de schémas).
2 – Le modèle pivot doit avoir un pouvoir de modélisation supérieur à ceux des modèles des bases de données composantes (lors de la traduction).
3 – Nécessité de compléter sémantiquement des modèles de bases de données composantes qui seraient trop pauvres.

I.10.5. Démarche d'*intégration :*

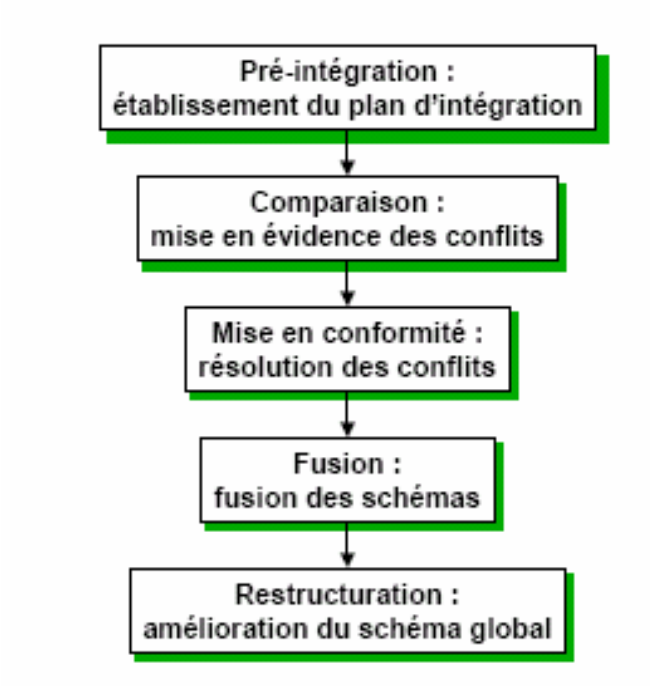


Fig I.17. Démarche d'intégration

Le processus d'intégration des bases de données s'effectue suivant la démarche suivante (voir figure précédente) :

1 – Pré-intégration : Comprend :

- ☒ L'extraction d'un maximum d'informations sur les différents éléments composants un schéma (attributs, classes, méthodes, …).
- ☒ La définition des relations et des règles entre éléments, ainsi que les équivalences entre domaines .
- ☒ La définition des contraintes globales .
- ☒ L'analyse des schémas de données pour mettre en évidence les dépendances induites par ces schémas.
- ☒ L'établissement d'un plan d'intégration.

2 – Comparaison : Comprend :

- ☒ La définition des synonymes, des homonymes, des types de données, des dépendances.
- ☒ L'identification des éléments communs entre les différentes bases de données, ainsi que les dépendances inter-schémas (inclusion, exclusion, union).
- ☒ L'identification des conflits.
- ☒ La définition des dépendances inter-schémas (inclusion, exclusion, union).

3 – Mise en conformité : consiste à :

- ☒ Etablir une certaine conformité entre les différents schémas initiaux.
- ☒ résoudre tous les conflits rencontrés :
  - Par renommage pour les conflits des noms.
  - Etude cas par cas pour les conflits structurels.

4 – Fusion des schémas en un schéma unique : c'est à partir de ce point que commence le processus d'intégration proprement dit qui consiste à construire un schéma intégré (schéma global), le schéma obtenu doit remplir les caractéristiques

suivantes :

- ☒ Complétude : pas de perte d'informations.
- ☒ Minimalité : pas de redondance.
- ☒ Clarté : aucune ambiguïté.

L'intégration des schémas ainsi que des données se réalisent suivant l'une des deux architectures suivantes : **[GAR 2001]**

a) - Schéma global (GAV : Global As View) : un schéma global est défini comme une vue intégrante sur les schémas locaux, il s'agit donc d'une approche ascendante depuis les sources vers le médiateur.
Parmi les systèmes qui utilise cette approche on cite : TSIMMIS, SIMS, GARLIC, …

Caractéristiques de schéma GAV :

- ☒ La qualité du système utilisant un schéma GAV dépend de comment les sources sont intégrées pour construire le schéma global .
- ☒ Lorsqu'une source se change ou une nouvelle source est ajoutée et doit participer au système, le schéma GAV doit être mis à jour et la phase de réécriture est très simple .

Inconvénient :

- ☒ L'ajout d'une nouvelle source de données entraîne la modification du schéma global

b) – Schéma local (LAV : Local As View) : Dans un schéma local chaque source est définie comme une vue locale du schéma global, il s'agit donc d'une approche descendante depuis le médiateur vers les sources .
parmi les système qui utilise cette architecture on cite : Information Manifold .

Caractéristiques du schéma LAV :

- ☒ La qualité du système utilisant un schéma LAV dépend de comment les sources sont caractérisées .
- ☒ Si le schéma global est bien spécifié à priori, la modification ou l'ajout d'une nouvelle source n'entraîne pas sa modification .
- ☒ La phase de réécriture est plus complexe .

Inconvénient :

- ☒ Difficulté de réconcilier source et vue locale.

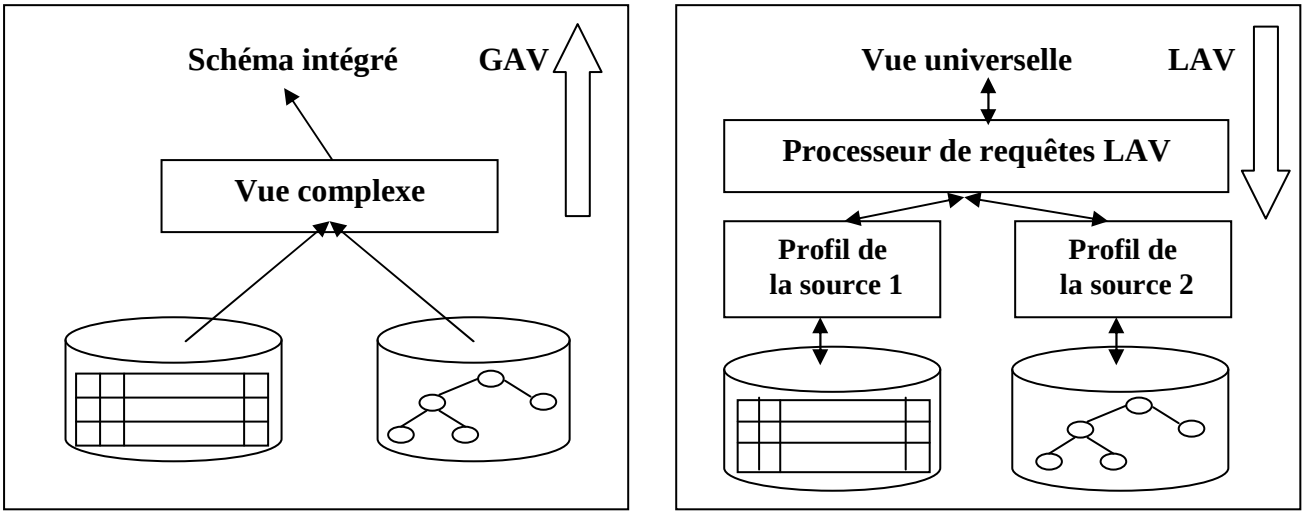


Fig I.18. Les architectures GAV et LAV

5 – Restructuration : consiste à améliorer et optimiser le schéma global obtenu.

I.10.6. Résolution des conflits lors de l'intégration : [BEN 2002]

• ***Hypothèse :*** les schémas locaux sont homogènes en terme modèle de données .
• ***L'exemple traité :*** Soient les deux schéma conceptuels locaux S1, S2 d'une bibliothèque centrale définis comme suit :
S1 utilise le modèle EA, S2 utilise le modèle objet.

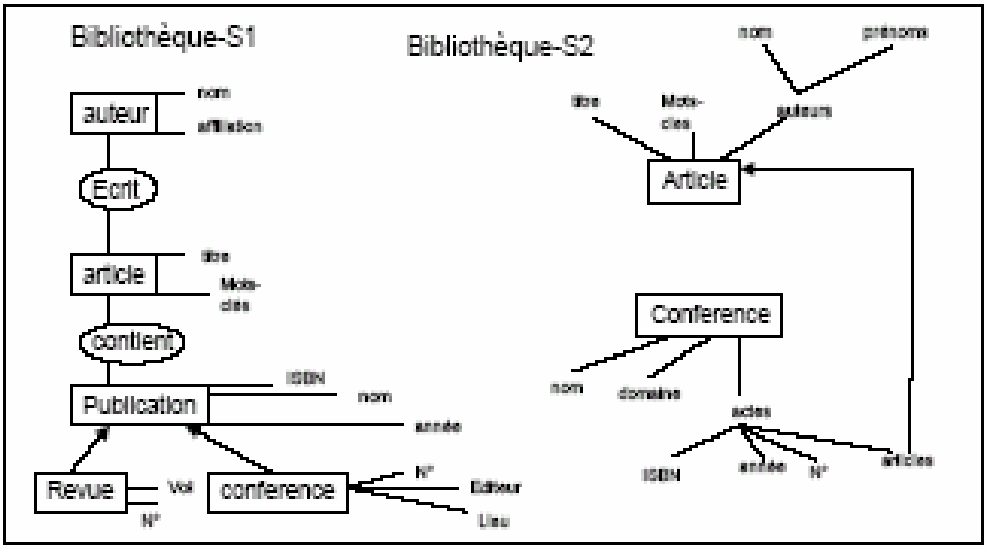


Fig I.19. Modélisation d'une bibliothèque

- ***Principaux conflits à résoudre :***

I.10.6.1. Conflits de classification : rappelons que les conflits de classification apparaissent lorsque les éléments en correspondance décrivent des éléments différents mais sémantiquement liés.

• ***Exemple :*** le schéma local S1 décrit les articles revues et conférences
le schéma local S2 décrit les articles conférences

Donc, il existe un conflit de classification entre S1 et S2

- ***Solution :*** Pour résoudre les conflits de classification on doit :

a) - Identifier les correspondances inter-schémas en utilisant une notation spécifique.
b) - Identifier les instances mises en correspondance .

a) - un format général est fréquemment utilisé pour désigner des correspondances inter-schémas entre 02 éléments A et B :

$$f(A)\ R\ g(B)$$

avec : R est une relation ensembliste : d'équivalence ($\equiv$), différence ($\neq$), inclusion ($\supseteq$), intersection ($\cap$).

F, g deux fonctions optionnelles utilisées pour résoudre un problème du à la représentation.

b) - Pour identifier les instances mises en correspondance, on utilise deux clauses :

**** La clause AIC :*** Cette clause est utilisée pour mentionner que deux objets appartenant à deux bases de données différentes ont un identifiant commun.

Exemple : Dans notre exemple de la bibliothèque centrale on a :

S1.Article R S2.Article AIC Titre

Cela veut dire qu'il y a une correspondance entre les deux schémas S1, S2 et que les articles des deux bases de données ont un identifiant commun (l'attribut titre).

**** La clause AAC :*** Utilisée pour spécifier les attributs en commun entre deux éléments pour éviter toute duplication d'attributs au niveau du schéma intégré.

Exemple : Dans l'exemple précédent on a :

S1.Conférence R S2.Conférence.Actes AIC ISBN, AAC Année, N°

Ça veut dire que les deux éléments conférence et actes conférences des deux bases de données de schémas S1, S2 ont un identifiant commun (ISBN), et deux attributs en communs (année, N°).

Exemple général : on prend l'exemple précédent (la bibliothèque – Fig I.19):

Et on essaye de tirer toutes les correspondances possibles entre les deux schémas locaux S1, S2 , on obtient alors :

Auteur ⊇ Auteur AIC Nom
Article ⊇ Article AIC Titre AAC mots-clés
Conférences ≡ Actes AIC ISBN AAC année, N°
Publication.Nom ⊇ Conférence.nom AIC nom
Auteur-Ecrit-Article ≡ Auteur-Article
Conférence-contient-article ≡ conférences.actes.articles …

I.10.6.2. Conflits structurels : Apparaissent lorsque les éléments en correspondance sont décrits par des concepts de niveaux différents.

- ***Exemple :*** La classe auteur dans S1 et l'attribut auteur dans S2.

- ***Solution :*** La solution des conflits structurels part du principe que l'élément intégré (obtenu par intégration de plusieurs autres éléments) doit refléter les possibilités des éléments initiaux, en terme capacité de décrire l'information, et pour cela on adopte la borne supérieure minimale, et en terme des contraintes implicites et explicites attachées aux éléments, et pour cela on adopte la borne inférieure maximale.

  Par exemple si on a une relation qui est en correspondance avec un attribut, alors les éléments correspondants qu'on va prendre dans le schéma intégré sera une relation.

  De même, si on a une classe d'objets qui est en correspondance avec un attribut, c'est en fin la classe d'objets qui sera prise dans le schéma intégré.

  Notons à la fin que le problème des conflits, ainsi que leurs solutions seront traités avec plus de détail dans l'approche des bases de données fédérées.

I.11. Conclusion :

L'interopérabilité est devenue une nécessité pour répondre aux besoins d'échange d'informations entre systèmes d'informations hétérogènes et répartis. Elle traduit en clair la capacité d'un système d'informations à se collaborer avec d'autres systèmes de natures par fois très différentes.

Au long du chapitre « ***Etat de l'art*** », nous avons présenté tous les concepts, les notions de base, et les principes fondamentaux invoqués par une telle interopérabilité entre systèmes hétérogènes, à noter :

- La définition de l'interopérabilité.
- La définition et les propriétés des systèmes interopérables.
- Les motivations et les objectifs de l'interopérabilité des systèmes hétérogènes d'une manière générale et les bases de données en particulier.
- Les causes pour lesquelles se pose le problème d'interopérabilité entre bases de données.
- Les différents niveaux d'interopérabilité.

En deuxième lieu, nous avons traité le cœur du problème de l'interopérabilité qui est la recherche d'une solution efficace et générale.

La solution idéale pour permettre un accès cohérent à des données : d'origine, de structuration et de représentation différentes est l'intégration de ces données, en terme de schémas et de données. Cette intégration s'articule autour les taches principales suivantes :

- Le traitement de l'hétérogénéité (syntaxique, structurelle, sémantique)
- La traduction des modèles locaux représentant les différentes sources de données en un modèle commun appelé modèle pivot ou modèle canonique.
- L'Intégration des différents schémas locaux en un schéma unifié appelé le schéma global ou intégré.

Le processus d'intégration tente de rencontrer plusieurs conflits de différents types (d'abstraction et de modélisation, de classes, de types, de schémas, de description et d'organisation, …etc), ce qui rend la résolution de tous ces conflits au cœur du processus d'intégration.

L'intégration est une opération semi-automatique progressif qui peut être décomposé en plusieurs phases, dont les principales sont :

- ***La phase de pré-intégration :*** qui consiste à extraire le maximum d'informations relatifs aux éléments composant un schéma, ainsi que les relations et les règles existantes entre eux.
- ***La phase de comparaison :*** qui consiste à chercher les éléments en commun entre les différentes bases de données, et des correspondances inter-schémas.
- ***La phase de mise en conformité :*** qui consiste à établir une certaine conformité entre les différents schémas initiaux et résoudre tous les conflits rencontrés.
- ***La phase de fusion :*** qui consiste à construire un schéma intégré à partir des schémas initiaux.
- ***La phase de restructuration :*** consistant à améliorer et optimiser le schéma global obtenu lors de la phase précédente (la phase d'intégration).

Il faut noter qu'un schéma intégré doit remplir trois critères essentiels :

la complétude, la minimalité, et la clarté.

Concernant la façon de réaliser cette intégration de bases de données et les exigences matérielles et logicielles, il existe plusieurs approches qui seront présentées en détail dans le prochain chapitre.

Chapitre 2

LES DIFFERENTES APPROCHES D'INTEROPERABILITE DES BASES DE DONNEES HETEROGENES ET REPARTIES

II.1. Préface : Après avoir exposer les différentes notions de base liées à l'interopérabilité des bases de données hétérogènes et réparties dans le premier chapitre, nous allons présenter dans le présent chapitre les différentes approches qui servent à garantir cette interopérabilité en donnant pour chaque approche son principe, ses fondements, et surtout ses avantages et ses inconvénients.

II.2. Interopérabilité par transfert de fichiers : [GAR et ALL 2002]

C'est le mode (l'approche) le plus élémentaire de l'interopérabilité, il consiste à transférer des fichiers entre deux systèmes applicatifs.

Dans ce cas, les données subordonnées au 'système A' peuvent être transmises pour traitement spécifique au 'système B', selon un format convenu de données.

Le vecteur de transfert du fichier (le support) peut être :

- La fonction de transport d'un protocole de communication (exemple : FTP sous TCP/IP)
- La messagerie (associer le fichier à transférer à un message)

II.3. Interopérabilité en temps réel : [GAR et ALL 2002]

S'applique à toutes les situations ou l'utilisateur doit accéder sans le moindre délai d'attente à des informations dispersées dans plusieurs systèmes applicatifs, par l'intermédiaire d'un programme spécifique (programme appelant).

Le programme appelant élabore basant sur les indications de l'utilisateur une requête dont les composants attribués peuvent résider dans plusieurs systèmes applicatifs.

La requête est transmise aux systèmes applicatifs concernés, qui après traitement retournent l'information au programme appelant.

Le mécanisme de dialogue avec les systèmes applicatifs utilise des fonctions de synchronisation inter applicatives (tache à tache), elles mêmes basées sur des fonctions de transport, fournies par les protocoles de communication.

Si le programme appelant et les systèmes applicatifs sont de même type, les mécanismes de dialogue peuvent être inclus dans les SGBDs comme par exemple « XXX-NET » sous INGRES et ORACLE.

Si les environnements impliqués sont hétérogènes, la recherche de solutions particulières (pour l'interopérabilité) s'imposent, en attendant une standardisation.

II.4. Interopérabilité en temps différé : [GAR et ALL 2002]

S'applique à toutes les situations ou l'utilisateur peut supporter un délai d'attente pour accéder à des informations dispersées dans plusieurs systèmes applicatifs à partir d'un programme spécifique (programme appelant).

Le mécanisme est le même que celui de l'interopérabilité en temps réel sauf que la possibilité de différer les échanges dans le temps autorise toute fois l'utilisation de fonctions de type messagerie, basées sur les fonctions de transport fournies par les protocoles de communication.

II.5. Interopérabilité par médiation (Approche de médiation de schémas) : **[GAR&ALL 01]**

II.5.1. Principe : En court terme, l'approche de médiation consiste à utiliser un médiateur et un ensemble d'adaptateurs chacun correspond à une source de données pour faciliter l'accès aux bases de données hétérogènes.

Un médiateur comprend un schéma global ou ontologie dont le rôle est central, c'est un modèle du domaine de l'application du système. L'ontologie fournit un vocabulaire structuré servant comme support pour exprimer des requêtes.

Dans cette approche, l'intégration de l'information est fondée sur l'exploitation de vues abstraites qui décrivent de façon homogène et uniforme le contenu des sources d'informations dans les termes de l'ontologie. Les sources d'informations pertinentes sont calculées par réécriture de la requête en terme de ces vues afin de répondre à cette requête. Le problème consiste donc à réécrire la requête en fonction des différentes vues.

L'approche de médiation présente l'avantage de pouvoir construire un système d'interrogation des sources de données sans toucher aux données qui restent toujours stockées dans leurs sources d'origine d'une façon distribuée. L'interrogation effective des sources se fait via des adaptateurs (Wrappers) qui traduisent les requêtes réécrites en terme de vues dans le langage de requêtes spécifique accepté par chaque source.

Parmi les grandes catégories d'applications qui adoptent l'approche d'intégration par médiation on cite :

1 – Les applications de recherche de l'information (moteur de recherche).
2 – Les application d'aide à la décision.
3 – Les application de gestion de connaissances au sens large.

L'approche de médiation s'appuie sur deux principes de bases :

1. Définition d'un langage commun par lequel le médiateur interrogera les adaptateurs
2. utiliser un format commun pour les résultats par lequel les adaptateurs répondront au médiateur, il s'agit souvent d'un langage propriétaire ou standardisé .

II.5.2. Schéma général de médiation : **[GAR 2001]**

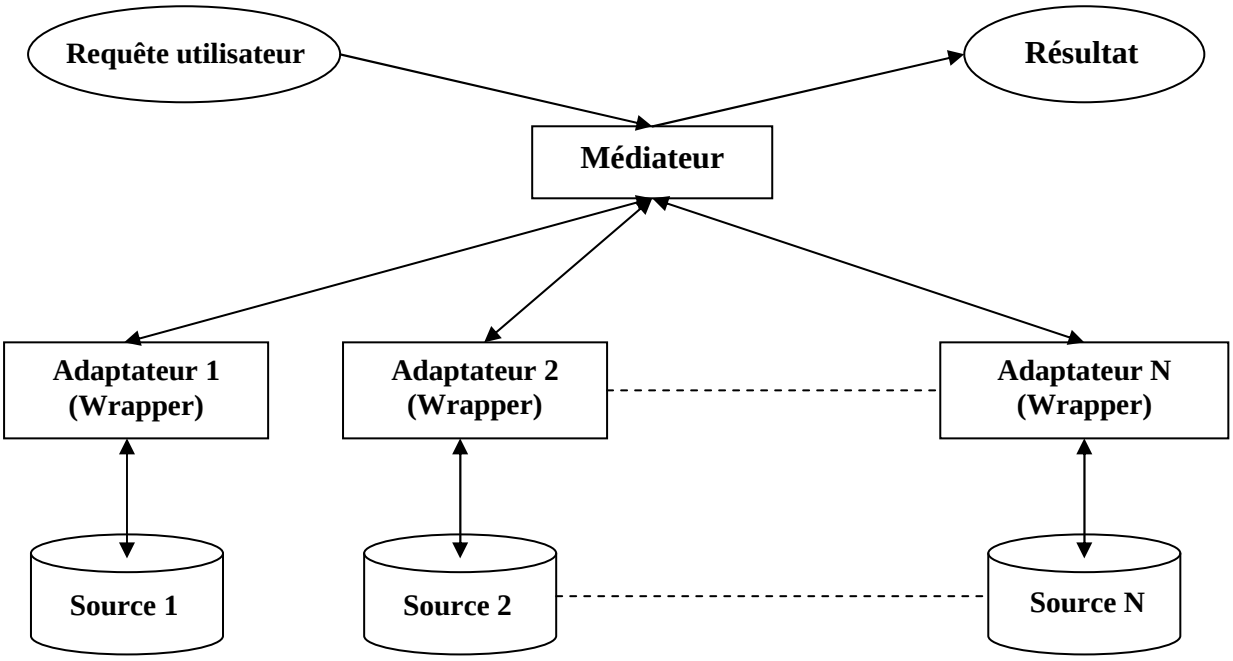


Fig II .1. Schéma général de médiation

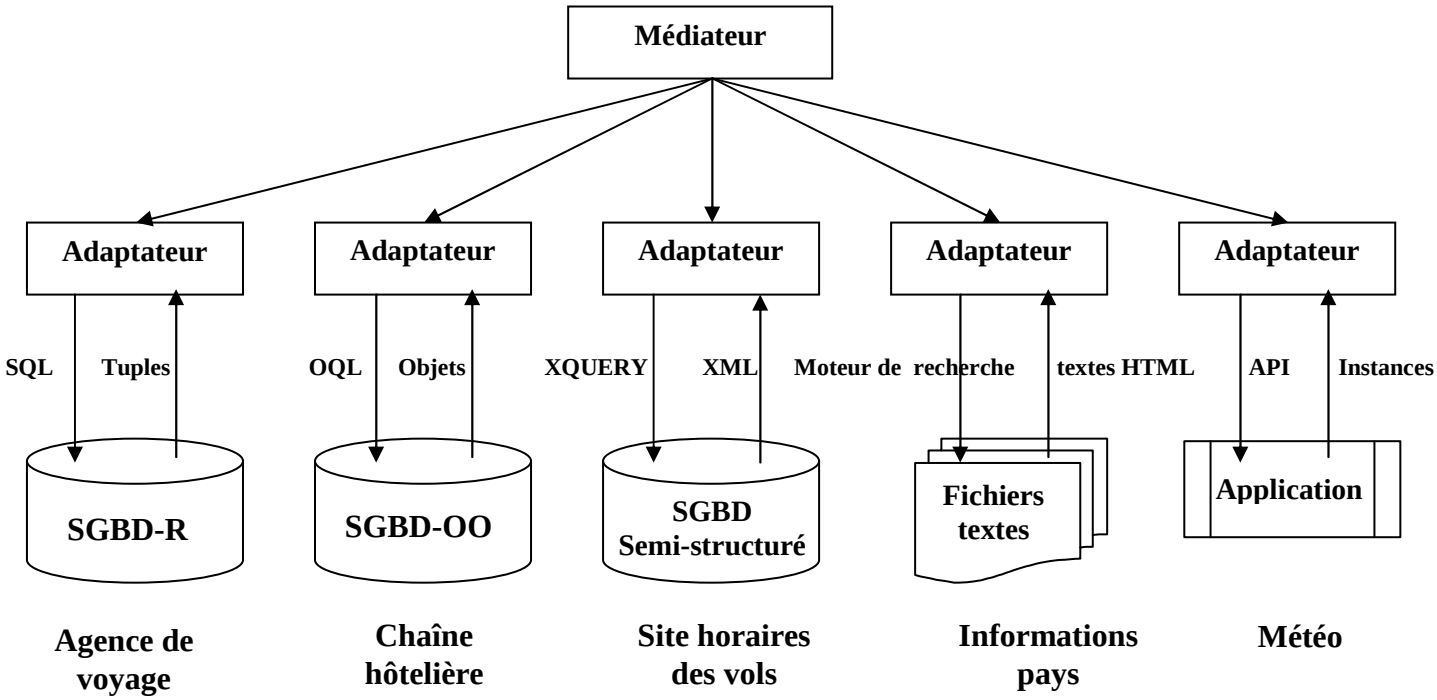


Fig II .2. Schéma détaillé de médiation

II.5.3. Avantages de médiation :

- ☒ Support de l'hétérogénéité des sources de données.
- ☒ Découvrir les sources pertinentes hétérogènes et transparence à la localisation de leurs données pour l'intérêt des applications.
- ☒ Aider à accéder ces sources en évitant à l'utilisateur d'interroger lui même chacune d'elles selon leurs propres modalités et leurs propres vocabulaires.
- ☒ Disponibilité accrue des données en cas de pannes du serveur .
- ☒ Offrir à l'utilisateur une vue globale uniforme et centralisée des données.

II.5.4. Architecture de médiation :

II.5.4.1. ***Médiateur (Mediator) :*** est un logiciel qui offre une interface unique et transparente à plusieurs bases de données, il résout avec efficacité le problème de distribution des sources.

- ***Rôle et fonctions :***

- ☒ Localisation des sources de données.
- ☒ Décomposition des requêtes pour les adapter à chacune des sources des données.
- ☒ Envoi des requêtes aux différentes sources .
- ☒ Recomposition des différents résultats provenant de chacune des sources .
- ☒ Interprétation des correspondances et des conflits.

- ***Panorama des médiateurs existants :*** **[HAC 2003]**

1) - Génération relationnelle (1975 – 1990) : souvent centrée sur un SGBD qui joue le rôle d'un médiateur.

Exemple : SDDI, SIRIUS DELTA, R*, INGRES-STAR, MERMAID, MULTIBASE, MSQ.

2) - Génération relationnelle étendue (1990 – 2000) : Fédère des bases de données hétérogènes autour SQL3

Exemples : médiateurs objets : OLE-DB, PEGASUS, IRO-DB
Médiateurs XML : medience server, information integrator

3) - Génération XML, XQUERY (> 2000) : utilise le standard XML

Exemples : OLE-DB.NET (Microsoft), Nimble, Xquark, Fusion, Liquida Data, (BEA), Enosys Software, IBM Xperanto, Mediance (INIRIA), IBM Information, Integrator, Denodo

- ***Déroulement de médiation***

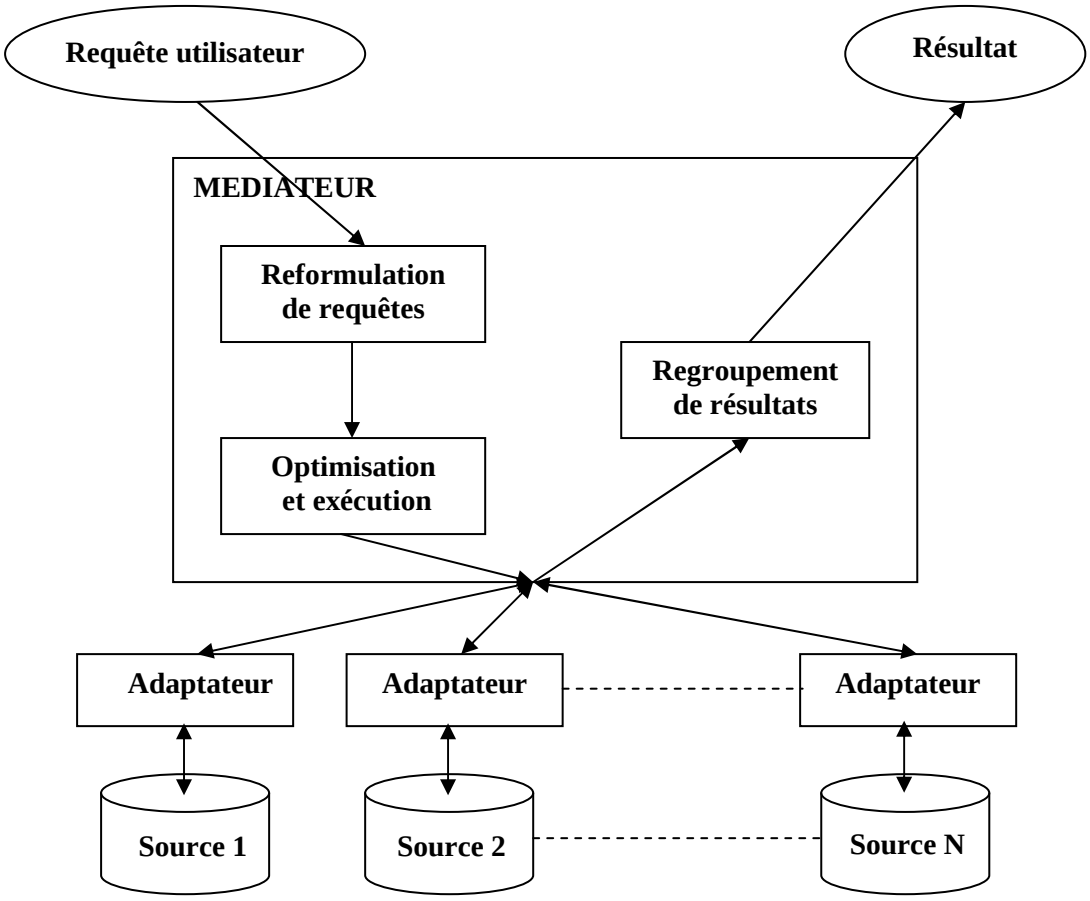


Fig II .3. Déroulement de médiation

a) - ***Médiation guidée par les sources :***

1 - Approche globale (GAV : Global As View) : provient du monde de bases de données fédérées et consiste à définir le schéma global (au niveau du médiateur) en fonction des schémas locaux des divers sources de données à intégrer**.** **[GAR 2001] [HAC 2003]**

Comme exemple des systèmes qui utilisent cette approche on cite : HERMES, TSIMMIS, MOMIS, … etc.

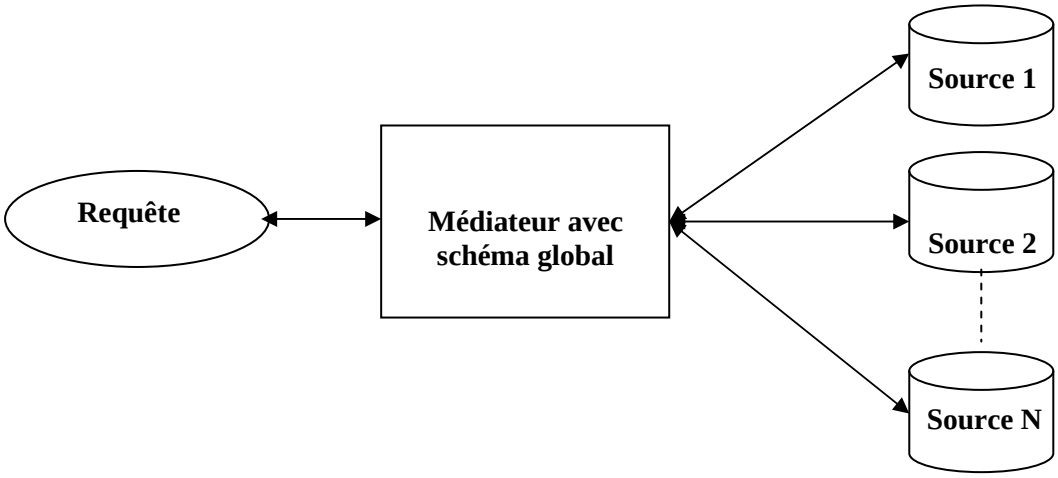


Fig II .4. Médiation guidée par les sources – Approche globale

Avantages de GAV :

1 – La construction des réponses à des requêtes dans un système adoptant l'approche GAV est simple, il suffit de remplacer les prédicats du schéma global de la requête par leur définition.
2 – Lorsqu'une source de données se change, ou une nouvelle source est ajoutée et doit participer au système, il suffit de mettre à jour le schéma GAV et la phase de réécriture dans ce cas est très simple.

Inconvénient :

L'ajout d'une nouvelle source, entraîne la modification du schéma global.

2 - Approche locale (Local As View) : le médiateur représente une vue intégrée de plusieurs sources, et de même chaque source de données est décrite comme une vue partielle sur le schéma global du médiateur.

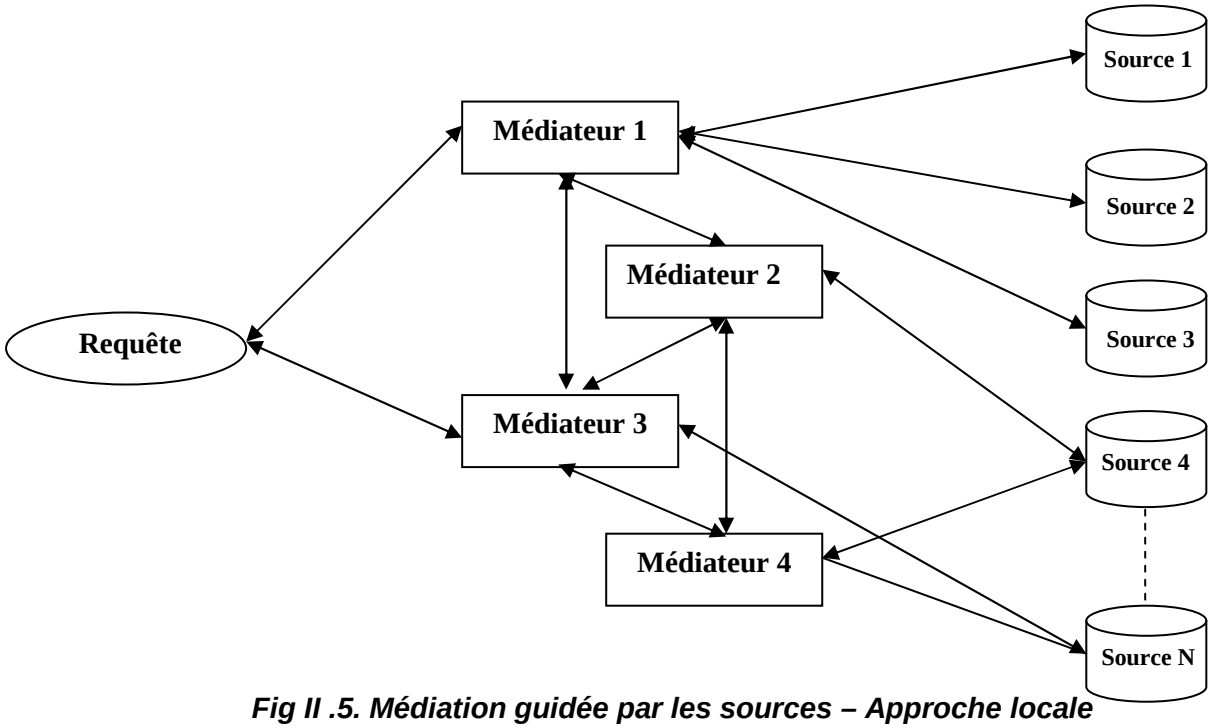


Fig II .5. Médiation guidée par les sources – Approche locale

L'approche LAV est adoptée dans les systèmes : Razor, Internet Softbot, Infomaster, Information manifold, Observer, Picsel, … etc

Avantages de LAV :

1 – En utilisant l'approche LAV, il est très facile d'ajouter une source de données, cela n' a aucun effet sur le schéma global (ouverture).

2 – La construction des réponses à des requêtes (réécriture des requêtes) est complexe.

3 – Il est aisé d'attacher des contraintes assez riches relatives au contenu des différentes sources (pouvoir d'expression).

Inconvénient :

Difficulté de réconcilier source et vue locale.

b) - Médiation guidée par les requêtes : lien direct entre sources et requêtes **[GAR 2001]**

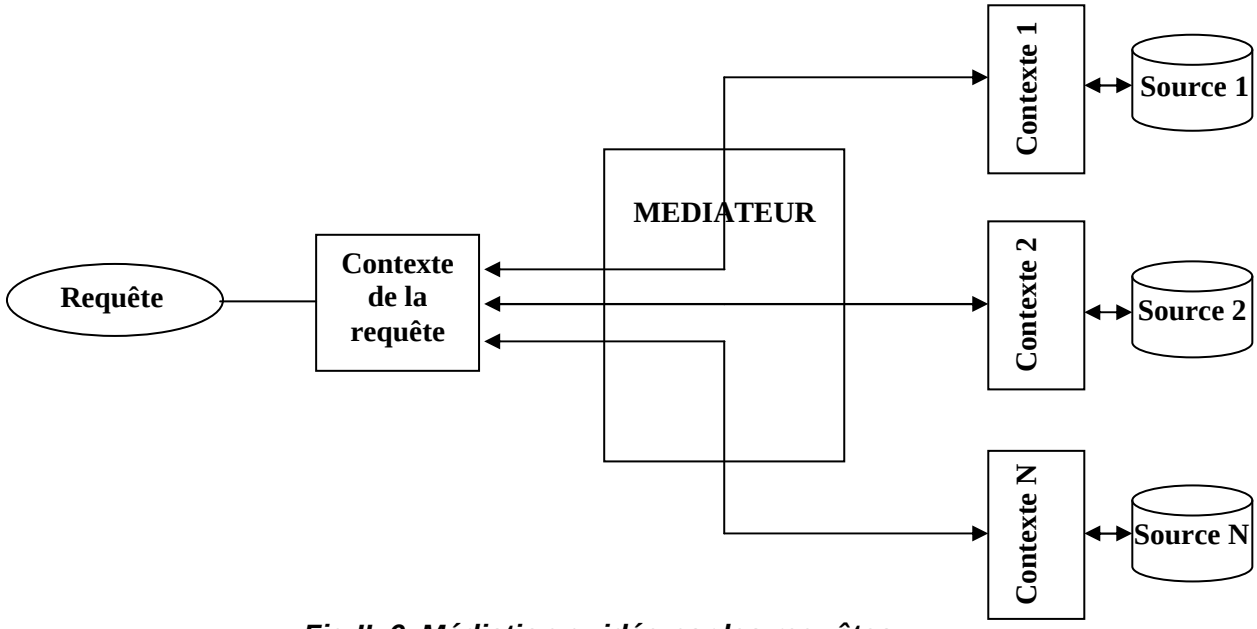


Fig II .6. Médiation guidée par les requêtes

II.5.4.2. ***Adaptateur (Wrapper) :*** est un logiciel qui convertit les données et les requêtes d'un modèle vers un autre modèle, il s'occupe donc de l'hétérogénéité des sources en offrant une interface homogène locale d'accès aux données

- ***Rôle et fonctions :***

☒ Traduction des requêtes du langage commun en langage natif propre à la source .
☒ Traduction des résultats du langage natif en langage commun.
☒ Simulation des contraintes et structures implicites

- ***Panorama des adaptateurs existants :***

1) - OLE DB Data Adapter : peut être utilisé avec n'importe quelle source de données exposée par le fournisseur OLE DB.

2) - SQL Data Adapter : propre à SQL Server, plus rapide que le précédent.

3) - ODBC Data Adapter : optimisé pour l'accès aux sources de données ODBC.

4) - Oracle Data Adapter : optimisé pour l'accès aux sources de données oracle.

II.5.5. Traitement des requêtes dans un système de médiation :

On distingue les phases suivantes :

- ☒ Analyse syntaxique et sémantique de requêtes.
- ☒ Décomposition de la requête globale en sous requêtes locales.
- ☒ Génération d'un plan optimisé pour l'exécution de la requête .
- ☒ Exécution des sous requêtes sur les différentes sources .
  - transformation de la requête du langage commun vers le langage de la source appropriée .
  - transformation du résultat du format de la source vers le format commun.
- ☒ Récupération des résultats intermédiaires et recomposition du résultat final .
- ☒ Combinaison des résultats locaux.
- ☒ Requête de recomposition sur système global .

II.5.6. Plan d'exécution des requêtes :

Un plan d'exécution décrit la méthode d'exécution d'une requête, il est souvent représenté par un arbre algébrique .

C'est un arbre ou les nœuds sont des opérateurs algébriques et les feuilles sont les sources de données.

Il peut exister une infinité de façon pour exécuter une requête ou chacune pourra être représentée par le plan d'exécution équivalent.

L'ensemble des plans d'exécution permettant de résoudre une requête est appelé espace de recherche . **[GAR 2001] [BEZ 2002]**

II.5.7. *Décomposition d'une requête :*

Exemple : chercher l'adresse de tous les personnes ayant des voitures vertes .

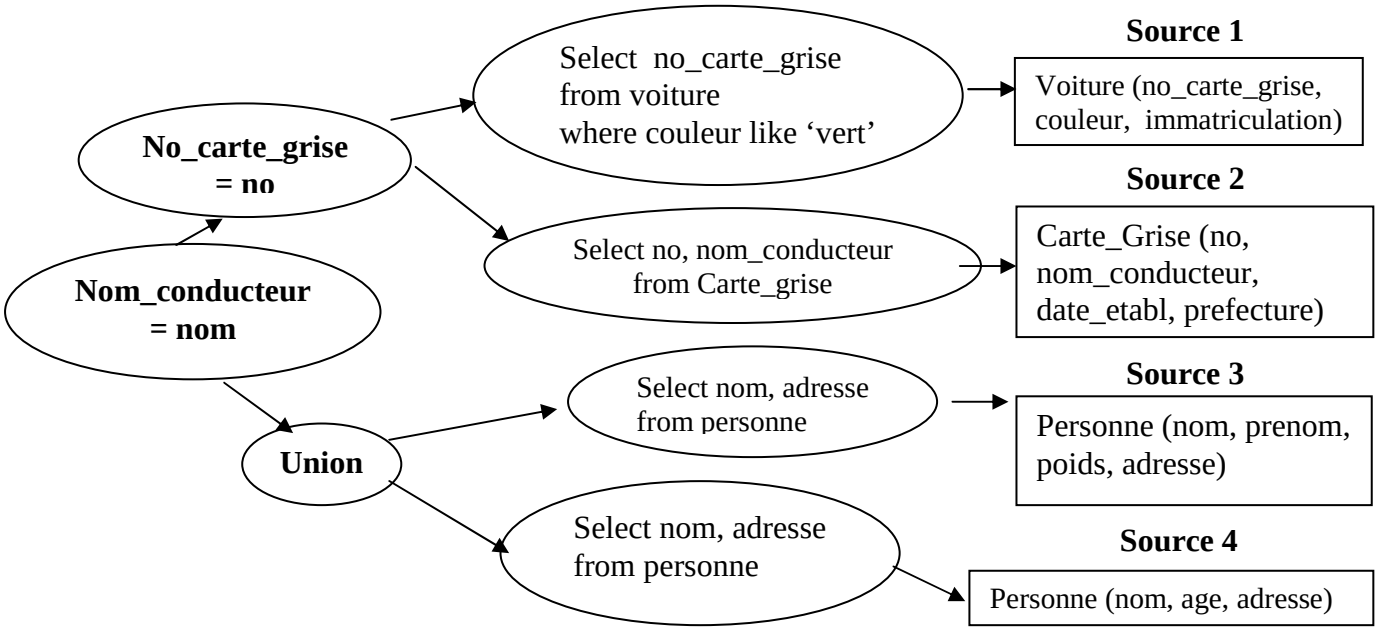


Fig II .7. Plan d'exécution d'une requête

II.5.8. *Puissance d'interrogation des sources :*

Les sources de données n'ont pas les mêmes possibilités d'interrogation et cela dépend de la source elle même :

- ☒ SGBD : possibilité de requêtes souvent complexes.
- ☒ Moteur de recherche : recherche par mots clefs.
- ☒ Fichier : recherche via un champs indexé.

Pour cela le médiateur ou l'adaptateur doit pallier aux déficiences de la source. A ce propos, on a deux approches pour l'implémentation des médiateurs et des adaptateurs :

- ***Implémentation basée sur les adaptateurs :*** implémentation complexe de chaque adaptateur de source, ce qui donne une intégration simple au niveau du médiateur.
- ***Implémentation basée sur les médiateurs :*** implémentation simple de chaque adaptateur source, ce qui donne une intégration complexe au niveau du médiateur cela nécessite aussi une communication des capacités de la source au médiateur.

II.5.9. *Optimisation et exécution des requêtes :*

On pense à l'optimisation des requêtes avant leur exécution car :

1- Les sources d'informations sont distribuées est autonomes (indisponibilité des statistiques, problème d'accessibilité, communication variée, …) .

2 – Les besoins des utilisateurs sur les traitements de requêtes sont différents

- Difficulté et diversification d'expression.
- Manque de connaissance sur les sources.

Et pour remédier à ces problèmes on donne les solutions suivantes :

- Optimisation et exécution dynamique de requête
- Spécification des contraintes d'évaluation, négociation dans le traitement pour raffiner les requêtes.

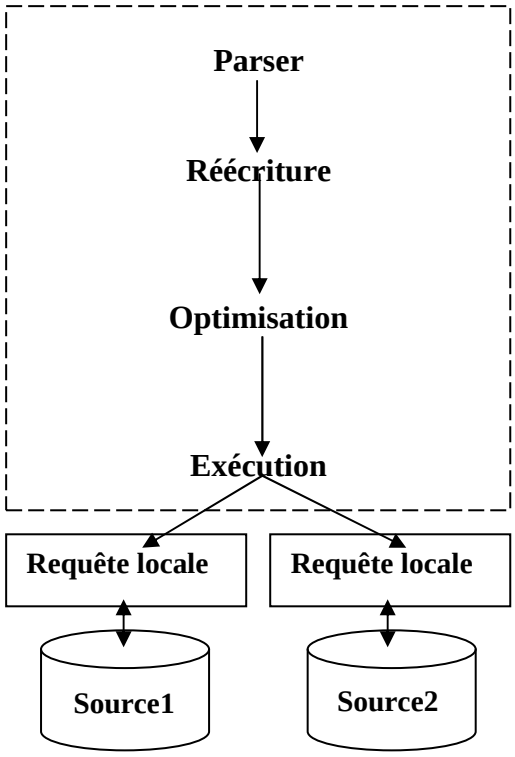


Fig II .8. Optimisation d'une requête

Il existe plusieurs méthodes d'optimisation de requêtes, parmi ces méthodes on cite :

- ☒ Ordonnancement des jointures
- ☒ Stratégie sur les jointures inter-sites
  - Par interrogation multiples d'une source avec les résultats du premier.
  - Par boucles imbriquées.
  - Par tri fusion .

a) - ***Optimisation statique de requêtes hétérogènes .***

- ☒ ajout de vues transitoires.
- ☒ Décomposition d'une requête.
- ☒ Simplification d'une requête.
- ☒ Prise en compte des capacités réduites des sources.
- ☒ Parallélisation d'une requête .
- ☒ Elaboration d'un plan d'exécution optimisé.

b) - ***Optimisation dynamique de requêtes hétérogènes.***

- ☒ Reformations dynamique du plan d'exécution .
- ☒ Prise en compte de sources indispensables.
- ☒ Ordonnancement dynamique des jointures.
- ☒ Optimisation adaptative.

II.5.10. Modèle de coût d'un plan d'exécution :

Un modèle de coût permet d'estimer le coût que prendra un plan d'exécution de requête.

Objectifs :

Choisir permet tous les plans d'exécution celui de coût minimal pour l'exécution de requête en appuyant sur des formules de coût, ainsi que des statistiques relatives au système :

- ☒ Système d'exploitation (CPU, E/S).
- ☒ SGBD (taille d'une page).

et des statistiques relatives aux données telles que les cardinalités des différentes collections, la sélectivité d'un attribut, … etc.

Et on distingue plusieurs modèle de coûts :

1) - Modèle de coût au niveau médiateur : **[GAR 2001]**

coût d'exécution d'une requête = coût-communication(1) + Opération médiateur(2) + coût sur les adaptateurs(3) + congestion du réseau(4).

Avec :

(1) : dépend du débit, de taille des données à transférer .
(2) : Formule classique de coût de calcul d'opérateurs en mémoire
(3) : On prend le coût maximal des adaptateurs selon le degré de parallélisme.
(4) : difficile à gérer : défini par la latence et le temps d'attente au moment de l'exécution de la requête.

2) - Coût sur les adaptateurs :

Il est clair que les sources de données dans un schéma de médiation sont indépendants et ne fournissent aucune information relative au coût. Plusieurs stratégies ont été proposées pour estimer le coût d'une requête sur une source.

- ***Estimation analytique :*** Propose des formules spécifiques de coût.
- ***Apprentissage progressif :*** qui base sur la gestion de l'historique de la source en fonction de sa réaction avec les différentes requêtes.

II.5.11. ***Quelques modèles de coût sur l'architecture de médiation :***

1) - Coût par calibrage (système PEGASUS) : base sur les principes suivants :

- exécution de requêtes types pour calibrer les paramètres de la source.
- Affiner le coût avec échantillonnage.

2) - Coût historique (système HERMES) : s'appuie sur les statistiques des requêtes précédentes.

3) - Coût générique (système DISCO) : s'appuie sur les coûts des différents adaptateurs.

II.6. Approche de médiation de contexte (Approche ontologique) : **[BEN 2000]**

II.6.1.Préface : La communication, le partage et l'échange d'informations sont des caractéristiques communes entre de nombreux systèmes qu'ils soient logiciels ou économiques, ainsi que des systèmes multi-agents à l'entreprise virtuelle .

Face aux nombreux problèmes que soulèvent la gestion des communications (langages différents, acteurs hétérogènes) d'une part, et la gestion des connaissances (modélisation, partage, échange, réutilisation) d'une autre part, les ontologies constituent un enjeu primordial. Il suffit pour s'en convaincre de citer les premiers projets d'ingénierie collaborative au début des années 90 jusqu'au projets les plus récents menés autour du WEB sémantique, sans oublier la technologie des agents logiciels et l'ingénierie de connaissances.

II.6.2. Présentation de l'approche : l'approche médiation de contexte ou aussi appelée approche ontologique est proposée dans le but d'adapter la médiation de schémas à des environnement ouverts et dynamique, tel que le WEB. Elle est basée sur une représentation explicite de la sémantique des différentes sources de données en utilisant les concepts d'ontologie et de contexte.

II.6.3. Une nouvelle approche de médiation de contexte ISIS (pour les systèmes SIG)

II.6.3.1. Définition d'un système SIG (Système d'Informations Géographique) :

Les SIG constituent l'un des domaines ou l'interopérabilité est complexe à réaliser, car les systèmes SIG manipulent des données possédant à la fois des attributs de localisation dans l'espace (GPS, Images satellites, photos aériennes, …) ce qui rend les méthodes de résolution des conflits de données encore plus difficiles.

II.6.3.2. Objectifs de l'approche ISIS (Interopérabilité des Systèmes d'Informations Spatiales) : **[BEN 2005]**

1 – Permettre à un utilisateur d'accéder d'une manière transparente à des ressources (données et opérations) fournies par plusieurs SIG hétérogènes et physiquement réparties.

2 – Offrir une interopérabilité non intrusive dans laquelle les systèmes coopérants ne sont pas modifiés dans leurs fonctionnalités habituelles (respecter l'autonomie de chaque système participant dans la coopération), mais aussi l'utilisateur les interroge en restant dans son contexte.

3 – Construire un système coopératif flexible permettant l'ajout de : fonctionnalités, de services, ou de nouvelles sources de données.

4 – incorporation d'un grand nombre de sources de données spatiales qui ne peuvent pas être envisagées en utilisant des techniques de fédération, dans ce cas, les sources d'informations déclarent tout simplement les données qu'elles partagent et la résolu--tion des conflits sémantiques et structurels est faite au moment de l'exécution des requêtes en utilisant la connaissance du domaine stockée dans l'ontologie et la notion d'objet de médiation.

II.6.3.3. Principe de l'approche ISIS : [BEN 2000] [BEN 2004]

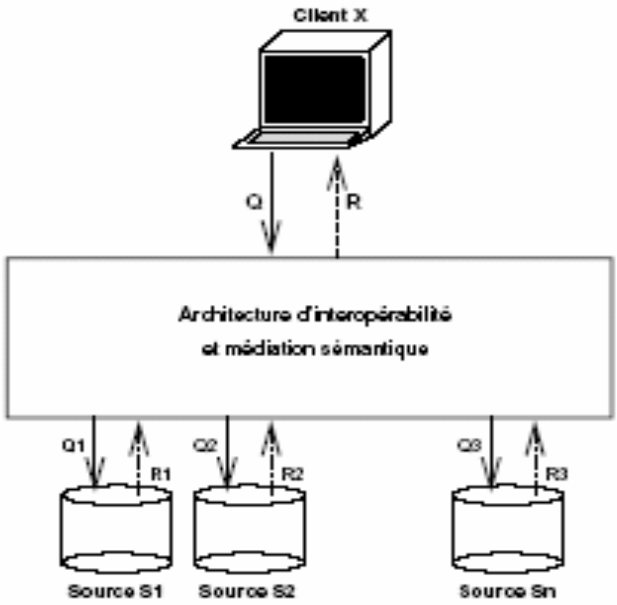


Fig II .9. Schéma général du principe d'interopérabilité dans ISIS

Le client X (un utilisateur ou une application) émet une requête Qx vers un ensemble de sources de donnée Si i=1,n et reçoit un résultat (une réponse) Rx

Dans cette application une interopération entre le client X et les sources Si suppose que :

- La source Si et le client X (l'application) adoptent un modèle de représentation commun.
- Si et X aient une compréhension mutuelle des éléments qu'ils partagent.
- X doit comprendre la signification des données mises à sa disposition par chaque source Si, et chaque source Si doit comprendre la signification de la requête Qx émise par X .
- X doit comprendre à son tour la signification du résultat Rx fourni par les sources Si.
- Un processus soit en mesure de transformer un objet de la représentation de la source Si vers la représentation du client X en adaptant dynamiquement sa sémantique et sa structure en fonction de la demande du client.

La réalisation de cette interopérabilité s'articule autour deux notions : les ontologies, les contextes.

1) - Une ontologie :

Définition 01 : Les ontologies sont des dictionnaires intelligents décrivant un domaine de discours .

La génération automatique et la gestion intelligente de telles ontologies sont deux objectifs à satisfaire pour supporter des bibliothèques électroniques intégrant des documents hétérogènes**. [BEN 2000]**

C'est encore un domaine à explorer en complément aux bases de données semi-structurées .

Définition 02 : une ontologie est une spécification explicite d'une conceptualisation d'un domaine d'application **[GRU 1993**]. La conceptualisation permet d'identifier par un processus d'abstraction les concepts et les termes essentiels d'un domaine.

La spécification rend explicite le sens associé aux différents termes. Généralement l'ontologie est définie par un vocabulaire commun constitué de termes et de relations entre les termes, en précisant leurs sémantiques les uns par rapport aux autres. **[ROC 2002]**

Les ontologies peuvent être représentées de façons différentes allant d'une simple liste de termes décrits par des informations textuelles (commentaires) aux langages plus labourés tels que KIF **[GEN 1992**], les langages terminologiques **[BOR 1994]**, les langages logiques **[KIF 1989]** et les langages objets.

2) - Un contexte : permet de compléter en utilisant un ensemble de connaissances les descriptions : statiques (structure) et dynamique (opérations) d'un objet pour en préciser explicitement la signification.
Ces connaissances peuvent être :

- Des informations textuelles (commentaires ou explications sur la définition d'un objet).
- Méta-données décrivant des données à l'aide de termes explicites ou des formules logiques qui expriment des contraintes sur l'utilisation de ces données.

II.6.3.4. Un exemple de SIG : Soient les deux applications : transport routier, tourisme envisagées par deux sites S1, S2

Extrait du schéma SH1 du site du transport routier S1 :

```
SH1=<{S1Route,S1Tronçon,S1Noeud},{}>
        Les trois classes précédentes sont définies comme suit :

   < S1Route, /*nom de la première classe*/
     [
       code : integer, catégorie : string,    /*liste des attributs*/
       tronçons : {S1Tronçon}, intersections : {S1Noeud}
     ],
    { distance(n1 : S1Noeud, n2 : S1Noeud):float } /*liste des méthodes*/
   >

   < S1Tronçon, /*nom de la deuxième classe*/
     [
       code_tronçon : integer, /*liste des attributs*/
       debut : S1Noeud, fin : S1Noeud,
       nb_voies_montantes : integer, nb_voies_descendantes : integer,
       limite_poids : float, limite_hauteur : float,
       limite_vitesse : integer,
       sens : string, geo : zone
     ],
    { get_longueur(): float,  /*liste des méthodes*/
      get_min_largeur(): float }
   >

   < S1Noeud, /*nom de la troisième classe*/
     [code_noeud : integer, geo : zone]> /*liste des attributs*/
```

Fig II .10. Extrait du schéma du site de l'application de transport_routier **[BEN 2000]**

Extrait du schéma SH2 du site toursitique S2 :

```
SH2=<{S2Route,S2TronçonRoute,S2Noeud},{}>
    Les trois classes précédentes sont définies comme suit :

   < S2Route, /* nom de la première classe*/
      [
       code : integer, type_route : string, /*liste des attributs*/
       tronçons : {S2TronçonRoute}, noeud: {S2Noeud}
      ],
      { distance(n1 : S2Noeud, n2 : S2Noeud):float } /*liste des méthodes*/
   >

   < S2TronçonRoute, /*nom de la deuxième classe*/
      [
       idf : integer, /*liste des attributs*/
       noeud1 : S2Noeud, noeud2 : S2Noeud,
       revêtement : string, geo : zone
      ],
      { get_longueur(): float } /*liste des méthodes*/
   >

   < S2Noeud, /*nom de la troisième classe*/
      [idf : integer, geo : zone],{}> /*liste des attributs*/
```

Fig II .11. Extrait du schéma du site de l'application de tourisme **[BEN 2000]**

Les deux applications précédentes mettent en jeu un grand nombre d'informations qu'on peut classifier comme suit :

- Des informations spécifiques à une application sans l'autre .
- Des informations communes entre les deux applications mais qui sont représentées de façon différentes pour chacune d'elles.
- L'application de transport routier définie sur le site S1 utilise un SIG qui stocke des données sur le réseau routier.
- L'application touristique définie sur le site S2 utilise un SIG qui stocke des données sur : les monuments, les sites touristiques classés, l'hôtelerie et la restauration, les transports (informations communes).

La solution d'interopérabilité sémantique d'un SIG proposée par le système ISIS base sur l'approche de médiation de contexte (l'approche ontologique). Et avant de traiter cette approche comme elle a été présentée par ISIS, nous présentons d'abord les trois principales solutions de médiation de contexte connues :

- OBSERVER **[MEN 100].**
- SEMWEB **[BIS 1997].**
- COIN **[GOH 1994].**

II.6.3.5. Principales solutions de médiation de contexte :

1) - Principe de la solution OBERVER : [BEN 2000]

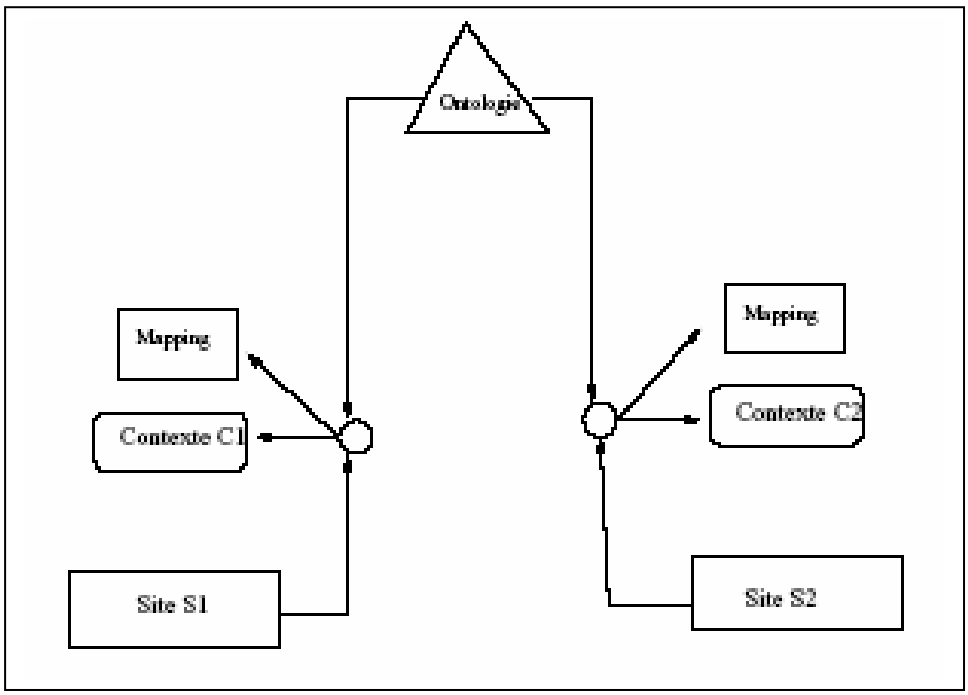


Fig II .12 . Principe de la solution OBSERVER .

- ***Le contexte d'un objet :*** est l'ensemble des informations contenues dans l'ontologie et qui caractérise cet objet, ainsi que ses usages possibles.

Formellement, un contexte C de l'objet O est défini comme suit :

$$C_{def}(O) = \{(C_i , V_i)\}$$

Avec :

C_i : des attributs issus de l'ontologie et caractérisent l'objet O.
V_i : des valeurs associées à ces attributs.

Exemple :

C_{def}(S1route) = {(Numéro , integer), (classe , {autoroute, nationale, départementale , contonale, chemin, avenue, boulevard, rue}) }

C_{def}(S2route) = {(Numéro, integer), (classe, {chemin, avenue, boulevard, rue}) }
/* classe : catégorie ou type de route*/

La traduction des objets d'un site à l'autre est réalisée par l'intermédiaire de leurs contextes, ceci est possible du fait que :

- Les contextes de tous les objets sont exprimés dans la même ontologie.
- Il existe des règles appelées des règles de mapping qui rend réalisable cette transformation de la définition locale d'un objet à sa définition globale et inversement.

2) - Principe de la solution SEMWEB : [BEN 2000]

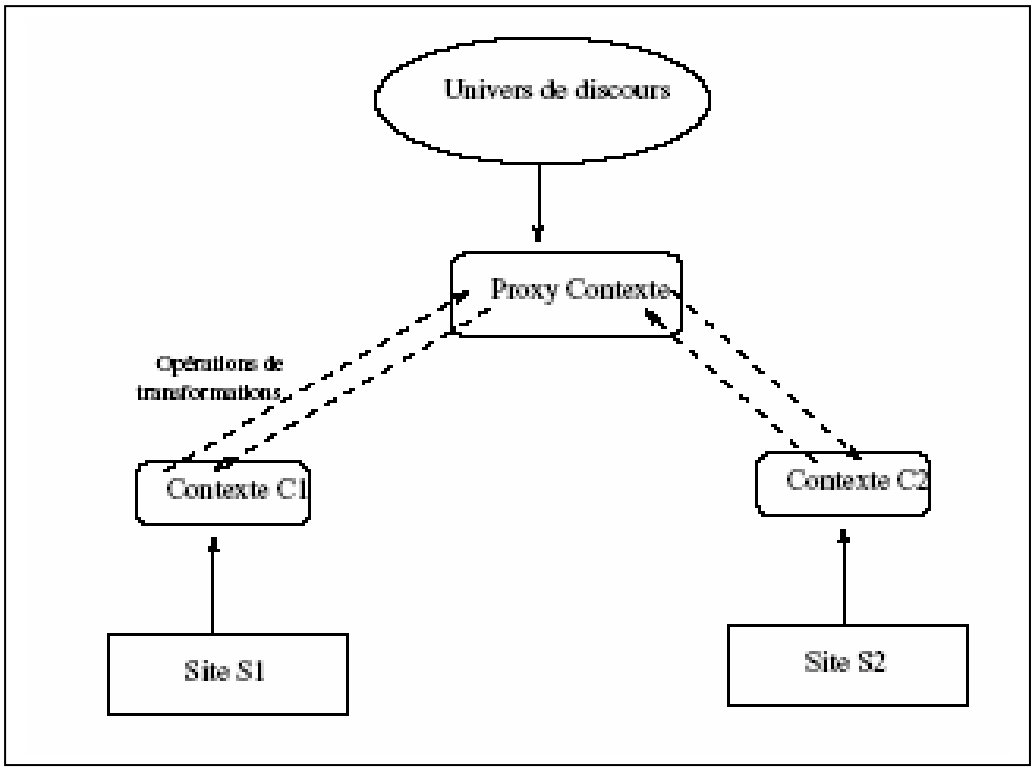


Fig II .13 . Principe de la solution SEMWEB .

Dans SEMWEB, le contexte est défini comme une caractérisation de la sémantique (seulement la sémantique) des différents objets. Ou bien le contexte d'un objet décrit le sens d'un objet tel qu'il est perçu localement.

La solution SEMWEB base sur deux types de contexte :

1 – Proxy contexte : sert à représenter la sémantique globale du domaine d'application.

2 –Contexte locale : constitué d'un ensemble de spécifications sémantiques des données locales.

Formellement, un contexte C que se soit local ou de proxy est défini comme suit :

C = <O.A , F , OP>

Avec :

O : Objet
O.A : Attribut A de l'objet O .
F : un ensemble de formules logiques qui caractérisent la sémantique de l'attribut A.
OP : un ensemble d'opérations de transformation de contexte permettant de transformer une partie de l'objet O du contexte C (contexte local) vers proxy contexte (contexte global) si C déjà un contexte local, soit vers tous les contextes locaux si C est déjà un proxy contexte.

La traduction des objets d'un contexte local C1 vers un contexte local C2 est réalisée en traduisant d'abord les objets du contexte C1 vers le proxy contexte et ensuite en traduisant les objets du proxy contexte vers le contexte C2 .

Exemple : Prenons toujours l'exemple des deux sites S1 et S2

C1 = < S1Route.Catégorie, catégorie ∈ {autoroute, nationale, départementale, cantonale, communale, avenue, boulevard, rue} , OP1 >

C2 = <S2Route.type_route, type_route ∈ {chemin, avenue, boulevard, rue}, OP2 >

Le proxy contexte C = <Route.Classe, classe ∈ {autoroute, nationale, départementale, cantonale, communale, chemin, avenue, boulevard, rue} , OP3 >

3) - Principe de la solution COIN :

La solution COIN sert à résoudre beaucoup plus les hétérogénéités de valeurs qu'aux hétérogénéités de structuration d'objets.

Le contexte dans COIN est défini pour associer la sémantique aux différentes valeurs et par conséquent, pour échanger des données entre systèmes hétérogènes.

La sémantique d'une valeur est exprimée en fonction de propriétés issues d'une ontologie commune et qui décrivent les différents aspects de la valeur (l'échelle, la monnaie, l'unité de mesure, …). Donc on parle d'un contexte de valeur plutôt que d'objet.

Formellement, le contexte C d'une valeur V est défini récursivement comme suit :

$$C = V(P_1 = V_1 , P_2 = V_2 , \dots , P_n = V_n)$$

Tels que :

C : Le contexte de la valeur V.
V : La valeur pour laquelle on veut attribuer un contexte C.
$P_1, P_2, \dots, P_n$: les différentes propriétés de la valeur issues d'une ontologie commune.
$V_1, V_2, \dots , V_n$: les différentes valeurs des propriétés $P_1, P_2, \dots, P_n$ associées à la valeur V

Exemple :

pour notre exemple précédent, considérons la valeur 4, représentant la hauteur minimum d'un tronçon routier.

On peut attribuer à cette valeur le contexte suivant :

C = 4 (Unité = 'mètre', précision = '99%') /* le contexte C de la valeur 4*/

Et qu'on peut expliciter par le fait que la hauteur minimum ayant comme valeur 4 est exprimée en mètre (affecter la valeur mètre à la propriété unité) avec une précision de 99% (affecter la valeur 99% à la propriété précision) .

- ***Critiques des solutions précédentes :***

1 – La solution OBSERVER :

- Considère l'ontologie comme une collection de termes et non pas d'une modélisation générale d'un domaine d'application indépendamment de tout besoin spécifique, d'ou il faut réécrire les schémas locaux en fonction des termes communs définis au niveau de l'ontologie (l'ontologie définie est assez restreinte) pour qu'ils soient compréhensibles de l'extérieur.
- La résolution des conflits structurels est limitée.
- Pas de résolution de conflits de valeurs.

2 – La solution SEMWEB :

- Malgré que la solution SEMWEB utilise le proxy contexte comme une modélisation d'un domaine d'application au contraire de la solution

OBSERVER, mais même cette modélisation est présentée dans un cadre théorique en utilisant la logique mathématique.
- La résolution des conflits structurels est limitée.

3 – La solution COIN :

- Ne résout que les conflits de valeurs, la résolution des autres types de conflits n'est pas du tout prise en compte (conflits structurels, de données, sémantiques, … etc), d'ou cette solution peut être efficace dans les applications d'échange de données ne supportant que des conflits de valeurs

4 – Inconvénients communs pour OBSERVER et SEMWEB :

Un inconvénient commun pour les deux solutions OBSERVER et SEMWEB et qu'elles supposent qu'il est toujours possible d'associer à un élément commun (de l'ontologie commun pour OBSERVER ou du proxy context pour SEMWEB) un et un seul élément local. Cela veut dire qu'aucune possibilité de restructuration des objets locaux n'est offerte afin de pouvoir correspondre à un élément commun une combinaison d'éléments locaux.

II.6.3.6. La solution ISIS (l'approche améliorée de médiation du contexte) :

Donnons quelques caractéristiques de la solution ISIS :

- Tenir compte de tous les types de conflits de données.
- Utiliser l'ontologie comme une modélisation abstraite d'un domaine d'application pour pouvoir ensuite interpréter son contenu en fonction des données locales.
- Les objets locaux peuvent être interprétés pour qu'ils puissent correspondre aux interprétations de termes d'ontologie.
-

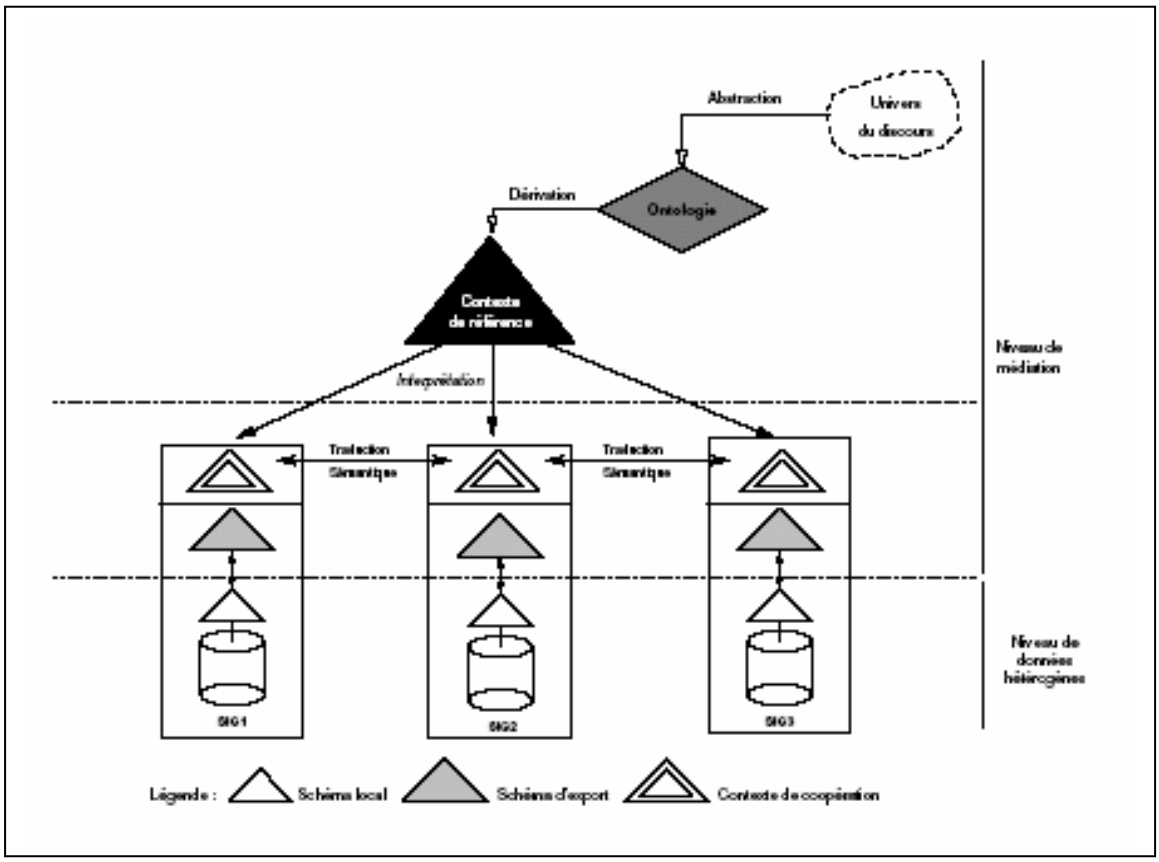


Fig II .14 . Structure générale de médiation de contexte dans ISIS .

a) - Les différentes étapes de médiation de contexte dans ISIS :

La médiation de contexte dans ISIS se réalise en quatre étapes sont :

1 – Extraction de la sémantique globale : la sémantique globale d'un domaine d'application constitue le moyen de communication et d'échange des données entre systèmes hétérogènes.

ISIS réalise sa sémantique globale en deux phases :

Phase I : Est un processus d'abstraction qui consiste à définir une ontologie spatiale générale d'un domaine d'application sous forme de termes abstraits et génériques.

Phase II : Est un processus de dérivation consistant à spécialiser les concepts de l'ontologie par dérivation en saisissant les propriétés structurelles et sémantiques des objets issus du monde réel.

Un schéma de médiation résulte de la dérivation constituera un contexte de référence indépendant de tout besoin particulier des systèmes hétérogènes (son contenu est différent des schémas locaux des différents SIG) et permet la coopération entre ces derniers.

2 – Création de schémas d'export : un schéma d'export permet à un site de définir par mécanisme de vue les données de schéma local partageables avec les autres sites.

3 – Interprétation locale du contexte de référence : réécrire un sous ensemble du contexte de référence en fonction de la structure et de la sémantique des objets locaux des schémas d'export.

Des schémas de coopération, résultent des interprétations par les différents systèmes SIG constituent des contextes de coopération.

4 – Traduction sémantique des données : l'échange de données entre SIG, est basé sur un processus de traduction sémantique. Qui consiste à faire la correspondance entre tout ou une partie du contexte de référence (contexte global) avec chacun des contextes de coopération afin de pouvoir transformer un objet défini dans le contexte d'un SIG, en un autre objet équivalent dans le contexte d'un autre SIG.

b) - Les différentes catégories d'ontologies :

Il existe plusieurs catégories d'ontologies, on se limite ici par deux qui sont :

1 – Les ontologies maximum : offre une description détaillée et exhaustive des concepts d'un domaine **[LEN 1990].** Elles permettent une représentation plus proche et plus fidèle du sens d'un vocabulaire. Ces ontologies sont généralement difficile à construire, car elles peuvent contenir un grand nombre de termes dont la sémantique doit être compréhensible et acceptée par tous les systèmes d'informations. De plus, l'ajout de nouveaux concepts peut introduire des incohérences au niveau de l'ontologie.

2 - Les ontologies minimum : sont composées d'un ensemble réduit de termes essentiels (c.à.d elles sont moins détaillées), elle visent à définir des concepts qui représentent les termes d'un ou plusieurs domaines.

Dans ISIS, nous avons adopté la catégorie des ontologies minimum car :

- Elle paraît moins compliquée et plus réaliste.

Il s'agit d'une ontologie spatiale caractérisée par deux propriétés :

1. Multi-niveaux
2. générique

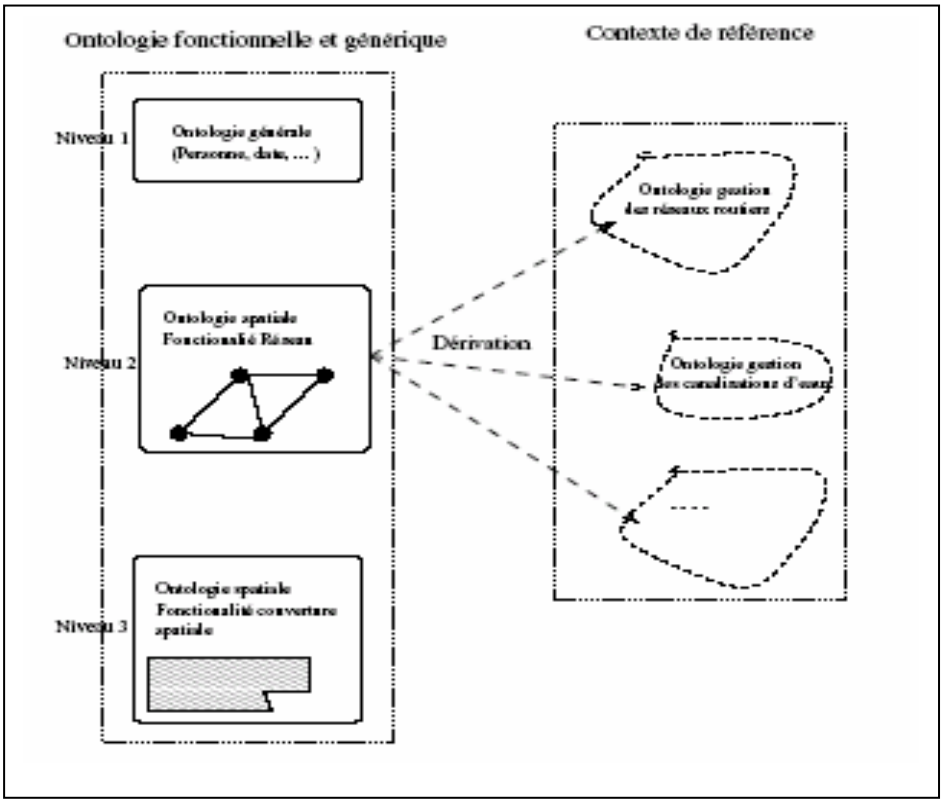


Fig II .15 . Ontologie multi-niveaux et générique .

1 – Une ontologie multiniveaux : une ontologie est multi-niveaux si elle est constituée de différents niveaux fonctionnels liés entre eux par des relations qui permettent de combiner leurs termes ultérieurement ; car les applications spatiales requièrent souvent des données provenant de systèmes hétérogènes organisés sous forme de couches thématiques partageant des fonctionnalités communes. **[BEN 2000]**

D'autres niveau peuvent être créés pour décrire d'autres fonctionnalités telles que :

- Le niveau couverture spatiale pour la gestion des espaces verts.
- Le niveau particulier dit de haut niveau définit une ontologie générale constituée de termes communs à plusieurs domaines d'application telles que (date, employé, point, localisation, … etc).

2 – Une ontologie générique : porte sur des concepts généraux qui sont indépendants d'un domaine ou d'un problème particulier, tels que les concepts : temps, espace, notions mathématiques. Les ontologies génériques peuvent être utilisées dans des situations diverses et variées. **[BEN 2000]**

Par exemple dans (Fig II.15) le niveau 02 représente une ontologie constituée de termes généraux pour les applications de réseau de routes et de canalisation d'eau en même temps (c'est une ontologie généralisée). Se sont des termes génériques : réseau, nœud, arc.

A ce niveau d'abstraction, la nature et la structure des objets traversant les arcs du réseau peuvent être des véhicules dans le cas d'un réseau routier, et des quantités d'eau dans le cas d'un réseau de canalisation d'eau. De même, les nœuds peuvent être des croisements de routes ou des vannes.

De ce fait, les termes : réseau, nœud, arc constituent des termes génériques dont la spécialisation viendra lors de la création d'un contexte de référence pour définir les entités réelles.

Un terme générique d'une ontologie est décrit par une classe abstraite constituée de spécifications algébriques **[EHR 1985].**

Cette classe est définie par un ensemble de signatures de méthodes et un ensemble d'axiomes.

c) - Contexte de référence : créé à a partir de l'ontologie par un processus de dérivation, il décrit un schéma de médiation permettant aux SIG d'interopérer.

d) - Processus de dérivation : permet de créer des concepts qui ont un sens dans un domaine d'application en spécialisant des termes générique de l'ontologie lorsque ceci est nécessaire.

Par exemple les concepts génériques : réseau, nœud, arc de l'ontologie urbaine peuvent être spécialisés basant sur ce processus de dérivation par les concepts : route, nœud_routier, tronçon_routier.

Ontologie urbaine

abstract class Réseau
uses Noeud, Arc
methods
ajouter_noeud : Noeud →Réseau
supprimer_noeud : Noeud→ Réseau
ajouter_arc : Arc→ Réseau
supprimer_arc : Arc→Réseau
nb_arc : →integer
nb_noeud : → integer
successeur : Noeud →Noeud
ancêtre : Noeud→Noeud
distance : Noeud x Noeud →float
chemin : Noeud x Noeud →**set of** Arc
extrémités : Arc → **set of** Noeud
plus_court_chemin : Noeud x Noeud →**set of** Arc
composante_fortement_connexe : Nœud → **set of** Nœud ……
...
axioms
∀n ∈ Nœud, ∀ r ∈ Réseau, ∀a ∈ Arc
r.nb_noeud+1 = (r.ajouter_noeud(n)).nb_noeud()
(r.ajouter_noeud(n)).supprimer_noeud(n)=r
r.distance_noeud(n,n) = 0
r.successeur(r.ancêtre(n))=n
r.ancêtre(r.successeur(n))=n

abstract class Noeud
methods
créer_noeud :_Noeud
supprimer_noeud : →
position_noeud : →Localisation
déplacer_noeud : Localisation → Nœud …..
...
abstract class Arc
methods
créer_arc : Noeud x Noeud→Arc
créer_arc_orienté : Noeud x Noeud →Arc
supprimer_arc : →
longueur_arc : →float
fixer_valeur : float →Arc
lire valeur : →float

Fig II .16 . Spécification des termes génériques de l'ontologie urbaine .

e) - Représentation du contexte de référence :

Un terme dérivé est représenté par une classe appelée classe de mediation

Formellement, on exprime une classe de médiation d'un terme dérivé par :

$$Cm \in CM = \langle Nom\_cm , List\_attr , list\_mét \rangle$$

Avec : nom_cm : nom de la classe de médiation
List_attr : liste des attributs de la classe de médiation
List_mét : liste des méthodes de la classe de médiation

Extrait du schéma de médiation :

```
SM=<{CM_Route,CM_Tronçon,CM_Noeud_Routier},{}>

< CM_Route,        /*nom de la classe de médiation*/
  [
   idf : integer, /*liste des attributs*/
   catégorie : string domval {"Autoroute", "Nationale", "Départementale",
   "Cantonale", "Communale", "Chemin",
   "Sentier", "Avenue", "Boulevard", "Rue"}, /*domaine de valeur*/
   tronçons : {CM_Tronçon}, noeud_routiers : {CM_Noeud_Routier}
  ],
  { ajouter_noeud_routier(n: CM_Noeud_Routier): Route, /*liste des méthodes*/
    supprimer_noeud_routier(n : CM_Noeud_Routier): Route,
    nb_tronçon(): integer,
    nb_noeud_routier(): integer,
    successeur(n : CM_Noeud_Routier):CM_Noeud_Routier,
    ancêtre(n : CM_Noeud_Routier):CM_Noeud_Routier,
    distance(n1 : CM_Noeud_Routier, n2 : CM_Noeud_Routier):float,
    plus_court_chemin(n1 : CM_Noeud_Routier, n2 :
    CM_Noeud_Routier):{CM_Tronçon}
  ...}>

< CM_Tronçon,          /*nom de la classe de médiation*/
  [
    idf : integer, /*liste des attributs*/
    noeud_debut : CM_Noeud_Routier, noeud_fin : CM_Noeud_Routier,
    nb_voies : integer, direction : string,
    nb_voies_montantes : integer, nb_voies_descendantes : integer,
    revêtement : string domval {"bitume", "béton",   /*domaine de valeur*/
    "pavés", "graviers", "sable", "terre"},
    max_hauteur: float (unité:"m"), /*sémantique de valeur*/
    adr_deb_dte : integer, adr_deb_gche : integer,
    adr_fin_dte : integer, adr_fin_gche : integer,
    geo : zone,
  ...],
  { supprimer_tronçon(), /*liste des méthodes*/
    créer_tronçon(n1 : CM_Noeud_Routier, n2 : CM_Noeud_routier):CM_Tronçon,
    get_longueur(): float,
    get_min_largeur(): float,
    get_adresse(float): integer,
  ...}>
< CM_Noeud_Routier,             /*nom de la classe de médiation*/
  [
    idf : integer, /*liste des attributs*/
    geo : zone
  ],
  { créer_noeud_routier(idf: integer, geo:zone),     /*liste des méthodes*/
    supprimer_noeud_routier(),
    /*cette méthode retourne le centroide du polygone (zone) */
    position_noeud_routier(CM_Noeud_Routier):Point,
  ...}>
```

Fig II .17 . Extrait du contexte de référence d'une classe .

f) - Construction du contexte de coopération :

Le contexte de coopération rassemble les interprétations locales des classes de médiation du contexte de référence. Il établi un lien entre les classes locales et les interprétations des classes de médiation.

La création du contexte de coopération se fait en trois étapes :

1 – Interprétation locale du contexte de référence : consiste à agréer certaines classes de médiation en fonction du contenu local. Ces agréments permettent d'identifier pour une classe de médiation les attributs et les méthodes qui ont une signification locale sur le site S

Le résultat de l'interprétation est appelé rôle de médiation.

2 – Restructuration des classes locales : l'interprétation d'une classe de médiation peut correspondre soit à une seule classe locale, soit à plusieurs classes locales à cause les conflits de schémas qui peuvent coexister .

De même, une même classe locale peut être attachée à plusieurs interprétations possibles issues de différentes classes de médiation.

Pour établir des correspondances 1-1 entre les interprétations d'une classe de médiation et les classes locales, on doit restructurer ces dernières et les intégrer dans le contexte de coopération, cela se réalise par la création de classes virtuelles.

3 – Unification des sémantiques : entre un rôle de médiation et une classe virtuelle en utilisant un ensemble de fonctions appelées fonctions de transformation de contexte permettant de transformer un rôle de médiation de la sémantique du contexte de référence dans la sémantique locale d'une classe virtuelle et inversement.

g) - Représentation du contexte de coopération :

Un contexte de coopération est représenté par un ensemble de classes de coopération qui décrivent les objets partagés accessibles depuis les autres systèmes appelés objets de coopération.

Une classe de coopération encapsule à la fois les concepts : rôle de médiation, classe virtuelle, et transformation de contexte.

Les objets de coopération comportent deux sémantiques :

1 – Sémantique de référence : qui confère à l'objet une compréhension par les autres sites de coopération au moyen de son rôle de médiation qui le relie à une classe de médiation.

2 – Sémantique locale : qui lui confère une signification sur un site au moyen de la classe virtuelle que le relie à des classes locales.

h) - Rôle de médiation :

Représente une interprétation possible d'une classe de médiation, c'est en fin une classe constituée d'un ensemble d'attributs et de méthodes.

Formellement, un rôle de médiation est défini comme suit :

$$rm \in RM = <Nom\_rm , cm\_rm , List\_attr , Liste\_mét , Q>$$

avec :
nom_rm : nom du rôle de médiation.
Cm : classe de médiation dont rm est une représentation.
List_attr : sous ensemble d'attributs de cm.
List_mét : sous ensemble de méthodes de cm
Q : la qualification associée à rm

```
< RM_Route_Tourisme, /*nom du rôle de médiation*/
  CM_Route, /*nom de la classe de médiation*/
   [
     idf : integer, /*liste des attributs*/
     catégorie : string domval {"Avenue", "Boulevard", "Rue", "Chemin"}
     tronçons : {CM_Tronçon}, noeud_routiers : {CM_Noeud_routier}
   ],
   {distance : noeud_routier(n : CM_Noeud_routier) float}, /*méthodes*/
   {}
>

< RM_Tronçon_Tourisme, /*nom du rôle de médiation*/
  CM_Troncon, /*nom de la classe de médiation*/
  [
    idf : integer,
    noeud_debut: CM_Noeud_Routier, noeud_fin: CM_Noeud_Routier,    /*liste des
                                                                    attributs*/
    geo : zone
  ],
   {get_longueur(): float}, /*liste des méthodes*/
   {},
>

< RM_Noeud_Routier_Tourisme, /*nom du rôle de médiation*/
  CM_Noeud_Routier, /*nom de la classe de médiation*/
    [
     idf : integer, geo : zone
    ],
    {},
    {}
>
```

Fig II .18 . Rôles de médiation définis pour l'application touristique .

i) - Les classes virtuelles :

Se sont des classes intermédiaires entre les classes locales et les rôles de médiation. Elles sont définies au niveau du contexte de coopération et exprimés dans la sémantique locale d'une source de données. Elles sont réalisées à l'aide de l'algèbre classique.

Formellement, une classe virtuelle CV est exprimée comme suit :

$$cv \subset CV = \langle Nom\_cv , List\_attr, List\_mét \rangle$$

avec :
Nom_cv : nom de la classe virtuelle.
List_attr : l'ensemble des attributs de cv
List_mét : l'ensemble des méthodes de cv.

```
                    Classe virtuelle du site du transport routier S1 :

< CV_Tronçon_Transport, /*nom de la classe virtuelle*/
  [
    code_tronçon : integer, /*liste des attributs*/
    debut : S1Noeud, fin : S1Noeud,
    sens : string, nb_voies_montantes : integer,
    nb_voies_descendantes : integer, limite_poids : float,
    limite_hauteur : float, geo : zone
  ],
  { get_longueur(): float, /*liste des méthodes*/
    get_min_largeur(): float
  }
>
                Classe virtuelle du site de l'application touristique S2 :

< CV_Tronçon_Tourisme, /*nom de la classe virtuelle*/
  [
    idf : integer, /*liste des attributs*/
    noeud1 : S2Noeud, noeud2 : S2Noeud,
    geo : zone
  ],
  {get_longueur(): float /*liste des méthodes*/
  }
>
```

Fig II .19. Deux classes virtuelles pour les applications : transport_routier , touristique .

j) - Les transformations de contexte :

Les fonctions de transformation de contexte permettent de transformer les objets de la classe virtuelle vers le rôle de médiation et inversement.

Elles définissent des correspondances bidirectionnelles au niveau de la structure et des domaines de valeurs entre les attributs du rôle de médiation et les attributs de la classe virtuelle afin de résoudre les conflits de schémas et les conflits liés à la sémantique des valeurs .

Transformation de la valeur d'un objet de la classe virtuelle vers le rôle de médiation

```
rm
cv  RM_Tronçon_Transport.max_hauteur(
              CV_Tronçon_Transport.hauteur_limitée)
          =
          {
            return(CV_Tronçon_Transport.hauteur_limitée/100);
          }
```

Transformation de la valeur d'un objet du rôle de médiation vers la valeur d'un objet de la classe virtuelle

```
rm
cv  CV_Tronçon_Transport.hauteur_limitée(
              RM_Tronçon_Transport.max_hauteur)
          =
          {
           return(100*RM_Tronçon_Transport.max_hauteur);
          }
```

Fig II .20. Extrait des transformations de contexte entre la classe virtuelle CV_Tronçon_Transport et le rôle de médiation RM_Tronçon_Transport

Avec :

$$\overset{rm}{cv}\ \ cv.A\ (rm.A_1, rm.A_2, rm.A_3\ , \ldots\ldots\ldots\ldots, rm.A_n\ )$$

Ou : A est un attribut de cv , A_i est un attribut de rm

$$\overset{rm}{cv}\ \ rm.A\ (cv.A_1, cv.A_2, cv.A_3\ , \ldots\ldots\ldots\ldots, cv.A_n\ )$$

Ou : A est un attribut de rm , A_i est un attribut de cv

les fonctions de type $\overset{rm}{cv}$ permettent de passer de la valeur du rôle de médiation à une valeur de l'objet local.

Les fonctions de type $\overset{rm}{cv}$ permettent de passer d'une valeur de l'objet local à une valeur de rôle de médiation.

k) - Classe de coopération :

Une classe de coopération encapsule trois composants :
une classe virtuelle, un rôle de médiation, un ensemble de transformation de contexte qui établissent des correspondances entre la classe virtuelle et le rôle de médiation.

Formellement, une classe de coopération est définie comme suit :

cc ∈ CC = <Nom_cc, CV, ContC>

avec :
nom_cc : nom de la classe de coopération.
Cv : une classe virtuelle.
contC : le contexte de la classe de coopération C

```
< CC_Troncon_Transport, /*nom de la classe de coopération*/
  < CV_Troncon_Transport, /*nom de la classe vituelle*/
   [
    code_tronçon : integer, /*liste des attributs de la classe vituelle*/
    debut : Noeud, fin : Noeud,
    sens: string, nb_voies_montantes: integer,
    nb_voies_descendantes:integer,
    poids_max_autorisé : float, hauteur_max_autorisée : float,
    geo : zone
   ],
   { get_longueur(): float, /*liste des méthodes de la classe vituelle*/
     get_min_largeur(): float
   }
  >
  <

< RM_Tronçon_Transport, /*nom du rôle de médiation*/
  CM_Tronçon, /*nom de la classe de médiatiom*/
   [
    idf : integer, /*liste des attributs du rôle de médiation*/
    noeud_debut : Noeud_Routier, noeud_fin : Noeud_Routier,
    nb_voies : integer, direction : string,
    nb_voies_montantes : integer, nb_voies_descendantes : integer,
    geo : zone
   ],
   { get_longueur(): float, /*liste des méthodes du rôle de médiation*/
     get_min_largeur(): float,
     get_max_hauteur(): float,
     get_max_poids(): float
   },
   {get_min_largeur>"2"}
 >,

/*fonctions de transformations de contexte*/
<| rm
   cv RM_Tronçon_Transport.nb_voies(
       CV_Troncon_Transport.nb_voies_montantes,
       CV_Troncon_Transport.nb_voies_descendantes)={
         CV_Troncon_Transport.nb_voies_montantes+
         CV_Troncon_Transport.nb_voies_descendantes}
| rm
  cv RM_Tronçon_Transport.idf(
      CV_Troncon_Transport.code_tronçon)=
      return(CV_Troncon_Transport.code_tronçon)}
...>
>
>
```

***Fig II .21. Description de la classe de coopération* CC_Troncon_Transport**

l) - La traduction sémantique :

La traduction sémantique définie dans ISIS consiste à traduire des données d'un site S2, appelé site fournisseur, à un autre site S1, appelé site demandeur, en tenant compte des conflits sémantiques des données.

m) - Génération des données sur le site fournisseur :

Un site fournisseur S2 , possédant une classe de coopération CC2 sémantiquement proche de la classe CC1, génère des données de la manière suivante :

- Création des objets de coopération : les données de site fournisseur S2 à envoyer au site demandeur S1 doivent être représentées dans la structure et la sémantique de la classe de coopération CC2.
- Calcul des valeurs de médiation : les fonctions de transformation de contexte définies dans la classe CC2 sont appliquées pour transformer les valeurs locales en valeurs de médiation et ceci pour chaque objet de coopération.
- Création d'unités de transfert : les données échangées entre sites sont réalisées au travers d'unités de transfert qui contiennent des données exprimées dans la sémantique du contexte de référence. Chaque unité de transfert contient des informations extraites d'un seul objet de coopération.

Formellement, une unité de transfert est définie par :

$$UT = Extraire_{rôle}(OC) = <Val_{rôle}>$$

Avec : extraire : est une fonction qui permet d'extraire la valeur du rôle de médiation d'un objet de coopération OC

Les différentes UT sont envoyées au site demandeur S1.

n) -Transformation des données du site fournisseur vers le site demandeur :

A la réception des unités de transfert, le site demandeur S1 procède à la génération des données conformes à sa propre sémantique. Cette génération est réalisée en plusieurs étapes qui sont :

- Création d'un objet de coopération : la classe de coopération CC1 est instanciée en affectant des valeurs de médiation des différentes UT aux différentes rôles de médiation.
- Calcul des valeurs locales : les fonctions de transformation de contexte définies dans la classe CC1 sont appliquées pour transformer les valeurs de médiation en valeurs locales dans chaque objet de coopération.

II.6.4. Une architecture idéale pour l'interopérabilité des SIG :

Notre objectif est toujours de développer une architecture d'interopérabilité des SIG dans un environnement dynamique et ouvert.

La majorité des architectures d'interopérabilité existant sont basées sur le paradigme objet, et donc leur implémentation utilise très souvent des architectures objets distribuées comme CORBA **[AMA 1997 , BAL 1998]**. Par contre très peu de solutions ont recourt au paradigmes agent **[FOW 1999**].

L'utilisation des agents pour implémenter la médiation de contexte dans ISIS est motivée par le fait que les agents ont des caractéristiques intéressantes notamment : L'autonomie et la coopération.

L'autonomie est la capacité d'un agent à exécuter un certain nombre de taches indépendamment des autres agents.

La coopération est la capacité d'un agent à interagir avec d'autres agents en vu de la résolution coopérative d'un problème.

II.6.4.1. Représentation générale de l'architecture :

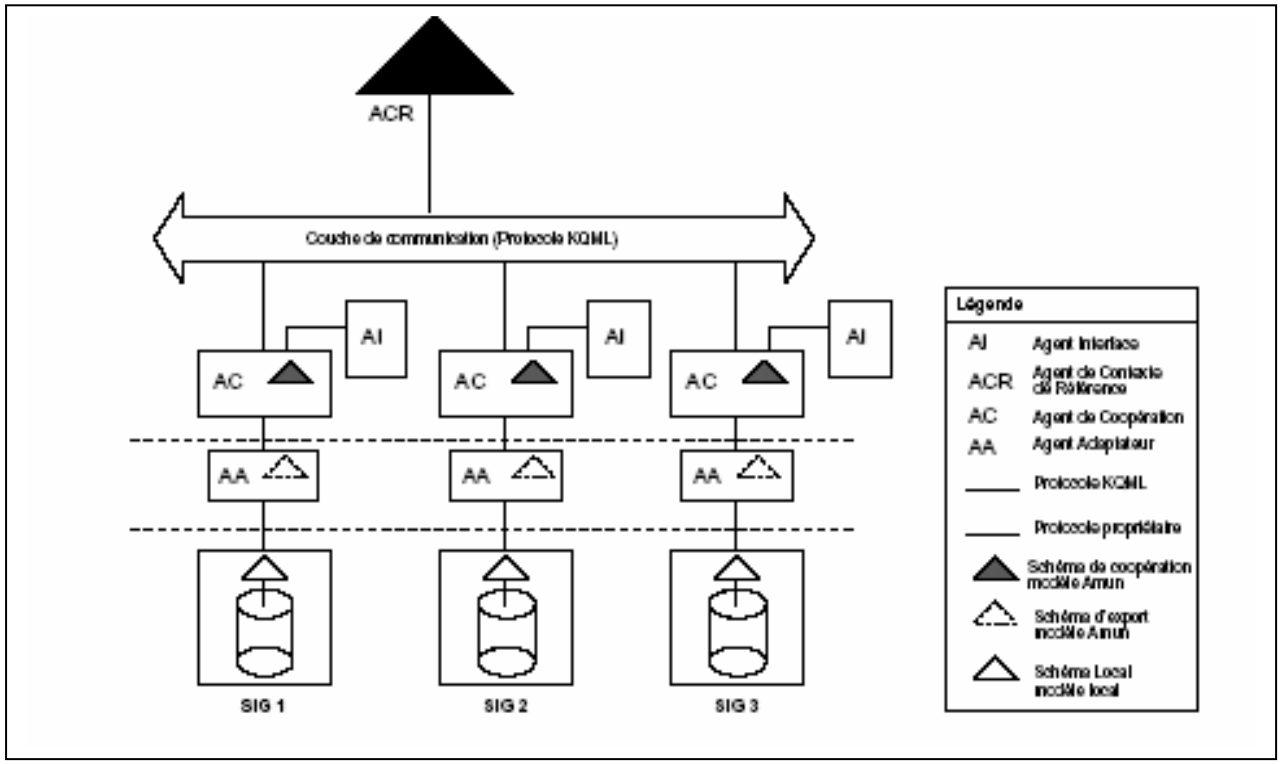


Fig II .22. Architecture de médiation de contexte multi-agents

L'architecture ISIS est un système multi-agents qui combine différent types d'agents réalisant des taches complexes de la médiation de contexte et des taches simples comme l'accès aux données du SIG.

ISIS se compose de quatre types d'agents :

1 – Agent de contexte de référence (ACR) : est unique dans l'architecture, il a pour tache principale de gérer la consultation et la mise à jour du contexte de référence.

2 – Agent de coopération (AC) : s'occupent de la traduction sémantique des données, ils sont associés chacun à une et une seule source de données et stockent de schémas de coopération correspondants.

3 – Agent adaptateur (AA) (Wrappers) : gèrent les accès aux sources de données.

4 – Agent interface (AI) : servir d'interface entre l'utilisateur et les autres agents du système, ils permettent à un utilisateur de formuler une requête et de l'adresser à un agent précis.

L'ensemble des agents de l'architecture communiquent en KQML (Knowledge Query Manipulation Language) **[FIN 1994]**. Notons que KQML est un langage de communication de haut niveau permettant aux agents de s'échanger des messages à l'aide d'un ensemble de primitives de communication (comme Tell, Ask-if, Forward) **[SEA 1969]**.

II.6.4.2. Prototype de l'architecture ISIS :

Deux phases de prototypage ont été menées :

Phase I : constituée de :

- Implémenter un concept d'agent générique qu'on peut instancier pour créer les différents types d'agents d'ISIS.

- Implémenter les primitives de communication KQML.

Cette phase a été réalisée dans l'environnement JAVA, et en utilisant l'outil JATLITE **[JAT 1997]** qui est une bibliothèque de classes JAVA pour le développement d'agents.

Phase II : consiste à :

Implémenter les différents agents adaptateurs pour fournir une interface uniforme d'accès au différents SIG. Cette phase consiste à mettre en œuvre les différents SGBD relationnels et objets (ORACLE, ACCES, POSTGRES, O2, …) et les différents API (Application Programming Inteface) propriétaires. Le protocole JDBC (Java Data Base Connectivity).

II.6.5. Conclusion :

L'interopérabilité des SIG est certainement une tâche complexe. Elle est de plus en plus inévitable si l'on veut réutiliser des données géographiques existantes ou concevoir des applications nécessitant des accès coordonnés à plusieurs sources de données.

L'approche ISIS a été proposée comme solution d'interopérabilité sémantique de SIG. Elle définit une approche de médiation de contexte basée sur l'utilisation de plusieurs concepts : une ontologie pour une conceptualisation abstraite et générique d'un domaine d'application, un contexte de référence pour la définition d'un schéma de médiation qui spécialise et précise le contenu de l'ontologie, des contextes de coopération qui interprètent le contexte de référence et par conséquent constituent la sémantique des objets locaux des SIG et enfin une traduction sémantique qui permet de traduire les objets d'un SIG à un autre en tenant compte des hétérogénéités structurelles et sémantiques des données.

II.7. Approche des bases de données fédérées :

II.7.1. Préface : Le développement d'une nouvelle application de traitement de données fait appel à des données déjà mémorisées dans des fichiers ou plusieurs bases de données indépendantes.

Dans les grandes entreprises, l'usage de l'informatique se traduit par un développement indépendant de plusieurs bases de données pour les applications spécifiques de chaque service ou filiale. Les frontières entre ces services ou filiales sont alors susceptibles de bouger, en créant de nouveaux centres d'intérêts, demandant de nouvelles applications qui devront être construites basant sur des données prises d'ici et là, plus qu'avec de nouvelles données spécifiques.

Ainsi, le développement des nouveaux systèmes d'informations repose sur la capacité du système informatique de réaliser l'interopérabilité entre bases de données existantes. **[CHR 1996]**

Cette interopérabilité peut être réalisée à trois niveaux :

1 - Au niveau le plus interne dans l'architecture des SI : en mettant en place des passerelles (gateways), c.à.d des programmes spécifiques qui établissent des connections entre systèmes de telle manière qu'un système donné peut accéder aux données d'un autre système. Par exemple les passerelles qui existent actuellement entre plusieurs systèmes de gestion de bases de données (SGBD).

2 - Au niveau intermédiaire : en installant un système dit système multi-bases de données **[LIT 1990]**, c.à.d une couche logicielle permettant à chaque utilisateur de définir sa vue sur un ensemble de bases de données. Ces systèmes garantissent les connections appropriées selon les vues définies et permettent l'accès aux données réparties, mais ne prennent pas en charge les contraintes de cohérence entre les différentes sources de données.

3 - Au niveau le plus élevé : en développant au dessus des systèmes existants, un système global pour fournir le niveau désiré d'intégration des sources de données.

Dans cette optique, deux approches ont été proposées :

1 - Les SGBD répartis **[CER 1987]** : qui s'appuient sur une intégration totale et un contrôle centralisé.

toutes les données existantes des bases de données sont intégrées en une base de données logique unique dite base de données répartie gérée d'une manière cohérente par une seule autorité globale de contrôle (un SGBD réparti).

Cette approche a démontré qu'elle ne satisfaisait pas la demande des entreprises dont le besoin d'accéder à plusieurs sources de données doit être satisfait sans interférer avec l'exploitation normale de ces sources par leurs propriétaires.

2 – Les systèmes de bases de données fédérées **[SET 1990]** : ces systèmes fédérés visent une intégration flexible (souple à l'extensibilité) en fonction de la demande, c.à.d équilibrer entre le besoin d'intégration de données et l'autonomie des sites.

Ils donnent à chaque administrateur de données la possibilité de définir les sous-ensembles des données locales mis à la disposition des utilisateurs du système fédéré.

Ces sous-ensembles sont intégrés en une ou plusieurs bases de données virtuelles (ne sont pas physiques) appelées BDF.

L'intégration, l'importation et l'exportation des données dans les BDF, … sont gérées par le système fédéré qui doit respecter les accords de coopération établis par les administrateurs en termes de :

- sémantique de données.
- Règles d'accès.
- Gestion des copies.

II.7.2. Définition d'une base de données fédérée : Une base de données fédérée est une base de données répartie, hétérogène, c.à.d constituée à partir de sources de données de nature variées (fichiers classiques, fichiers textes, documents HTML, XML, base de données relationnelle ou objet, …etc).

L'objectif de fédération des bases de données est de fournir aux utilisateurs une vue intégrée de différentes données de l'entreprise**. [CHE 2001]**

Le besoin de fédération des sources de données est important dans les grandes entreprises, notamment pour supporter l'alimentation des entrepôts de données (data warehouses).

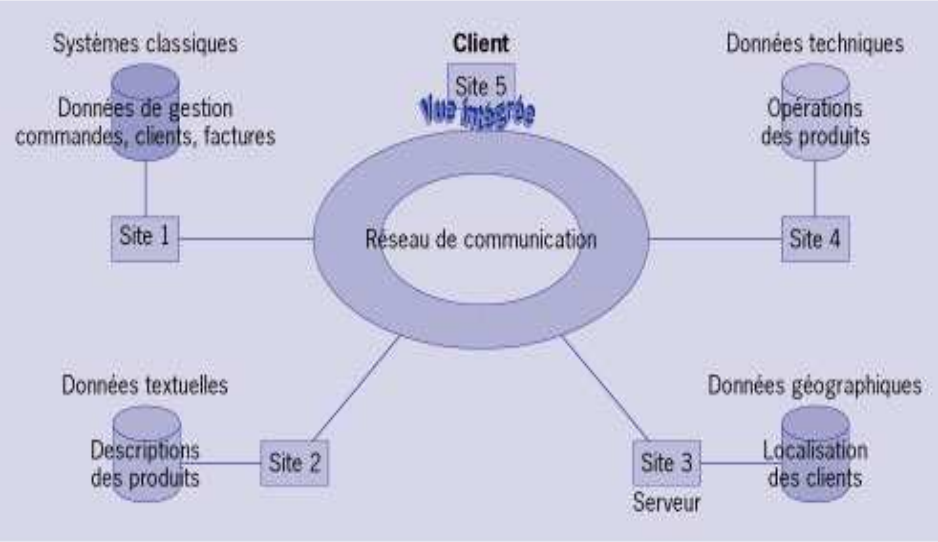


Fig II.23. Exemple d'une base de données fédérée

II.7.3. Architecture de système fédéré :

Historiquement, les bases de données fédérées se sont développées autour du modèle relationnel et SQL. Celui-ci constitue le modèle pivot de la première génération de système fédéré.

CORBA peut être perçu comme une deuxième génération permettant d'intégrer les applications au delà des données (fédération des applications).

Du coté architecture, les bases de données fédérées ont convergé vers un standard proposé par DARPA et GIO WIDERHOLD : l'architecture I3. **[BEN 2002]**

Cette architecture s'articule sur trois niveaux :

- Le niveau central est le médiateur chargé de l'intégration des données en provenance des différentes sources pour composer les réponses aux requêtes des utilisateurs .
- Le niveau le plus interne est composé d'adaptateurs ou WRAPPER chargés de transformer les données locales de sorte à les rendre assimilable par le médiateur et aussi de filtrer autant que possible les données suite aux requêtes.
- Le niveau externe est composé de facilitateurs, outils chargés à la fois de localiser les données pertinentes (disponible tout le temps) et de les mettre en forme pour les applications.

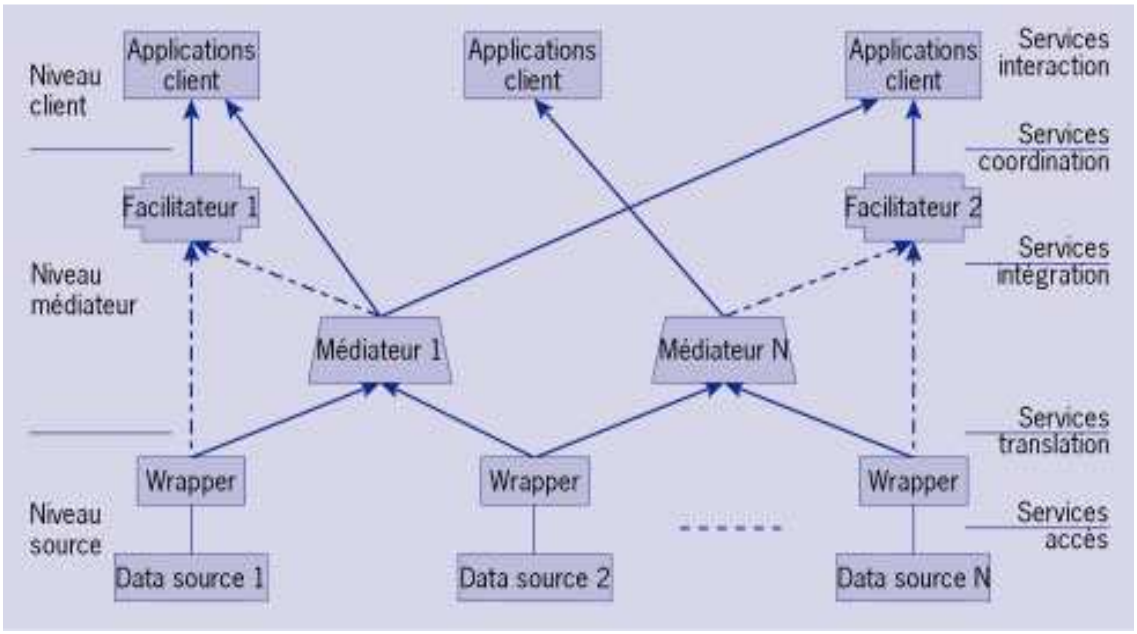


Fig II.24. **Architecture DARPA I3**

II.7.4. Avantage majeur de l'approche fédération :

Alors que les passerelles (Getways), et les systèmes multi-bases de données fournissent aux utilisateurs un langage d'accès multi-bases de données (souvent de type SQL), sans aucune tentative d'unifier les sémantiques des données des différentes sources.

Les systèmes de bases de données distribuées et les systèmes de bases de données fédérées basent leurs services sur une description intégrée des données qu'ils gèrent. Ce qui permet à leurs utilisateurs d'accéder aux données comme sur une base de données centralisée, sans avoir s'inquiéter de la localisation physique des données accédées (les sources), ni du type de SGBD qui les gère localement. C'est pourquoi les chercheurs et les entreprises sont très intéressés par l'approche fédérée. **[CHR 1996]**

II.7.5. Inconvénients :

Plusieurs problèmes restent à résoudre avant que cette approche devienne commercialisable.

1 – Problèmes relatifs aux aspects de conception ; comment peut-on construire une vision commune globale et optimale des données partagées.
2 – Problèmes relatifs aux aspects système ; développement de telles techniques adaptées à un environnement distribué .

II.7.6. Quelques solutions proposées :

1 – Concernant le problème de conception, les recherches actuelles en conception portent notamment sur l'intégration des schémas et des bases de données, le travail coopératif, l'évolution des schémas et des bases.
2 – Pour le coté système, les recherches portent sur les nouveaux types de transaction (transactions longues, transactions emboîtées, …), le traitement des requêtes, la sécurité des données, etc.

II.7.7. Définition du processus d'intégration (de fédération) :

Le cœur du problème de conception est le processus d'intégration des bases de données. Ce processus consiste à prendre en entrée un ensemble de bases de données (schémas et populations), et à produire en sortie une description unifiée des schémas initiaux (le schéma intégré) et les règles de traduction (mapping) qui vont permettre l'accès aux données existantes à partir du schéma intégré.

II.7.8. Déférentes actions du processus d'intégration (Différentes étapes) :

1 - Pré-intégration : c'est l'étape dans laquelle les schémas en entrée sont transformés de différentes manières pour les rendre plus homogènes du coté sémantique et syntaxique.

2 - Recherche des correspondances : une étape consacrée à l'identification des éléments semblables dans les schémas initiaux et à la description précise de ces liens inter-schémas;

3 – Intégration : c'est l'étape finale dans le processus d'intégration, elle permet d'unifier les types en correspondance en un schéma intégré et de produire les règles de traduction associées entre le schéma intégré et les schémas initiaux.

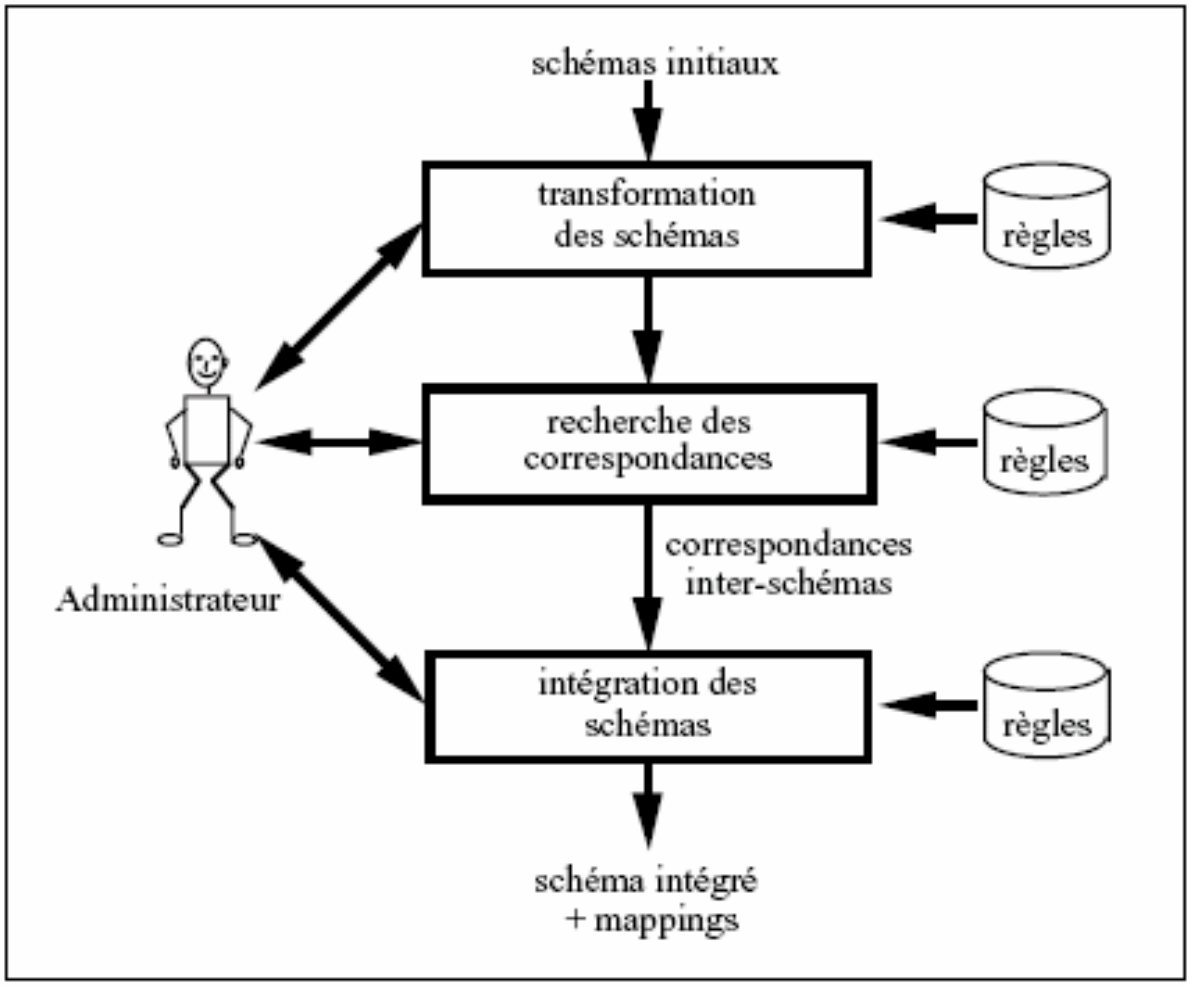


Fig II.25. Le processus global d'intégration

II.7.9. Démarche d'intégration :

II.7.9.1. L'étape de pré-intégration :

Dans la plupart des cas, les bases de données à intégrer auront été développées indépendamment et seront par conséquent hétérogènes.

La différence la plus évidente est lorsque les bases de données existantes ont été installées avec des SGBD utilisant des modèles de données différents (relationnel, Entité-relation, Réseau CODASYL, objets, relationnel objet, ...), c'est le cas ***d'hétérogénéité des modèles.*** **[BEN 2002]**

Mais, même si ces bases de données sont toutes converties dans un même modèle, des différences apparaissent au niveau de la richesse de description de chaque base de données.

En effet, certains modèles de données possèdent dans leurs concepts plus de sémantique que d'autres modèles.

Par exemple, un schéma entité-association utilise des concepts séparés pour modéliser les entités et les associations, tandis que le schéma relationnel équivalent va décrire les entité et les associations de la même manière sans faire distinction entre les deux.

Alors, pour parvenir à un même niveau de compréhension des deux schémas (EA et R), on doit spécifier sur le schéma relationnel, quelles relations décrivent des entités et quelles relations décrivent des associations. On parle ici ***d'une hétérogénéité de schémas***. **[BEN 2002]**

Une autre cause d'hétérogénéité réside dans le non-déterminisme du processus de modélisation des bases de données. Deux concepteurs qui doivent représenter la même réalité (le même univers) en utilisant le même modèle de données vont inévitablement créer deux schémas différents.

- ***Pré-intégration et résolution des problèmes dus à l'hétérogénéité (résolution des conflits) :***

a) - Problèmes dus à l'hétérogénéité des modèles de données :

Le problème consiste à transformer des structures de données et des opérations d'un SGBD dans les structures de données et les opérations d'un autre SGBD dans une phase préliminaire dans le processus d'intégration (lors de la phase de pré-intégration). **[CHR 1996]**

La grande majorité des chercheurs en intégration supposent que les schémas initiaux sont tous exprimés dans un même modèle de données unifié, appelé modèle commun ou modèle pivot sur lequel le logiciel d'intégration est construit.

L'étape de traduction (des schémas) devient un pré-requis indispensable à l'intégration. Les traductions requises sont celles entre les modèles de données locaux et le modèle commun.

Malheureusement, pour établir cette traduction, il existe peu d'outils capables d'effectuer ces traductions automatiquement (***problème 1***), sauf pour passer d'un schéma entité-association à un schéma relationnel).

Les développements en cours se focalisent sur les traductions entre modèles orienté-objets et relationnel (***solution 1***). **[CHR 1996]**

Quelques chercheurs ont aussi étudié le problème complémentaire de la traduction des opérations dans le but d'obtenir un système totalement polyglotte, c.à.d un système dans lequel chaque participant parle son propre langage.

L'un des points non résolus, est le choix du modèle de données commun. (***problème 2***).

Typiquement, deux choix ont leurs partisans. La majorité préfère actuellement le modèle orienté-objets, car il possède tout les concepts sémantiques des autres modèles

à cause sa richesse, et que des méthodes peuvent être utilisées pour implanter des règles de traduction spécifiques (***solution 2***). **[CHR 1996]**

Un autre problème est de trouver lequel des nombreux modèles orienté-objets existants est le meilleur dans ce rôle (***problème 3***).

Un autre problème est que plus un modèle est riche en concepts, plus que le processus d'intégration sera complexe, puisqu'il devra résoudre les nombreuses divergences dues aux différents choix de modélisation faits par les divers concepteurs (***problème 4***).

Pour rendre l'intégration plus simple, l'alternative est de choisir un modèle de données avec un minimum de sémantique, et dans lequel la représentation des données est assurée par des faits élémentaires pour lesquels il n'existe pas d'alternative de modélisation. Les modèles binaires sont alors les meilleurs candidats comme modèle commun (***solution 3 + solution 4***). **[CHR 1996]**

Des chercheurs étudient la spécification d'outils génériques, capables de réaliser n'importe quelle traduction.

Les outils traditionnels mettent en jeu des algorithmes spécifiques permettant de traduire des schémas exprimés dans un modèle $\mathbf{M_i}$ en des schémas exprimés dans le modèle commun $\mathbf{M_C}$ et vice versa. **[CHR 1996]**

L'ajout d'un nouveau modèle de données soit $\mathbf{M_j}$ à la fédération implique l'ajout de deux nouveaux algorithmes de traduction :

- L'un pour traduire $\mathbf{M_j}$ à $\mathbf{M_C}$.
- L'autre pour traduire $\mathbf{M_C}$ à $\mathbf{M_j}$.

Les approches génériques utilisent un méta-modèle comme modèle pivot pour les traductions, il s'agit d'un modèle adapté à la description des différents modèles de données (un modèle de classe plus supérieure).

Le méta-modèle connaît tout les concepts de modélisation et sait comment transformer des structures de données entre elles.

Ajouter un nouveau modèle $\mathbf{M_j}$ signifie dans ce cas décrire les concepts de $\mathbf{M_j}$ en utilisant les concepts du méta-modèle.

Traduire un schéma d'un modèle $\mathbf{M_i}$ dans un modèle $\mathbf{M_j}$ (passage entre modèles) se ramène à un processus de restructuration de données à l'intérieur de la base de données du méta-modèle.

b) - Problèmes dus à l'hétérogénéité des puissances d'expression :

Les traductions soulèvent des problèmes difficiles tels que :

- Les modèles de données ne peuvent exprimer toute la sémantique du monde réel (***problème 1***). L'incomplétude de leurs spécifications (en particulier sémantiques) conduit à des ambiguïtés sur l'interprétation d'un schéma. L'intervention de l'administrateur de données est nécessaires pour résoudre ces ambiguïtés, et acquérir ainsi des informations supplémentaires sur la sémantique d'un schéma, c'est ce qu'on appelle processus d'enrichissement sémantique (***solution 1***). **[CHR 1996]**
- D'autres problèmes surviennent dans la compréhension des schémas pauvres en sémantique. Le cas le plus difficile se présente lorsque les données sont stockées dans des fichiers, mais les bases de données relationnelles posent un sérieux défi pour leur ré-interprétation avec des concepts plus sophistiqués (avec les concepts orientés-objets par exemple) (***problème 2***).

Des techniques de rétro-ingéniérie sont actuellement à l'étude pour permettre de distinguer parmi les relations qui représentent des objets et leurs attributs, et celles qui représentent des liens de généralisation/spécialisation (***solution 2***). **[CHR 1996]**

Il s'agit d'un processus d'analyse qui exploite toutes les informations que peut donner le schéma : clés primaires, clés candidates, clés externes et dépendances.

L'enrichissement sémantique est typiquement un processus de décision humain, utilisé pour fournir plus d'informations soit sur la manière dont un schéma décrit la réalité, soit sur le schéma lui-même.

Exemple : un premier niveau d'enrichissement sémantique est la définition explicite de la sémantique d'un objet :

L'objet "Employé" est défini comme les personnes qui ont un contrat de travail avec une entreprise.

c) - Problèmes dus à l'hétérogénéité des modélisations :

Comme nous l'avons mentionné, le processus de modélisation des bases de données n'est pas déterministe (les schémas dépendent du point de vue du concepteur). Ce qui exige une richesse sémantique du modèle de données (***problème 1***).

Ces différences sémantiques peuvent être réglées en imposant des règles de modélisation y compris des règles terminologiques, ainsi que des règles de normalisation qui sont bien connues dans le contexte relationnel, mais doivent encore être élaborées pour les bases de données orienté-objets. **[CHR 1996]**

II.7.9.2. Etape d'identification des correspondances :

Après avoir établir la conformité demandée sur les schémas initiaux, on procède à l'identification des éléments communs des bases existantes.

Nous disons que deux bases de données ont quelques choses en commun si les portions du monde réel qu'elles représentent ont des éléments communs, ou ont des éléments en relation entre eux par un lien d'intérêt pour des applications futures. **[BEN 2002]**

Ce dernier cas est, par exemple, celui d'une entreprise multi-filiales où la base de données de chaque filiale enregistre les employés de la filiale (et/où) l'on voudrait voir dans la base de données intégrée un type d'objet « Employé » représentant la population globale des employés de toutes les filiales.

Nous disons aussi que deux éléments (tuple, occurrence, valeur, …) de deux bases de données sont en correspondance s'ils décrivent le même élément du monde réel (objet, propriété, lien). **[CHR 1996]**

La définition d'une correspondance est appelée aussi une assertion de correspondance inter-schémas. **[CHR 1996]**

Le processus d'intégration consiste à trouver ces correspondances entre les éléments des bases de données existantes, et d'associer pour chaque correspondance un élément global virtuel (non matérialisé) qui inclut toutes les informations utilisables sur cet élément.

La base fédérée enregistre uniquement la définition de l'élément global et des règles de traduction entre l'élément global et ses correspondants locaux. Ces règles font partie du résultat du processus d'intégration.

Les spécifications suivantes doivent être fournies pour définir une assertion de correspondance inter-schémas :

a/ Quels sont les éléments en correspondance ?
b/ Comment leurs extensions sont liées ?
c/ Comment les instances correspondantes sont identifiées dans les extensions
d/ Comment leurs représentations sont liées ?

a) - Quels sont les éléments en correspondance ?

Les éléments en correspondance peuvent être soit des occurrences ou des valeurs (par exemple : auteurs, articles), soit des chemins reliant des occurrences ou des valeurs (par exemple le chemin liant un auteur aux articles qu'il a écrit).

Exemple :

- Comme exemple sur le premier cas, soit l'assertion qui peut relier les occurrences S1.*Auteur* à S2.*Article.auteurs*, car les deux éléments représentent des personnes qui ont écrit un article.
- comme exemple sur le deuxième cas, soit l'assertion reliant S1.(*Auteur-Ecrit-Article*) à S2.(*auteurs-Article*) pour spécifier que ces deux chemins ont la même sémantique (envisagent la même réalité).

Cela signifie que, si dans le schéma S1 l'auteur X a écrit l'article Y et si X et Y appartiennent aussi au schéma S2, alors obligatoirement la valeur X apparaît dans l'attribut *auteurs* de l'*Article* Y.

b) - Comment leurs extensions potentielles sont liées ?

Pour construire le schéma intégré, la relation entre les extensions des éléments, dans le monde réel, doit être connue. Ces extensions sont considérées comme des ensembles d'objets du monde réel.

Pour décrire une telle assertion on utilise des opérateurs ensemblistes tels que :
L'équivalence (≡), l'inclusion (⊇), l'intersection (∩), disjonction (≠) .
Par exemple, on a :

S1.*Article* ⊇ S2.*Article*

c.à.d tous les articles représentés dans S2 sont dans S1 la réciproque est fausse. De même pour :

S1.*Conférence* ≡ S2.*Conférence.actes*

c) - Comment les instances en correspondance sont identifiées ?

Quand l'utilisateur cherche à accéder à un objet via le schéma intégré, il peut accéder à certaines de ses propriétés dans une base de données et à certaines d'autres dans une autre base de données.

Le système doit savoir trouver dans une base de données l'objet (occurrence/ valeur) correspondant à un objet donné (occurrence/valeur) d'une autre base.

Pour ce faire, chaque assertion doit comprendre une clause qui spécifie la correspondance entre instances, c'est la clause "Avec Identifiants Correspondants" (AIC).

Exemple :

S1.*Article* ⊇ S2.*Article* AIC *titre*

qui spécifie que les deux objets « articles » des deux bases de données ont un identifiant commun, qui l'attribut « *titre* ».

d) - Comment les représentations sont liées ?

Les éléments en correspondance possèdent généralement des propriétés communes, autres que celles utilisées pour leur identification.

Pour ce faire, et pour éviter la duplication des propriétés communes dans le schéma global, une assertions décrit de telles correspondances en utilisant la clause "Avec Attributs Correspondants" (AAC).

Exemple :

S1.Conférence ≡ S2.Conférence.actes AIC ISN AAC année, N°

Signifient que ***S1.Conférence***, ***S2.Conférence.Actes*** partagent en plus l'identifiant ISN, les propriétés : année, N°

Le format général pour décrire une correspondance entre deux propriétés A et B est :

f(A) R g(B)

avec : R est une relation ensembliste (≡,⊇, ∩, ≠)
f et g sont deux fonctions utilisées, si nécessaire, pour résoudre un conflit de représentation.

- ***Cohérence des correspondances :***

Etant donné un ensemble d'assertions entre deux bases de données, il convient de tester s'il est cohérent et minimum.

Exemple :

Soit un schéma S contenant un lien de généralisation entre A et B (A est-un B), et un autre lien de même type entre C et D (C est-un D).
A ≡ D, B ≡ C est un exemple d'assertion de correspondance incohérente.

Certaines assertions sont déductibles d'autres.

Par exemple, si A ≡ C est déclarée, les correspondances B ⊇ C et D ⊇ A peuvent alors être déduites.

L'ensemble d'assertions pour l'exemple des bibliothèques est comme suit :

Correspondences S1/S2 : **[BEN 2002] [CHR 1996]**

Auteur ⊇ Auteur AIC Nom
Article ⊇ Article AIC Titre AAC mots-clés
Conférences ≡ Actes AIC ISN AAC année, N°
Publication.Nom ⊇ Conférence.nom AIC nom
Auteur-Ecrit-Article ≡ Auteur-Article
Conférence-publication-contient-article ≡ actes-articles : articles
Publication.Nom-Publication-Conférence ≡ Conférence-actes.

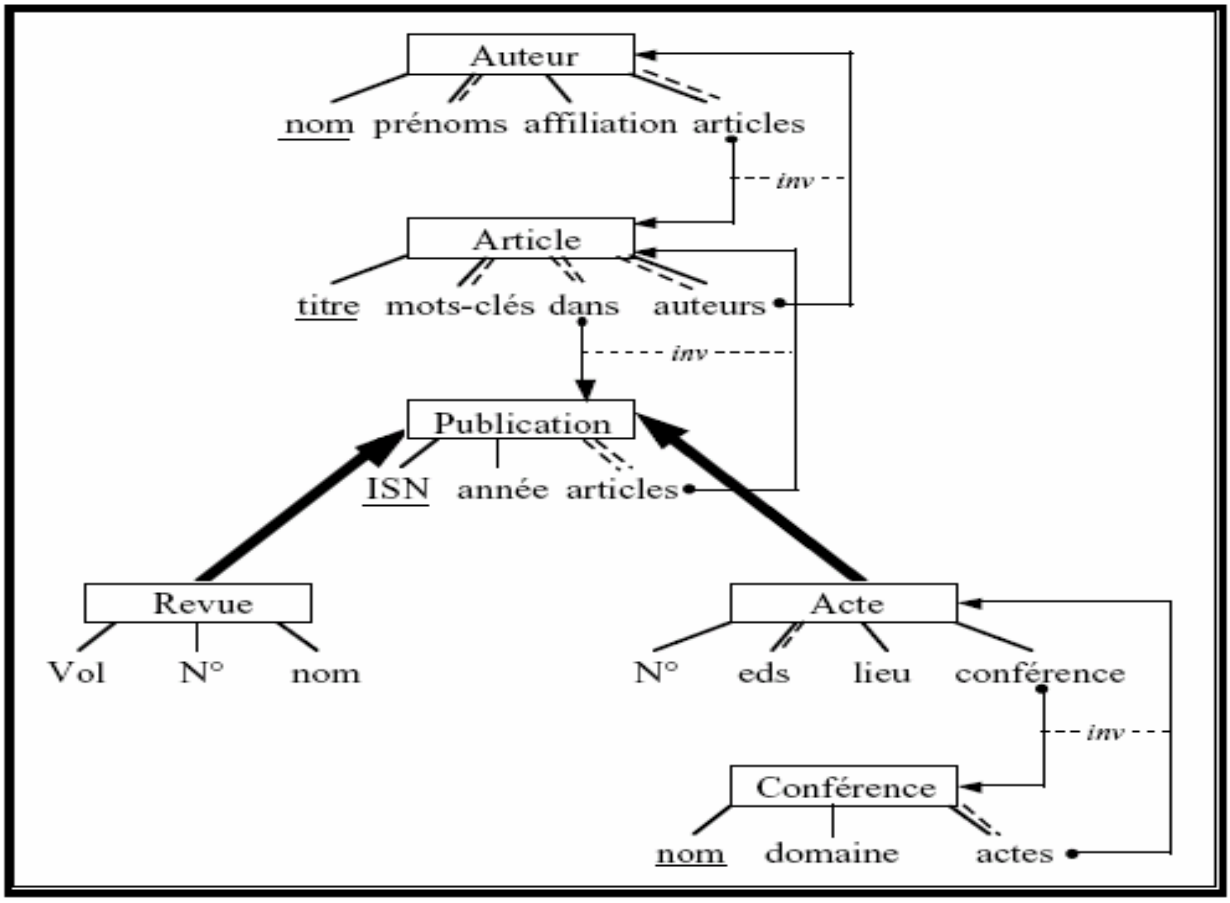


Fig II.26. Schéma intégré Orienté-Objet pour l'exemple de bibliothèque

- ***Recherche des correspondances :***

Lorsque l'intégration porte sur un nombre important de schémas, ou de schémas avec un nombre important de types d'objets, l'identification de toutes les correspondances n'est pas triviale.

Des efforts significatifs ont été faits pour définir des outils pour l'identification automatique des correspondances. **[GOT 1992].**

Ces outils mesurent la similitude entre deux éléments de schémas en recherchant des caractéristiques identiques ou similaires (noms, identifiants, composants, propriétés, attributs, méthodes, …).

L'étape finale est l'interaction avec l'administrateur des données pour confirmer ou infirmer les correspondances, et pour obtenir les informations nécessaires à la définition complète des assertions.

La plupart des outils disponibles n'analysent que les aspects syntaxiques et structurels, et sont par conséquent confrontés aux problèmes usuels des synonymes et homonymes.

Pour limiter ces problèmes, certains outils ont recours à des bases terminologiques qui permettent de décrire les termes utilisés dans le domaine de l'application et les liens qui peuvent exister entre eux **[FAN 1992]**.

- ***Les conflits : taxonomie et solutions :***

Une fois que les correspondances ont été décrites, l'intégration proprement dite peut commencer.

Chaque assertion est analysée pour déterminer quelle est la représentation des éléments en correspondance qui doit être incluse dans le schéma intégré et quelles sont les règles de traduction entre le schéma intégré et les schémas initiaux.

Ces règles seront utilisées pour évaluer des requêtes fédérées et pour transformer chaque requête définie sur le schéma intégré en sous requêtes locales soumises aux SGBD locaux afin de récupérer toutes les informations locales nécessaires pour construire l'information souhaitée.

Lorsqu'une assertion décrit les types en correspondance comme étant identiques, c.à.d ils ont la même représentation et des populations équivalentes. leur intégration est évidente, le type intégré dans ce cas est identique aux types en entrée et le mapping est l'identité.

Si les types en correspondance vont présenter des différences, que ce soit dans leurs représentations ou dans leurs populations, on parle ici d'un conflit.

Plusieurs taxonomies des conflits on été proposées :

Type de conflit	Explication (signification)
Conflit de classification	Les populations du monde réel représentées par les deux types en correspondance sont différentes
Conflit de description	- Les types en correspondances ont des propriétés différentes . - Ces propriétés sont représentées de manières différentes
Conflits de structure	Les concepts utilisés pour décrire les types en correspondance sont différents
Conflits d'hétérogénéité de modèles de données	Les modèles de données utilisés sont différents
Conflits de méta-données	Les deux types en correspondances ne sont pas liés à un même méta-type (ensemble de noms de types)
Conflits de données	Les instances en correspondance ont des valeurs différentes pour les propriétés en correspondance

Tableau II.1. Une taxonomie des conflits entre deux types en correspondance **[CHR 1996]**

Beaucoup de propositions ont été faite pour la résolution des conflits de classification et de description **[KIM 1993]**.

Les conflits de méta-données ont été abordés plus récemment, notamment par **[SAL 1992].**

Les conflits de structure, d'hétérogénéité et de données ont fait l'objet de quelques travaux **[SPA 1991, SHE 1994].**

Il n'existe pas de méthode d'intégration complète qui résolve tous les conflits, aussi moins de méthodes qui puissent être automatisées.

Plus les problèmes dus aux conflits, une autre faiblesse de la recherche actuelle est de ne pas spécifier explicitement le but précis recherché dans cette étape d'intégration proprement dite.

Plusieurs objectifs sont possibles :

- ***Produire des résultats sensiblement différents :*** Par exemple, on peut rechercher la simplicité, et dans ce cas produire un schéma intégré avec un nombre minimal de types d'objets, de façon à ce que le schéma soit lisible facilement.

- La complétude du schéma intégré : au sens où chaque élément des schémas initiaux apparaît dans le schéma intégré. Ceci permet aux administrateurs locaux de retrouver facilement leurs données dans le schéma résultat, et facilite la mise à jour lorsque les schémas locaux évoluent.

- On peut aussi vouloir être exhaustif : en faisant apparaître dans le schéma résultat tous les types d'objets initiaux, les types d'objets dérivables à partir des types d'objets initiaux (définis par leurs intersection/union, …). Ceci alourdit le schéma intégré mais, prépare l'intégration de nouvelles bases de données qui viendraient s'ajouter à la fédération existante.

a) - Conflits de classification et leurs solutions :

Deux types en correspondance ont un conflit de classification lorsqu'il décrivent des ensembles d'éléments du monde réel différents, mais sémantiquement liés .

Exemple : (Bibliothèque centrale)
Le schéma S1 décrit les articles de revues et de conférences, alors que le schéma S2 ne décrit que les articles de conférences.

Un conflit de classification se traduit dans une assertion par l'utilisation d'une relation ensembliste autre que l'équivalence ($\supseteq$, $\cap$, $\neq$).

Solution 01 : une première solution standard pour ces conflits est d'inclure dans le schéma intégré la hiérarchie de généralisation/spécialisation appropriée, entre les éléments en correspondance (voir Tableau 02 (a), (b)) **[LAR 89, GOT 92]**.

Cette solution vise l'objectif de complétude des schémas résultats, les types initiaux sont copiés dans le schéma intégré et reliés par des liens de type (IS-A. – est un) ou (spec/gen) .

Si nécessaire, un sous type (ou supertype) commun est ajouté pour réaliser la connectivité de la hiérarchie de généralisation/spécialisation.

Ceci a pour effet de réduire les traductions entre schéma intégré et schémas initiaux à des simples fonctions d'identité.

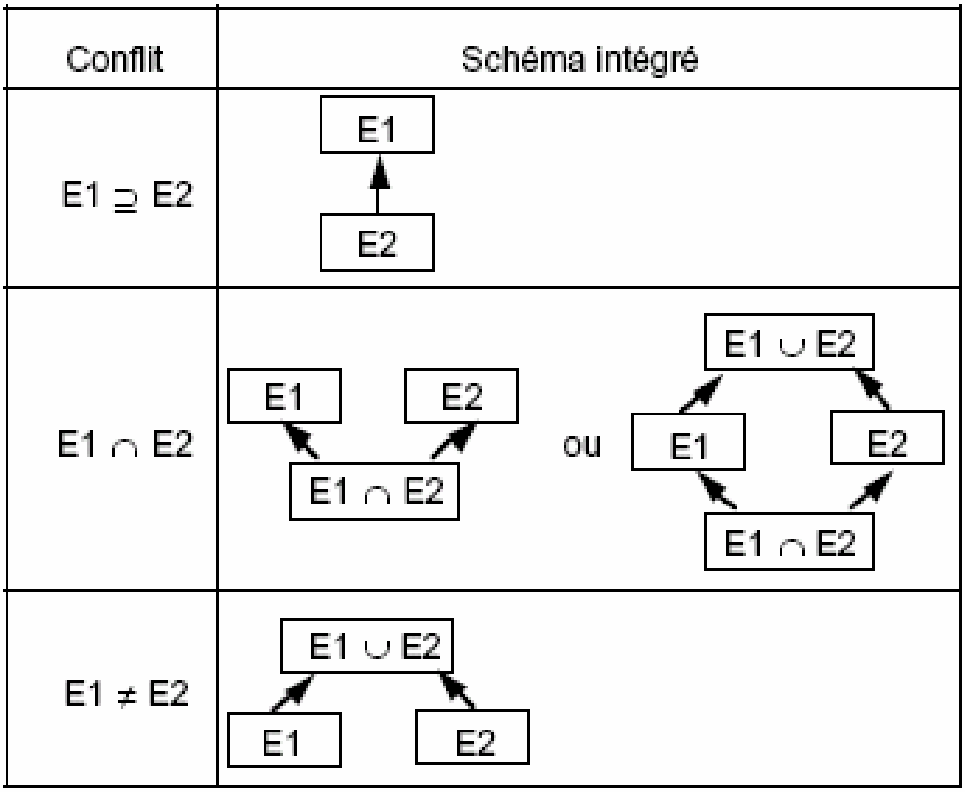


Tableau II.2. a. Résolution des conflits de classification – solution standard (préservation des types locaux) **[CHR 1996]**

Solution 02 : Cette solution est basée sur l'objectif de simplicité, elle consiste à insérer dans le schéma intégré un seul type d'objet (voir tableau 2 (b) -Technique de fusion).

Le schéma intégré décrit alors l'union des extensions. Les règles de traduction utilisent l'opérateur de sélection pour dériver les populations initiales de la population globale (dans une requête par exemple).

Solution 03 : l'objectif de cette solution est de créer un schéma intégré exhaustif (voir tableau 02 (b) Technique exhaustive), en incluant dans le schéma intégré les types initiaux, plus leur union (un super-type), leur intersection, ainsi que les compléments de l'intersection (trois sous-types) (voir tableau 02 (b) autres solution) .

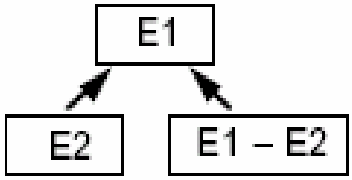
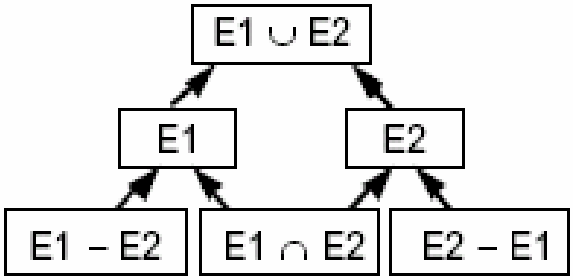
<table>
<tr><th rowspan="2">Conflit</th><th colspan="2">Schéma intégré</th></tr>
<tr><th>Technique de fusion</th><th>Technique exhaustive</th></tr>
<tr><td>$E1 \supseteq E2$</td><td>E1</td><td> 
</td></tr>
<tr><td>$E1 \cap E2$</td><td>$E1 \cup E2$</td><td> 
</td></tr>
<tr><td>$E1 \neq E2$</td><td>$E1 \cup E2$</td><td></td></tr>
</table>

Tableau II.2. b. autres solutions (technique de fusion, technique exhaustive) **[CHR 1996]**

b) - Conflits structurels et leurs solutions :

Deux types en correspondance ont un conflit structurel lorsqu'ils sont décrits en utilisant des concepts de niveaux de représentation différents ou soumis à des contraintes différentes : par exemple, une classe d'objets et un attribut, ou un type d'entité et un type d'association.

Ce type de conflit a été peu étudié dans la littérature et uniquement pour des modèles de données particuliers **[KIM 1993, SHO 1991]**.

La solution des conflits structurels **[SPA 1991]** part du principe que le schéma intégré doit pouvoir décrire la population des deux éléments en conflit.

L'élément intégré doit donc refléter les possibilités des éléments initiaux : en termes de capacité à décrire l'information (on adopte la borne supérieure minimale) et en termes des contraintes implicites et explicites attachées aux éléments (on adopte la borne inférieure maximale).

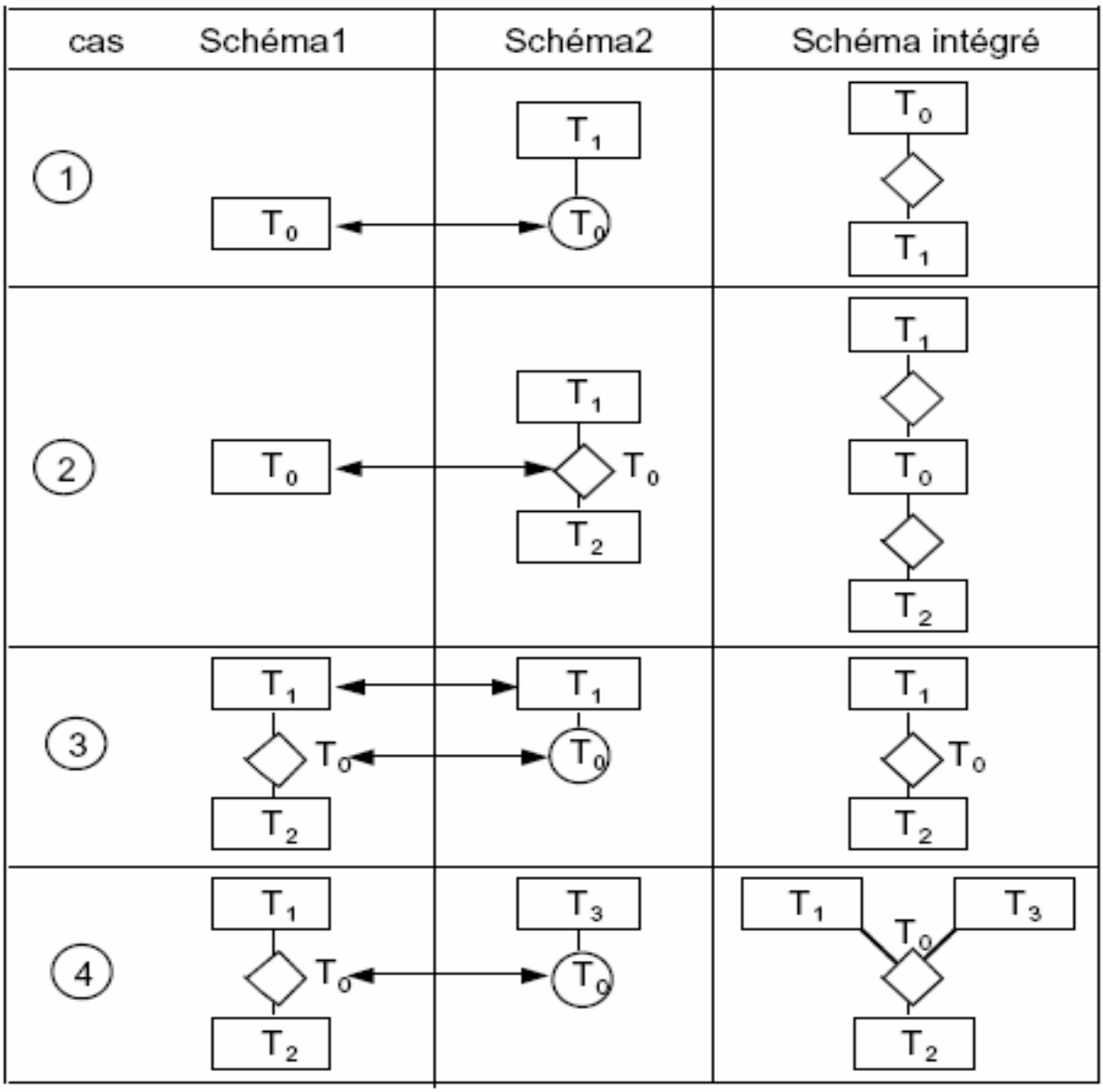
Par exemple si une assertion entre schémas relationnels met en correspondance une relation (valeur + liens) et un attribut (valeur ou lien), l'élément correspondant dans le schéma intégré sera une relation.

Parmi les contraintes qu'il faut prendre en compte, les contraintes relatives aux différentes cardinalités et les dépendances d'existence. Par exemple, l'existence d'un attribut dépend de celle de son propriétaire, tandis qu' un objet n'est pas, soumis à des contraintes d'existence.

Exemple : Une assertion met en correspondance ***une classe d'objets*** (**S1.Auteur**) avec ***un attribut*** (**S2.Article.auteurs**), ***c'est la classe qui est inscrite dans le schéma intégré*** (la borne inférieure maximale est dans ce cas l'absence de contraintes).

Les conflits structurels peuvent être simples comme ils peuvent être complexes, ils peuvent par exemple faire appel à des expressions de jointure **[KIM 1993, DUP 1994]**.

Une solution générale reste à trouver (voir tableau 03 – solutions des conflits structurels).



Ces diagrammes sont exprimés sur la base d'un modèle de données générique, où:

- représente un type d'objet (classe, type d'entité, relation, ou type d'article)
- représente une relation entre types d'objets (attribut référence, type de relation, clé externe, ou set CODASYL)
- représente un attribut
- ⟷ représente une assertion de correspondance de type équivalence

Tableau II.3. Solutions des conflits structurels **[CHR 1996]**

c) - Conflits descriptifs et leurs solutions :

Un conflit descriptif survient lorsque les types en correspondance ont des propriétés différentes. Ces conflits ont été étudiés **[LAR 1989, KIM 1993]**. Soit pour proposer une résolution automatique, soit pour guider un intégrateur humain, et on distingue plusieurs taxonomie des conflits :

1. Les types d'objets en correspondance peuvent différer selon leurs :

- Noms : homonymes et synonymes.
- Clés : clé primaire ou clés candidates sont différentes.
- Attributs : - les ensembles d'attributs des types en correspondance sont différents.
  - Les attributs en correspondance sont en conflit
- Méthodes : - les ensembles des méthodes des types en correspondance sont différents
  - les méthodes en correspondance sont différentes.
- Contraintes d'intégrité : sont différentes.
- Droits d'accès : sont aussi différents.

2. Les attributs en correspondance peuvent différer selon leurs :

- Noms : homonymes et synonymes.
- Portées : locale à une base de données ou globale, (c.à.d conserve sa signification dans la base de données intégrée).
- Structures : - simple ou complexe (composée d'autres attributs).
  - Cardinalités : mono-valué ou multivalué, facultatif ou obligatoire
- Types de données : différentes échelles, différentes unités, différentes opérations, différentes valeurs par défaut
- Contraintes d'intégrité : différentes.
- Droits d'accès : différents.

3. Les méthodes en correspondance peuvent différer selon leurs :

- Noms : homonymes et synonymes
- Signatures : nombre de paramètres différents , types de paramètres différents, résultats différents.

Les solutions traditionnelles proposées pour résoudre les conflits descriptifs sont :

- Conflit de nom : - renommer en utilisant l'un des noms (en cas de synonyme).
  - renommer en utilisant un préfixe (en cas d'homonyme), ce préfixe est généralement le nom de la base de données source.
- Conflit de type de données : une fonction de conversion est utilisée pour passer du domaine d'un attribut au domaine de l'autre attribut.
- Conflit de clé : une fonction de conversion est utilisée pour associer à chaque valeur d'une clé une valeur de l'autre clé.
  Cette fonction doit être bijective (1-1) pour être opérationnelle.
- Contraintes d'intégrité : utiliser l'approche de la borne inférieure maximale.
- Structures d'attributs : aucune solution vraiment disponible.
  Les méthodologies actuelles traitent :
  - les attributs simples monovalués.
  - rarement les attributs multivalués **[LAR 1989]**.
  - Ignorent les attributs complexes.

d) - Conflits d'hétérogénéité des modèle de données :

Les méthodologies d'intégration actuelles n'intègrent que des schémas exprimés dans leur propre modèle de données (le modèle commun ou pivot). Donc les schémas qui ne sont pas déjà exprimés dans ce modèle doivent être traduits pendant la phase de pré-intégration, avant d'être pris en considération pour intégration.

Les auteurs montrent que les différents conflits et leurs solutions sont les mêmes quelque soit le modèle de données, d'ou, Il devrait ainsi être possible de définir l'ensemble des règles fondamentales et nécessaires d'intégration, puis reformuler chaque règle selon la particularité du modèle considéré.

On peut alors construire un outil capable d'assurer directement l'intégration de schémas hétérogènes et de produire un schéma intégré dans n'importe quel modèle.

e) - Conflits de données/méta-données :

Les conflits de ce type surviennent lorsqu'une donnée dans un schéma d'une base de données est en correspondance avec une méta-donnée (le nom d'un type) dans le schéma d'une autre base.

Exemple : soient des bases de données relationnelles de gestion de bourse définies comme suit : **[CHR 1996]**

BD1 : Cours (date , action , cotation)

951001	IBM	77.72
951002	IBM	79.23
.....	.....	.....
951001	HP	60.02
951002	HP	61.45
.....	.....	.....

BD2 : Cours (date , IBM , HP , ……..)

951001	77.72	60.02 , …...
951002	79.23	61.45 , …...
.....	.....	....

BD3 : IBM (date , cotation)

951001	77.72
951002	79.23

HP (date , cotation)

951001	60.02
951002	61.45
.....	....

Dans cet exemple une valeur pour l'attribut action dans BD1, correspond à un nom d'attribut dans BD2 et à un nom de relation dans BD3. Cela crée un conflit.

Dans les langages relationnels traditionnels on peut pas transformer un nom d'attribut ou de relation en valeur (passer de BD2 ou BD3 à BD1). Mais on peut passer d'un nom d'attribut à un nom de relation (passer de BD2 à BD3) et inversement, on peut également passer d'un nom de valeur à un nom d'attribut ou un nom de relation (passer de BD1 à BD2 ou BD3).

Pour ce fait, et pour résoudre ce genre de conflits, il est indispensable d'avoir un langage de correspondance qui permet la manipulation simultanée des données et des méta-données.

L'algèbre relationnelle étendue de **[SAL 1992]** et la logique d'ordre supérieure de **[LAK 1993]** sont des exemples de tels langages.

Les approches orienté-objets fourniront des opérations de transformation de schéma pour effectuer toutes les mises en correspondance nécessaires, soit en partitionnant une classe en sous-classes selon la valeur d'un attribut discriminant (passage de BD1 à BD3 selon l'attribut action), soit en créant une super-classe commune à plusieurs classes avec un nouvel attribut discriminant (passage de BD3 à BD1 avec un nouvel attribut action).

f) - Conflits de données :

Ce type de conflit survient lorsque des occurrences en correspondance ont des valeurs en conflit ; par exemple, un même article est stocké dans les deux bases de données de la bibliothèque avec des mots clés différents.

Les causes éventuelles de ce conflit peuvent être :

- Erreurs de saisie.
- Sources d'information différentes.
- versions différentes de la base de données.
- Mises à jours différées, … etc.

Les conflits de données sont normalement détectés lors de l'exécution d'une requête d'accès aux données.

Le système peut se contenter de signaler l'erreur à l'utilisateur, mais il peut aussi appliquer une heuristique pour produire une valeur appropriée.

Les heuristiques courantes sont :

- Choisir la valeur dans la base qui est la plus "crédible".
- Unifier les valeurs en conflit d'une certaine façon (agrégation des valeurs simples, union des ensembles de valeurs).
- Fournir aux utilisateurs un langage pour manipuler eux-mêmes toutes les valeurs candidates, l'ensemble des valeurs possibles étant la réponse à la requête chaque fois que le conflit survient **[TSE 1993]**.

II.7.9.3. L'étape d'intégration :

- ***Stratégies d'intégration :***

Jusqu'à maintenant, Nous avons analysé les principes et les techniques d'intégration des schémas. Nous allons aborder la possibilité d'adopter des différentes stratégies pour diriger le processus d'intégration. Plusieurs choix sont possibles :

1) - Processus manuel : Puisque seul l'administrateur connaît de façon certaine la sémantique des données, les stratégies manuelles laissent l'administrateur diriger tout le processus d'intégration.

Les méthodes manuelles ou sem-automatiques fournissent un langage de manipulation de schémas que l'administrateur va utiliser pour construire lui-même (si le langage est procédural) ou spécifier (si le langage est déclaratif) le schéma intégré.

Les langages procéduraux offrent des primitives de transformation de schéma qui permettent de restructurer les schémas initiaux jusqu'à ce qu'ils puissent être fusionnés en un seul schéma.

Le système génère automatiquement les règles de traduction entre les schémas initiaux et le schéma intégré **[MOT 1987]**.

Les langages déclaratifs sont plus faciles à utiliser, mais l'établissement des règles de traduction est plus délicat (difficile).

Les stratégies manuelles supposent que l'administrateur sache comment structurer le schéma intégré.

2) - Stratégies semi-automatiques :

Ont pour but de construire un outil qui effectue automatiquement l'intégration lorsque les assertions de correspondance ont été définies.

L'outil détermine aussi automatiquement les règles de traduction **[SPA 1991]**.

L'administrateur est seulement impliqué dans l'identification des assertions de correspondance, et éventuellement dans le choix de la stratégie d'intégration.

3) - Stratégie en un coup ou par incréments :

Dans cette stratégie, l'intégration est vue comme un processus atomique qui produit un résultat final.

Cette stratégie paraît préférée dans le cas des applications réelles et complexes pour lesquelles l'intégration peut être un processus très long et pénible qui provoque la suspension d'exploitation des bases de données en attendant la fin de l'intégration.

Les stratégies par incréments permettent d'effectuer graduellement l'intégration tout en continuant à utiliser les systèmes existants.

Par exemple elles permettent d'intégrer deux occurrences en correspondance ou deux types en correspondance chaque fois qu'une telle correspondance est détectée et semble intéressante.

En d'autre mots, l'intégration peut être effectuée petit à petit (au fur et à mesure) au fil des jours **[SCH 1992]**.

4) - Stratégie d'intégration à partir d'un thesaurus :

Cette méthode d'intégration suppose qu'un nouveau schéma intégré est construit directement à partir des schémas initiaux.

Si une nouvelle base doit être intégrée après que le schéma intégré ait été construit, le processus est reconduit avec comme schémas initiaux le nouveau schéma et le schéma intégré existant.

Une alternative consiste à donner un rôle particulier au schéma intégré existant, celui de thesaurus de toute la connaissance de l'entreprise.

Si des nouvelle bases sont à prendre en compte, elles sont utilisées uniquement pour enrichir le thesaurus avec les connaissances supplémentaires qu'elles peuvent contenir.

En d'autres mots, elles sont absorbées par le thesaurus. Et en cas de conflit, c'est le thesaurus qui prime sur les nouvelles bases **[COL 1991].**

II.7.10. Conclusion :

L'intégration de bases de données est certainement une tâche complexe. C'est pourtant une tâche que les entreprises peuvent difficilement éviter si elle veulent mettre en route de nouvelles applications ou réorganiser le système d'information existant pour une meilleure productivité. Nous avons montré que les problèmes à résoudre sont bien identifiables et que des solutions existent pour nombre d'entre eux.

Bien que la recherche ait été active dans ce domaine depuis vingt ans, avec une croissance significative des efforts ces dernières années, plusieurs problèmes importants doivent encore être résolus. A titre d'exemple nous citerons: l'intégration des objets complexes (tels qu'on les trouve dans les bases orienté-objets), les correspondances complexes entre plusieurs types (et non plus uniquement entre deux types), la prise en compte des contraintes d'intégrité et des méthodes, l'intégration directe de bases hétérogènes. Des études théoriques sont nécessaires pour valider les règles d'intégration et leurs propriétés (commutativité, associativité, ...). Il est donc important que l'effort pour résoudre les problèmes d'intégration soit poursuivi et que les méthodes proposées puissent être évaluées lors d'expérimentations avec des applications réelles.

II.8 . Approche entrepôt de données (Data warehouse) :

II.8.1. Préface :

Les sociétés de téléphone gardent au moins un an les positions géographiques et les consommations de leurs abonnés << mobiles >>. Les grands magasins et les entreprises de Vente Par Correspondance (VPC) conservent les achats de leurs clients (tickets de caisse en grande distribution, commandes en VPC), collectent des informations sur leurs clients grâce à des systèmes de cartes de fidélité ou de crédit, et achètent des bases de données géographiques et démographiques.

Les sites web conservent des traces de connexions sur leurs sites marchands. En résumé, les entreprises en secteur très concurrentiel conservent les données de leur activité et achètent même des données.

Les motifs qui ont présidé à la conservation de ces données étaient : des obligations légales pour pouvoir justifier les facturations, des raisons de sécurité pour pouvoir détecter les fraudes, des motifs commerciaux pour suivre l'évolution des clients et des marchés. Ce constat, valable pour les sociétés du secteur marchand, peut être étendu à de nombreux domaines comme la médecine, la pharmacologie. Il faut donc définir des environnements permettant de mémoriser de grands jeux de données et d'en extraire de l'information.

Les structures qui accueillent ce flot important de données sont des entrepôts de données ou *data warehouse*. Ils sont construits sur une nouvelle architecture permettant d'extraire l'information, est une architecture bien différente de celle prévue pour l'informatique de production, basée elle sur des systèmes de gestion de bases de données relationnelles et des serveurs transactionnels. Un entrepôt de données est construit en l'alimentant *via* les serveurs transactionnels de façon bien choisie et réfléchie pour permettre aux procédures d'extraction de connaissances de bien fonctionner. L'organisation logique des données est particulièrement conçue pour autoriser des recherches complexes. Le matériel est évidemment adapté à cette utilisation.

L'approche d'entrepôt de données consiste donc à spécifier une machine puissante souvent parallèle, avec des middlewares de collecte de données (logiciels intermédiaires de collecte de données) et des outils d'analyse afin de gérer les données de l'entreprise.

II.8.2. Systèmes de production et systèmes décisionnels : **[HAC 2002]**

Un système de production représente une activité constante constituée de : modifications et d'interrogations fréquentes des données par de nombreux utilisateurs (ajouter une commande, modifier une adresse de livraison, rechercher les coordonnées d'un client, ...).

Il faut garder la cohérence des données en interdisant la modification simultanée d'une même donnée par deux utilisateurs différents. Il s'agit donc de privilégier un enregistrement rapide et sûr des données.

À l'inverse, les utilisateurs des systèmes d'information décisionnels n'ont aucun besoin de modification ou d'enregistrement de nouvelles données. Ils vont plutôt interroger le système d'information et les questions posées seront de la forme :

<< ***quelles sont les ventes du produit X pendant le trimestre A de l'année B dans la région C*** >>.

Une telle interrogation peut nécessiter des temps de calcul importants. Il faut donc prévoir une nouvelle organisation qui permette de mémoriser de grands jeux de données et qui facilite la recherche d'informations.

II.8.2.1 . Système de production :

Le modèle Entité-Association (ou modèle EA) est l'un des formalismes les plus utilisés pour la représentation conceptuelle des systèmes d'information de production à cause sa simplicité et son efficacité.

D'un point de vue logique, ce sont les bases de données relationnelles qui ont su au mieux représenter ces modèles conceptuels.

Conserver la cohérence de la base de données, c'est l'objectif et la difficulté principale pour l'informatique de production.

Les systèmes transactionnels (à temps réel) OLTP (*On-Line Transaction Processing*) garantissent l'intégrité des données.

Les utilisateurs accèdent à des éléments de la base par de très courtes transactions indécomposables, isolées. Ils y accèdent très souvent pour des opérations d'ajout, suppression, modification mais aussi de lecture.

Les contrôles effectués sont élémentaires. Les temps de réponse seront acceptables (inférieurs à la seconde) dans un environnement avec de nombreux utilisateurs. Et pour garantir des performances, une requête dont le calcul prendrait trop du temps est inacceptable.

Par exemple, une jointure sur de grosses tables à l'aide de champs non indexés est interdite. Les travaux sur une telle base sont souvent simples et répétitifs, et dès qu'un travail plus important est nécessaire, l'intervention des programmeurs et d'administrateurs de la base est requise et une procédure *ad-hoc* est créée et optimisée.

L'informatique de production a donc été conçue pour privilégier les performances de tâches répétitives, prévues et planifiées orientées vers la production de documents standards (factures, commandes...).

L'intervention de l'utilisateur est guidée à travers des outils spécifiques proposés par une équipe de développeurs.

Une dernière caractéristique des bases de données de production est qu'elles conservent l'état instantané du système. Dans la plupart des cas, l'évolution n'est pas conservée. On conserve simplement des versions instantanées pour la reprise en cas de panne et pour des raisons légales.

II.8.2.2. Système décisionnel :

Dans un système d'informations décisionnel, l'utilisateur final formulera des questions du type :

- ***Comment se comporte le produit X par rapport au produit Y ?***
- ***Comment se comporte le produit X par rapport à l'année précédente ?***
- ***Quel type de client peut bien acheter le produit Z ?***

Ces questions permettent de mettre en évidence les faits suivants :

- Les questions doivent être formulées dans le langage de l'utilisateur en fonction de son << métier >> ou secteur d'activité (centre d'intérêt).
- La prévision des interrogations est difficile car elles sont du ressort de l'utilisateur. De plus, ses questions vont varier selon les réponses obtenues (Evoluent au cours du temps en fonction des réponses obtenues).
- Des questions ouvertes (exemple : les profils du client envers le produit Z) vont nécessiter la mise en place de méthodes d'extraction d'informations.

Une caractéristique des besoins, est la possibilité de poser une grande variété de questions au système, certaines prévisibles et planifiées et d'autres imprévisibles.

- Pour les entrepôts de données, on recherche toujours plus de lisibilité, de simplicité que dans le cas des SGBD.
- La modélisation introduit les notions de *fait* et *dimension (on va les détailler par la suite)*.
- Avec les entrepôts de données, Il sera souvent nécessaire de : filtrer, d'agréger, ,compter, sommer et réaliser quelques statistiques élémentaires (moyenne, écart-type,...).
- La structure logique doit être prévue pour rendre aussi efficace que possible toutes ces requêtes.
- On est aussi amené à introduire de la redondance dans les informations stockées en mémorisant des calculs intermédiaires (par exemple, stocker toutes les sommes de ventes par produit ou par année), ce qui est contradictoire avec le principe de non redondance des bases de production.
- Si le critère de cohérence semble assuré avec les techniques du transactionnel, il apparaît toute relative. Cette cohérence se contrôle au niveau de la transaction élémentaire mais pas au niveau global et des activités de l'organisation.
- Pour les entrepôts, on requiert une cohérence interprétable par l'utilisateur lui même. Par exemple, si les livraisons d'un produit n'ont pas toutes saisies dans le système, on peut pas garantir la cohérence de l'état du stock .

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 très importante des entrepôts, et qui envisage une différence fondamentale avec les bases de production, est qu'aucune information n' est jamais modifiée. En effet, on mémorise toutes les données sur une période donnée et terminée, il n'y aura donc jamais à remettre en cause ces données car toutes les vérifications utiles auront été faites lors de l'alimentation.

L'utilisation des entrepôts se résume donc à :

- Un chargement périodique.
- Des interrogations non régulières, non prévisibles, parfois longues à exécuter.

II.8.3. Définition d'un entrepôt de données (data warehouse) :

Un entrepôt de données (Data Warehouse) est défini comme un ensemble de données orientées sujets (classées selon leurs sujets ou centres d'intérêts), organisées, coordonnées, intégrées, variables dans le temps et non volatiles (disponible tout le temps) Ces données sont orientées vers le processus d'aide à la décision (marketing, commerce électronique, …).

En autre terme, un entrepôt de données est une réalisation particulière d'une grosse base de données (généralement une base de données fédérée) qui organise des données opérationnelles et historiques d'une entreprise afin de faciliter l'analyse et l'interrogation

complexe de ces données. Alors l'entrepôt de données répond à tous les problèmes des données dispersées sur de multiple systèmes hétérogènes ; notamment l'intégration de ces données.

Le marché des ED est estimé à plus de 20 millions de dollars avec une progression de 10% à 20% par an.
95% des 1000 grandes entreprises américaines sont équipées d'un ED **[HAC 2003]**

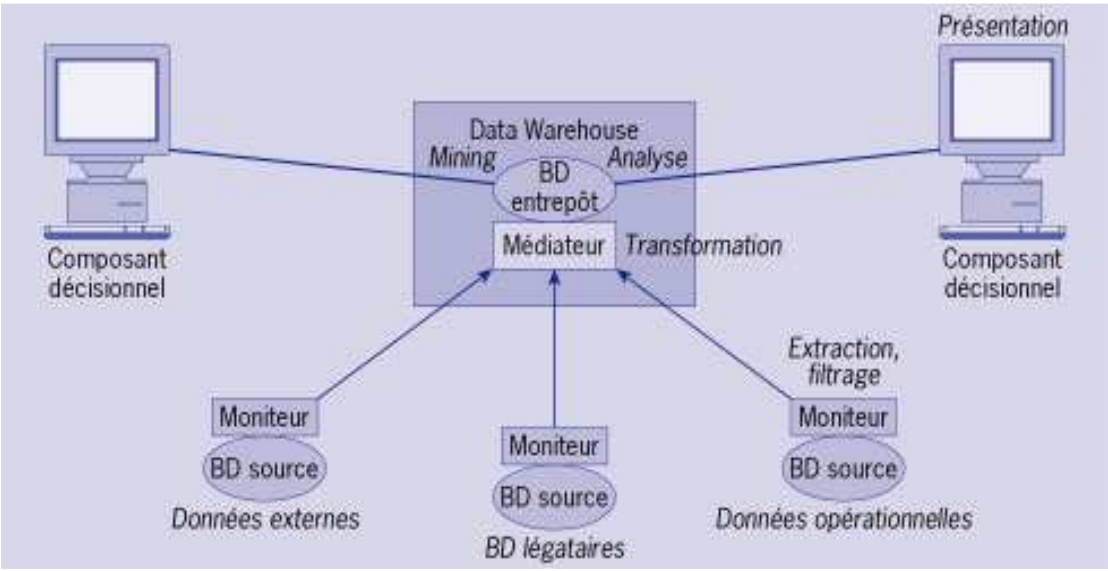


Fig II.27. Architecture d'un entrepôt de données

Un entrepôt de données est entièrement construit selon une approche dimensionnelle, c.à.d qui fait appel aux techniques qui favorise l'analyse multidimensionnelle des données.

La réalisation d'un ED donne naissance à des bases métiers par filtrage thématique et fonctionnel et en prenant compte aux profils utilisateurs, à noter :

- Une base de métier (data mart) est défini par R.Kimball comme un sous ensemble logique du data warehouse, et pour mieux répondre aux différents besoins des utilisateurs, on essaye de personnaliser les réponses du système, en représentant l'utilisateur et ses comportements dans la base métier.
- En ce qui concerne l'extraction des données, l'ED est alimenté à partir des sources de données divers (fichiers classiques, bases de données, documents WEB), d'oû il convient de s'assurer de la cohérence de l'ensemble de ces données et de permettre la mise à jour régulière et périodique (rafraîchissement) en accord avec les besoins des décideurs.
- Pour la structuration de l'ED, les modèles des données utilisés pour structurer des bases de données classiques ne sont généralement pas adoptés, de nouveaux modèles multidimensionnels ont été proposés pour offrir aux décideurs une représentation simple des données.
- La manipulation des données d'un ED s'effectue souvent au travers de logiciels d'analyse de données, c'est pourquoi les données doivent être sélectionnées selon certains critères ou certaines dimensions grâce à des opérateurs ad-hoc (multiparties) qui agrègent, ou au contraire les répartissent selon les axes d'étude.
- L'évaluation de l'entrepôt n'est pas uniquement liée aux extraction des données qu'il reçoit régulièrement des divers sources, son schéma peut être aussi modifié au fil du temps pour s'adapter à l'évolution du processus d'analyse.
- L'ED est généralement basé sur un serveur relationnel (peut être objet-relationnel dans un futur proche), afin de mieux intégrer les données multimédias, et pour qu'il puisse

garder des données historisées comme l'on a mentionné au moins pour une période assez longue, afin de valider les décisions à prendre.
- Les volumes de données générées sont donc assez importants, de plus les requêtes décisionnelles issues peuvent être complexes.
- La mise en œuvre d'un serveur parallèle est souvent nécessaire.

Exemple : Une entreprise de distribution utilise un ED comme suit :

- Les données de vente sont enregistrées dans des différents magasins.
- Chaque nuit les données des différents magasins sont transférées dans un ED au siège de la firme.
- Les données de l'ED sont utilisées pour mettre au point des stratégies commerciales, des compagnes de promotion, …

II.8.4. Motivation de construction d'un entrepôt de données :

1 – Les informations accumulées dans les bases de données opérationnelles exploitées pour la gestion quotidienne, sont aussi utile pour une gestion plus stratégique de l'entreprise en basant sur la notion d'ED.

2 – Le type de requêtes nécessaires nécessite souvent un traitement qui peut être :

- Soit lourd, et donc ayant un impact sur les performances de la base de données opérationnelle.
- Soit difficile à exprimer en SQL.
- Soit ces requêtes impliquent des données se trouvant dans des bases de données distinctes.

D'ou, il est nécessaire pour effectuer ce type de traitement de consolider les données dans une base de données spécifiquement conçue à cet effet c'est l'entrepôt de données (data warehouse) .

II.8.5. Architecture d'un entrepôt de données :

II.8.5.1. Aspect matériel :

L'utilisation de machines parallèles est très adéquat pour les ED, et paraît facile à mettre en œuvre.

Les architectures parallèles considérées sont des machines à plusieurs processeurs qui partagent : la mémoire, le disque ; ou rien du tout, un réseau de communication qui relie les processeurs, … etc.

Ces machines améliorent les performances de plusieurs façons suivant :

- Leurs architectures.
- La répartition des données sur tous les disques et le grain de parallélisme envisagé.

a) - Architecture à mémoire partagée :

Plusieurs processeurs travaillent sur une mémoire commune alimentée par des disques capables d'effectuer des E/S parallèles . ce type de solution est efficace et moins coûteux pour des bases de données moyennes ne permet guère de dépasser 10 processeurs du fait .

Cette architecture pose des problèmes de contention mémoire, ainsi que des problèmes de faisabilité, la mémoire est un composant central unique.

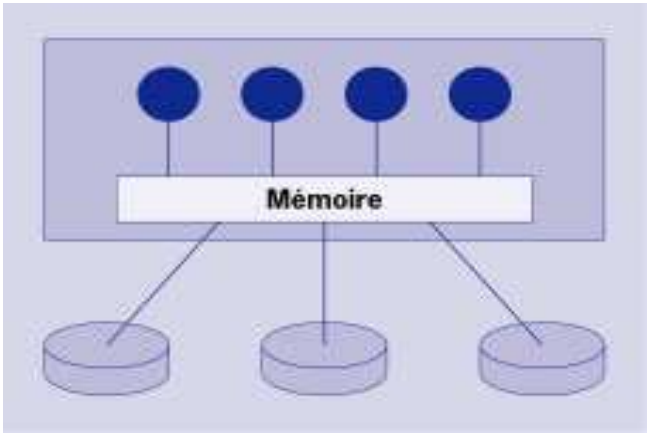


Fig II.28. Architecture à mémoire partagée

b) - Architecture sans partage (share nothing) :

Cette architecture réalise l'interconnexion de composants (processeurs et disques) avec mémoire par un réseau intelligent capable de les coordonner.

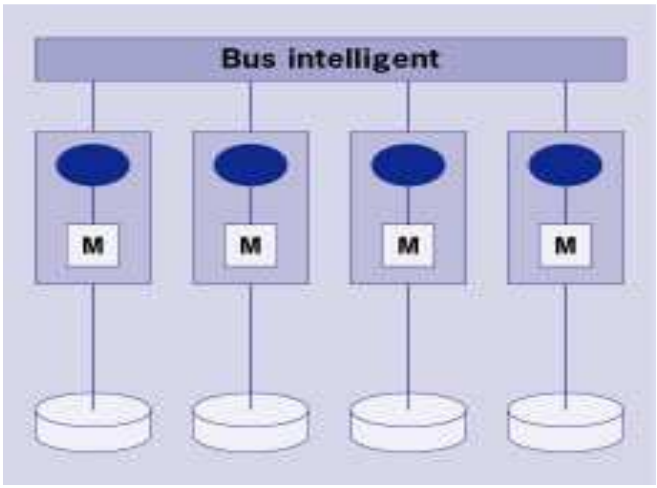


Fig II.29. Architecture sans partage

Le réseau peut être organisé sous forme d'un hyper-cube ou chaque composant peut être lui même une machine multiprocesseurs à mémoire partagée .

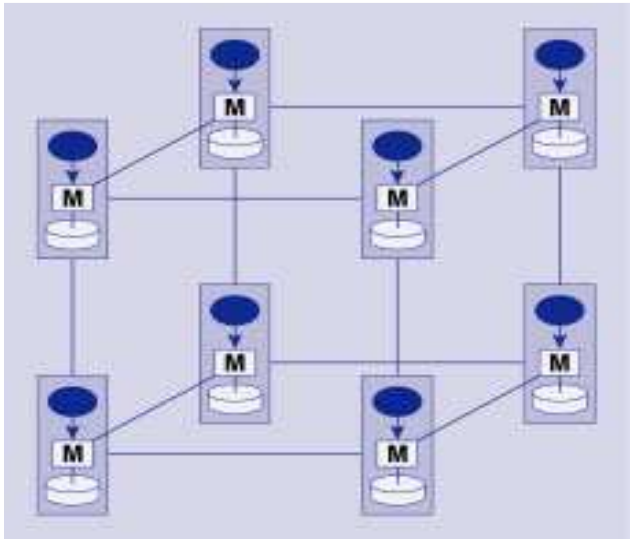


Fig II.30. Architecture de type hyper cube

Dans ce type d'architecture on peut considérer l'unité indécomposable est la requête, ou bien les opérations de base comme : la jointure, la sélection, le tri, le regroupement, … etc.

Un pipe line de taches est parfois possible, par exemple, le résultat d'une sélection faite par un processeur est directement envoyé vers un seconde processeur qui effectue un tri, et ce tri peut commencer avant que la sélection soit terminée.

Si les données nécessaires pour que deux taches s'exécutent en parallèle se trouvent sur la même unité de disque, l'exécution dans ce cas ne sera pas optimale, puisque les deux tâches devront partager une ressource commune qui est l'accès au même disque.

Inversement, une diversité des sources de données va pénaliser les tâches qui perdront du temps en communication.

Dans cette approche, la répartition des données est donc très importante. Les répartitions usuelles tiennent compte d'une table de hachage, des clés, du schéma des données ou encore se réalisent (les répartitions) à la demande.

De point de vue SGBD, la gestion de l'entrepôt par une machine parallèle nécessite le support de parallélisme inter-requête et surtout du parallélisme intra-requête.

1 - Parallélisme inter-requête (Inter-Query parallelism) :

Technique de parallélisme permettant de faire prendre en charge les requêtes par des processeurs différents .

Ce parallélisme nécessite un SGBD à serveurs multiple capable d'assigner à chaque processeur un serveur et de dispatcher les requêtes entre les différents serveurs (différents processeurs).

La synchronisation entre serveurs accédant aux mêmes données lors de l'exécution des requêtes doivent aussi être gérées, les gains en performances restent limités.

2 - Parallélisme intra-requête (Intra-Query parallelism) :

Technique de parallélisme consistant à découper une requête en unités fonctionnelles, ou chaque unité pouvant être prise en charge par un ou plusieurs processeurs différents travaillant en parallèle.

Pour réaliser totalement le parallélisme intra-requête, il faut décomposer la requête en unités fonctionnelles (sélection, jointure, tri, …). Ces unités doivent être exécutées en parallèle par un mécanisme de communication en pipeline.

Par exemple dés que deux sélections ont produit un tuple en résultat la jointure peut commencer .

Chaque unité fonctionnelle peut à son tour être parallélisée à l'aide d'un algorithme de sélection, jointure, tri parallèle, …Ces algorithmes basent sur le principe de « Diviser pour régner » . Les tables sont partitionnées par des processeurs associés aux disques afin de profiter à la fois du parallélisme au niveau des disques et de processeurs .

Les sélections sont exécutées en parallèle par les différents processeurs sur des disques différents, les jointures sont aussi effectuées en parallèle.

Ces types d'algorithmes sont connus depuis les années 80, ils sont mises en œuvre dans la version parallèle d'ORACLE, INFORMIX, DB2 .

II.8.5.2. Aspect logiciel :

Le processus de mise en oeuvre d'un entrepôt de données est généralement progressif. Souvent, les responsables informatiques commencent par :

1 - La modélisation de la structure de données cible (la mise en place d'un datamart ou base de métier relative à un département donné) : constitue l'étape initiale du *Data Warehousing*. Pour cela, le responsable du projet établit avec la direction de l'entreprise les taches suivantes : **[HAC 2003]**

- Construire un dictionnaire des formats de données, et de méta-données.
- Une fois extraites, les données sont agrégées et filtrées afin d'harmoniser leurs formats et d'éliminer les redondances.
- L'attribution de valeurs cohérentes aux variables mal ou non initialisées complète l'ensemble des étapes qui précédent l'alimentation d'un entrepôt de données.
- Une fois mis en place, l'entrepôt de données doit être mis à jour pour que le format des données corresponde aux besoins des utilisateurs.
- De plus, les changements dans les sources d'information doivent être répercutés à son niveau.
- Chaque source d'information est connectée a un wrapper/moniteur. Le wrapper est responsable de la transformation de l'information du format source au format et au modèle de données utilisés par le système du warehousing.
- Le moniteur est quant à lui chargé de détecter automatiquement les changements intéressants au niveau de la source de données et de les reporter vers l'intégrateur.
- L'intégrateur a pour rôle la mise en place de l'information dans l'entrepôt de données.

2 - Ensuite, deux types d'évolution sont possibles. Selon le choix organisationnel de la direction générale, on se dirige soit vers :

- La centralisation progressive des données stratégiques.
- Soit vers autant de datamarts que de services spécifiques.

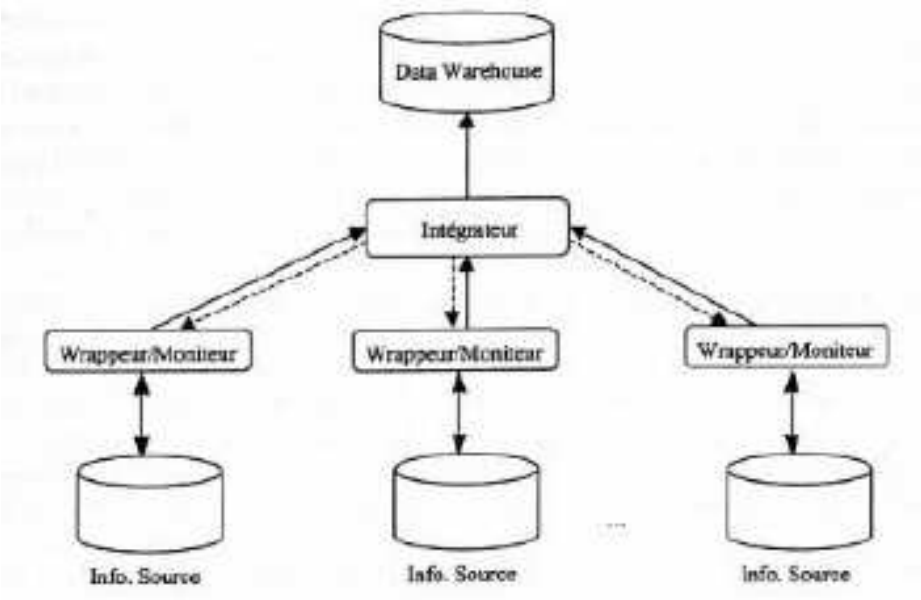


Fig II.31. Architecture de base d'un système de data warehousing – aspect logiciel

II.8.6. Caractéristiques d'un entrepôt de données :

Un entrepôt de données est caractérisé par :

- Ses données sont orientées vers l'usage décisionnel.
- Alimenté par les applications informatiques du SI opérationnels (SI de production) sur des application informatiques dites de décision (marketing, finance, produits, services, qualité, …etc).
- Ses données sont prêtes à l'emploi pour l'exploitant des données.
- Ses données sont les plus détaillées possibles, qualifiées et contrôlées clairement, définies et compréhensible, pertinentes, facilité et rapidité de modification.
- La qualité d'une donnée dans l'ED est plus importante que l'architecture du système ou la qualité du fonctionnement du SI lui même.

II.8.7. Avantages principales d'un entrepôt de données :

- Un volume très important d'informations stockées et traitées par l'ED.
- Disponibilité des données en grande quantité et à tout moment (facilité d'accès)
- Cohérence des informations disponibles.
- Possibilité de découpage des données, suivant les besoins des décideurs, ainsi que des utilisateurs.
- Un entrepôt permet aux décideurs de prendre des décisions plus pertinentes et plus rapides.
- L'ED permet l'analyse statistique des données regroupées et de leur exploitation en gestion stratégique d'une entreprise.

II.8.8. Quelques considérations pour concevoir et implanter un entrepôt de données:

Tous les avantages pré-cités impliquent des considérations pour concevoir et mettre en œuvre un ED :

- Possibilité de stocker des volumes importants de données.
- Les données doivent être soigneusement rassemblées à partir des différentes sources d'informations, bien nettoyées, filtrées, et n'être diffusées qu'après validation de leur qualité.
- L'information fournie par un ED n'est pas une données brute, mais une information qui éclaire le décideur, orientée sujet, agrégée, facile à accédér, fiable, et pertinente, non volatile et possède un contexte temporel spécifique.
- Intégration des outils d'analyse et de présentation des informations.
- Rendre possible l'accès aux informations d'analyse en temps réel (intégration en temps réel).
- Intégration d'outils de requêtes.

II.8.9. Quelques produits d'entrepôts :

Les entrepôts de données sont un lieu de prédilection pour les vendeurs de machines parallèles telles que :

- BULL (machine ESCALA).
- IBM (RS 6000) .
- ICL (GOLD RUSH).
- COMPAQ (CLUSTER).
- SUN, HP, …

Ces machines standards supportent des versions parallèles d'ORACLE, INFORMIX, DB2, SYBASE, …

Les SGBD précités ne répondent pas vraiment aux besoins de ce domaine nécessitant des systèmes rodés et bien optimisés, adaptés au matériel, capable de supporter différents niveaux de parallélisme.

Des machines spécialisées telles celles de TERRADATA de NCR peuvent offrir une solution très performante .

II.8.10. Champs d'application d'entrepôts de données :

a) - Le commerce électronique : le commerce électronique nécessite la présentation de catalogues, le contact du consommateur, et la mémorisation de ses goûts, l'exploration des produit par parcours de liens vivants, l'accès hiérarchique aux descriptions des produits et aux prix, la visualisation multimédia en 2D ou 3D (les démonstrations), … etc.

Tout cela nécessite des connexions fortes aux bases de données de l'entreprise. d'autre part, le paiement sécurisé, la saisie et le suivi de la commande requiert une base de données utilisée en temps réel (couplage d'un ED et d'un site WEB).

b) - Business intelligence : proposé par IBM, MicroSoft, Oracle, … pour présenter l'information dans des formats plus utiles en utilisant des outils de visualisation avancés et des techniques d'intelligence artificielle.

Le marché de business intelligence (BI) est vite élevé de 3.5 millions de dollars (2002) à 8.8 millions de dollars (2004)

II.8.11. Classes des entrepôts de données : **[CGI 2004]**

a) - Les mini-entrepôts de données : permettent aux utilisateurs de lancer eux-mêmes des requêtes ponctuelles impliquant des données complexes.

On parle de requêtes « ponctuelles » puisque les questions des utilisateurs peuvent varier d'une journée à l'autre et qu'ils doivent obtenir des réponses facilement, sans l'apport de programmeurs chevronnés.

La mise en place de cet environnement requiert un ensemble très intuitif de structures de données jumelées à des outils analytiques évolués destinés aux utilisateurs.

Une approche consiste à utiliser la modélisation dimensionnelle lorsqu'il est judicieux de le faire et à créer un mini-entrepôt dans le cadre de la stratégie d'entreposage des données ou à développer un nouveau mini-entrepôt à partir d'un entrepôt de données existant.

b) - Les entrepôts de données : sont idéaux pour mettre sur pied un environnement d'analyse robuste qui fusionne et intègre les données au-delà des frontières départementales et technologiques traditionnelles.

Les analystes chevronnés guident le processus d'intégration en contribuant à la conception d'un solide modèle de données logiques. À l'aide d'outils évolués, nous sommes en mesure d'extraire et de transformer les données, de les charger dans l'entrepôt de données et de documenter les règles d'approvisionnement en données.

Les experts en modélisation et les analystes de données conçoivent une architecture de données évolutive et extensible afin qu'elle puisse prendre en charge la croissance du volume de données et permette l'incorporation de nouveaux types de données lors des prochaines étapes d'entreposage.

d) - Les magasins de données : intègrent les données au-delà des frontières organisationnelles et technologiques traditionnelles pour optimiser les systèmes transactionnels plutôt que les environnements analytiques (qui relèvent plutôt des mini-entrepôts et entrepôts de données).

Une solution de magasin de données fournit des données cohérentes et consolidées aux systèmes stratégiques d'une entreprise, réduisant de façon significative les dédoublements dans la saisie de données à l'échelle de l'entreprise.

L'exemple le plus courant est le magasin de données orienté client, qui offre une vue unifiée aux applications d'affaires stratégiques d'une entreprise.

II.8.12. Difficultés rencontrées par l'approche d'entrepôts et solutions possibles :

1 . Mauvaise dimensionnement de la machine (l'architecture matérielle d'une manière plus générale).
2 . De très gros volumes de données stockées, analysées et traitées.
3 . Données mal ciblées .
4 . Sujets mal centré.
5 . Redondance non souhaitable de certaines données et Redondance insuffisante pour certaines d'autres.
6 . Période de mise à jour ou d'historisation mal choisie .
7 . la transformation des données pour l'analyse est une procédure coûteuse en temps (80 % du temps passé dans l'analyse des données est typiquement consacré à la transformation)
8 . Autres risques …

Pour cela il faut avancer dans la conception pas à pas en ajoutant une vision claire des objectifs .

Il reste aussi indispensable de développer une méthodologie de conception avec des outils d'aide, cette méthodologie devrait comporter des phases importantes telles que :

- L'étude préalable .
- Le prototypage
- La conception
- La réalisation
- Le déploiement
- La livraison
- L'évaluation

II.8.13. Les sources d'information d'un entrepôt de données :

Les sources d'informations d'un entrepôt de données peuvent être multiples (SGBD, SGF, documents HTML et XML, bases de connaissances) et sont classées en fonction de la nature de leur participation dans le procédé du Data Warehousing.

Et on distingue les types de sources de données suivants : **[HAC 2003]**

a) - Les sources coopératives : qui permettent la répercussion automatique des changements intervenant sur des données concernées par l'entrepôt de Données.

b) - Les sources 'logged' : maintiennent un 'log', différent des bases de données opérationnelles (des versions différentes de la base de données). Ce log peut être interrogé ou inspecté, pour extraire les changements intéressants sur les sources.

c) - Les sources interrogeables : qui permettent au wrapper/moniteur de demander de l'information à la source, de sorte qu'une liste de questions est utilisée périodiquement pour détecter les changements intéressants.

d) - Les sources 'photographiques' (snapshot) : fournissent périodiquement une photographie des données en tâche de fond, les changements dans ce cas sont détectés par comparaison des 'photographies' successives.

II.8.14. Outils d'accompagnement d'un entrepôt de données : [HAC 2003]

a) - Wrapper/Moniteur : a deux rôles principaux qui sont la traduction et la détection des changements.

1 - La traduction : consiste à faire apparaître les sources d'informations comme si elles sont représentées conformément au modèle de données utilisé par le système du warehousing.

Par exemple, si la source d'informations a une structure de fichiers et que le modèle adopté par l'ED est relationnel, alors le wrapper/moniteur dans ce cas doit disposer d'une interface présentant les données de la source d'information comme si elles étaient relationnelles, il est aussi appelé "traducteur" ou wrapper.

2 - La détection des changements : permet le contrôle de la source d'informations pour les changements des données concernant l'entrepôt de Données et la propagation des changements vers l'intégrateur.

Notons que les changements intervenus sur les données doivent aussi être traduits, à partir du format et du modèle de la source d'information, au format et au modèle de l'entrepôt De Données, tout comme les données elles mêmes.

Une approche consiste à ignorer la fonctionnalité de détection des changements du wrapper/moniteur (aucun signal relatif à ces changements n'est délivré) et à transmettre tout simplement des copies entières des données concernées à partir de la source d'informations vers l'entrepôt de données de façon périodique (dans le cas de tout changement).

L'intégrateur combine ces données avec celles de l'entrepôt de données provenant d'autres sources, ou il demande une information complète à partir de l'ensemble des sources et recalcule les données du data warehouse.

***b) -* L'intégrateur :** son rôle est la mise en place de l'information dans l'entrepôt de données, cela peut inclure le filtrage de cette information, son résumé, ou sa fusion avec des informations issues d'autres sources.

Parfois, des informations supplémentaires de la même source ou de sources différentes peuvent s'avérer nécessaires à l'intégrateur afin que la nouvelle information soit proprement intégrée dans l'entrepôt de données.

Une fois l'intégrateur soit chargé par son ensemble initial de données, sont travail consiste à recevoir les notifications des changements des wrappers/moniteurs et à les répercuter au niveau de l'entrepôt de données.

Dans un environnement data warehousing, il peut être nécessaire de transformer les données de base (par agrégation, simplification, correction des données erronées, élimination des doublons) avant leur intégration dans l'ED.

Le problème de traduction des données n'est pas spécifique au data warehousing, mais il se pose pour toutes les approches d'intégration des données.

Les problèmes relatifs à la détection des changement dépendent de la nature des sources d'informations.

II.8.15. Les différents types d'intégration dans un entrepôt de données :

Une première classification des différentes approches repose sur le contexte d'intégration, le type des entrées/sorties du processus d'intégration, et le but du processus lui-même. Nous distinguons :

II.8.15.1. Intégration de schémas : l'entrée du processus d'intégration est un ensemble de schémas sources, ainsi que la spécification de la façon d'associer les schémas des données sources à des parties du schéma résultant (cible). Dans la sortie on trouve un schéma de données correspondant à la représentation intensionnelle réconciliée de tous les schémas en entrée (un schéma intégré).

II.8.15.2. Intégration de données virtuelle (médiateurs) : L'entrée est un ensemble de données sources, et la sortie est une spécification décrivant la façon de fournir un accès global, unifié et transparent aux sources pour satisfaire certains besoins en informations, sans mettre compte l'autonomie des sources.

II.8.15.3. Intégration de données matérialisée : l'entrée est un ensemble de données sources, et la sortie est un ensemble de données représentant une vue réconciliée des sources au niveau intensionnel et extensionnel.

II.8.16. L'organisation des données d'un entrepôt – schéma en étoile :

On distingue habituellement deux types de données dans un ED :

II.8.16.1. Les faits : correspondent à l'activité de l'entreprise ou l'établissement ; se sont les ventes pour une entreprise commerciale, les communications pour une entreprise de télécommunication, … etc.

II.8.16.2. Les dimensions : se sont les critères sur lesquels on souhaite évaluer, quantifier, qualifier les faits.

Les dimensions usuelles sont : le temps, le client, le magasin, la région, le produit, …etc.

Exemple : on considère un ED dans lequel on accumule (regroupe) des informations concernant la consommation d'un boisson dans les cafétérias.

Les faits sont accumulés dans une relation dont le schéma pourrait être :

VENTE(cod_caf, cod_bois, cod_cons, date, heure, prix)

Les dimensions pourraient être représentées dans des relations dont les schémas seraient les suivants :

CAFETERIA(cod_caf, adresse, gérant)
BOISSON(cod_bois, goût, fabriquant)
CONSOMMATEUR(cod_cons, adresse, date_nais)

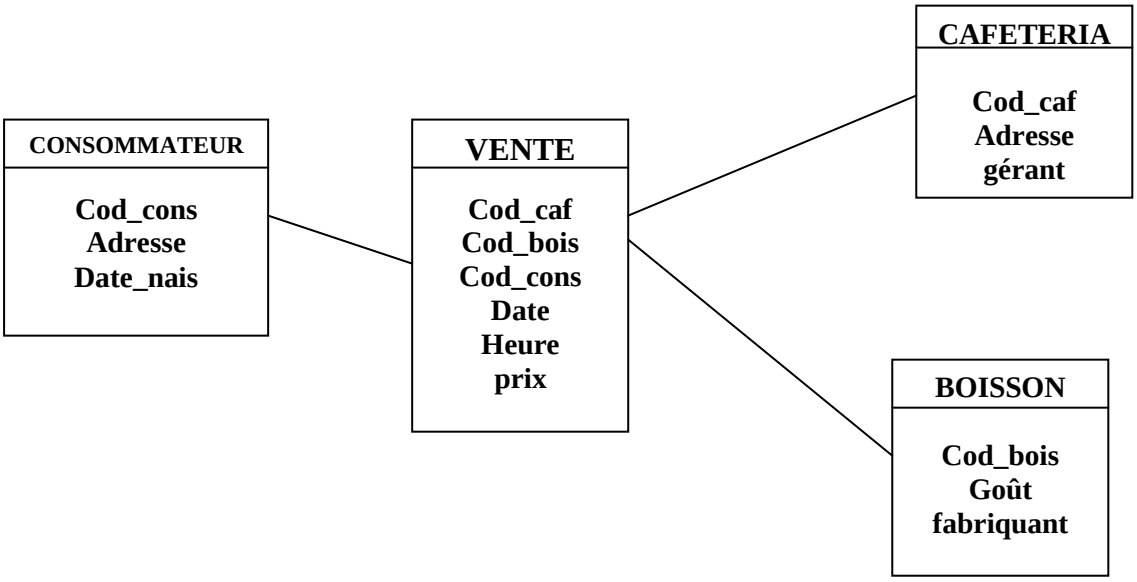


Fig II.32. Organisation d'un ED par un schéma en étoile

II.8.16.3. Attributs de dimension et attributs dépendants :

On distingue deux types d'attributs dans la table des faits :

a) - Les attributs de dimensions : qui sont les clés des relations dimensionnelles
Exemple : cod_caf, cod_bois, cod_cons)

b) - Les attributs dépendants : leurs valeurs sont déterminées par des attributs de dimensions.
Exemple : l'attribut Prix dans la relation des faits VENTE est un attribut dépendant, car sa valeur est déterminée à partir des attributs de dimensions : cod_caf, cod_bois, cod_cons plus les attributs de faits ; date, heure.

II.8.17. Extraction des informations d'un entrepôt de données :

On appelle extraction des données, le processus par lequel les données opérationnelles sont transférées vers l'entrepôt. Il faut signaler que :

- Le schéma de l'entrepôt peut être considéré comme une vue sur les données opérationnelles, on parle de vue matérialisée.
- Il est utile mais parfois difficile de pouvoir mettre à jour les données de l'entrepôt sans régénérer l'entièreté de celui-ci, on parle de mis à jour incrémentale.
- Pour extraire des informations d'un ED, il y a deux approches principales :

II.8.17.1. Technique ROLAP : consiste à exprimer les requêtes en SQL, on parle dans ce cas d'une approche (technique) ROLAP (Relational On-line Analytical Processing).

Une requête ROLAP est en général exprimer comme suit :

1 - On calcule le joint de la table des faits avec les tables de dimensions.
2 - On sélectionne des tuples en fonction des données dimensionnelles.
3 - On groupe ces données suivant certaines dimensions.
4 - On calcule une valeur agrégé (le plus souvent une somme)

Exemple : pour chaque cafétéria à M'SILA, trouver la somme des ventes de chaque boisson produit par Hammoud.

2 – On filtre le joint de la table des faits et des tables dimensionnelles par :

(fabricant = 'Hammoud') et (adresse = 'M''sila')

3 – on groupe le résultat par cafétéria et boisson.
4 – on agrège on calculant la somme des prix.

Inconvénient de la technique ROLAP :

Les requêtes ROLAP peuvent être très coûteuse en ressources.

II.8.17.2. Technique MOLAP : Dans cette technique on considére l'entrepôt comme une base de données multidimensionnelle, dont le modèle est un cube N-dim (Data Cube).

On parle dans ce cas d'une technique MOLAP (Multidimensional On-line Analytical Processing). La construction du data cube se fait comme suit :

- Les clés des tables de dimensions sont considérées comme les axes du data cube
- Les attributs dépendants apparaissent comme les points du cube.
- Le data cube peut aussi contenir des valeurs agrégées (en général la somme) suivant : les axes, les plans, hyperplans, … du cube.

Dans l'approche MOLAP, les requêtes consistent à effectuer les opérations suivantes :

1 – sélectionner et agréger les données suivant certains axes (ROLL-UP). Par exemple grouper les cafétérias par région, les consommateurs par boissons, par cafétéria, …etc
On parle aussi de : - SLICING (sélectionner suivant une valeur)
- DICING (sélectionner suivant un domaine de valeurs).

2 - Désagréger certains groupement (DRILL DOWN).
Exemple : après avoir grouper les boissons par goûts, les séparer par fabricant.

3 – Eliminer certaines dimensions (projeter), c.à.d en remplaçant les points du cube par les agrégats correspondants, on parle dans ce cas de PIVOTING.

Exemple : Soit une table VENTE d'attributs : Numéro de vente, Produit, Région, Date, Quantité, …

Cette table peut être issue d'un résultat de requête sur la base du data warehouse.

Une vue 3D représentant la mesure quantité pourra être obtenue en fonction de produit, région, date comme suit :

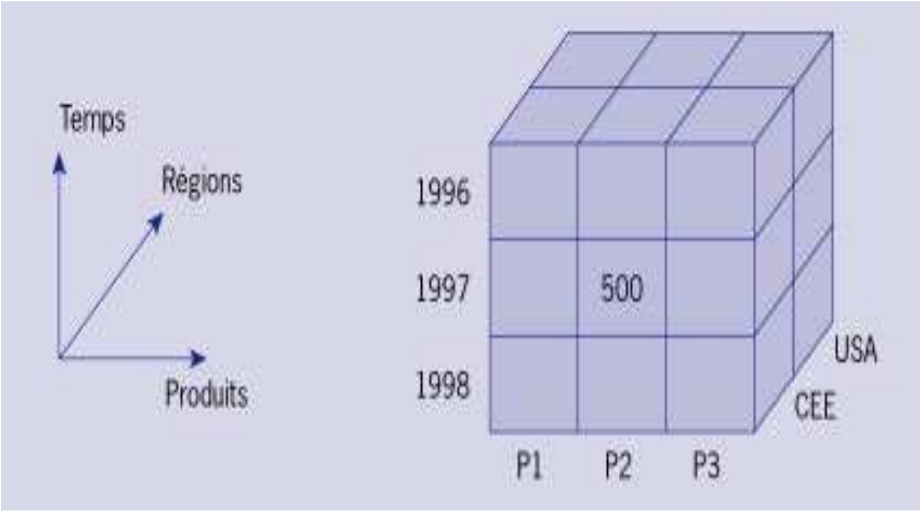


Fig II.33. Technique MOLAP - Data Cube

Il est possible d'étendre le data cube par exemple un gestionnaire sélectionnera l'année 1996 et désirera étudier l'évolution des ventes par mois, cette opération est appelée « Drill Down », l'inverse est « Drill Up ».

Le cube multidimensionnel propose une présentation synthétique des données permettant rapidement d'obtenir des courbes, des camemberts, etc, …

Il facilite aussi la sélection selon un axe, le passage à un niveau plus bas de détail, et les calculs d'agrégation (somme, moyenne, écart, min, max, …).

Le choix d'une structure de données convenable pour stocker le cube et ses agrégats reste un problème ouvert.

Il reste de nombreux produits pour l'analyse interactive des données utilisables autour des entrepôts.

Par exemple : business object, oracle express, méta cube racheté par INFORMIX, DB2 OLAP, SQL SERVER OLAP, IMPROMPTU, POWERPLAY, BRIO , … etc.

En plus les techniques ROLAP et MOLAP il existe une autre technique d'analyse multidimensionnelle appelée OLAP (On-line Analytical Processing) : est une technique informatique qui permet au décideurs d'avoir accès rapidement et de manière interactive à une information pertinente présentée sous des angles divers et multiples selon leurs besoins particuliers.

II.8.18. Construction d'un entrepôt de données :

Le data warehousing repose sur l'idée d'extraire des informations utiles et de les combiner dans un ensemble cohérent c'est le data warehouse ou l'entrepôt de données.

La cohérence de ces données est mesurée d'une façon globale, elle est jugée de point de vue du manager qui cherche à savoir si par exemple un chargement de données en cours est bien un ensemble de données complet et cohérent.

Alors comme nous l'avons mentionné, un ED est tout a fait différent des bases de données de production, car leurs besoins sont différents. Il contient des informations :

- Historisées.
- Globalement cohérentes.
- Organisées selon les métiers de l'entreprise pour l'intérêt du processus de décision.

L'entrepôt n'est pas un produit ou un logiciel mais un environnement. Ses données sont puisées dans les bases de production, nettoyées, normalisées, puis intégrées. On trouve aussi des méta-données qui décrivent les informations dans cette nouvelle base pour lever toute ambiguïté concernant leur origine et leur signification.

Il faut signaler qu'un entrepôt se conçoit avec un ensemble d'applications qui vont réaliser les fonctions pour lesquelles il a été construit : l'aide à la décision.

Ces applications sont des outils d'accès aux données (requêteurs, visualisateurs, outils de fouille, ...).

En général, on distingue trois parties interdépendantes qui relèvent de la construction d'un entrepôt de données :

1. L'étude préalable qui va répondre à la question pourquoi construire l'ED en définissant ses objectifs, et précisant la démarche de construction.
2. L'étude du modèle de données qui représente l'entrepôt conceptuellement et logiquement.
3. L'étude de l'alimentation qui reprend à un niveau plus précis l'examen des données, le choix des méthodes et des dates auxquelles les données entreront dans l'entrepôt.

II.8.18.1. Étude préalable :

Cette partie de l'étude ressemble à toute étape préliminaire à l'implantation de n'importe quel nouveau système d'information automatisé. Notons ici que les principes des méthodes connues pour le système de production restent toujours valables (méthodes comme : MERISE, UML, MOF, …).

a) - Étude des besoins :

L'étude des besoins doit déterminer :

- Le contenu de l'entrepôt et son organisation, d'après les résultats attendus par les utilisateurs.
- Les requêtes qu'ils formuleront.
- Les projets qui ont été définis.

Le besoin des informations peut provenir du système de pilotage en général ou d'un service particulier de l'entreprise. C'est souvent un projet issu d'un schéma directeur qui va déclencher l'étude et la réalisation d'un cahier des charges comme premier pas.

L'existant ici est généralement un ensemble de petits produits ou développements sans grande intégration ni relations.

A ce niveau, l'information est dupliquée, les traitements sont répétés et aucune stratégie d'ensemble n'est encore définie.

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).
- Des analyses prédictives (par exemple déterminer les achats potentiels pour un type de client, déterminer les clients qui risquent d'abandonner l'entreprise, ...).

L'étude des besoins doit aussi mettre en évidence le suivant :

- Les interviews doivent permettre de préciser les faits à suivre et dans quelles dimensions. Il faut aussi recenser les données nécessaires à un bon fonctionnement de l'entrepôt.
- Il faut alors recenser les données disponibles dans les bases de production, qui ne sont pas toutes utiles dans l'entrepôt.
- Il faut aussi identifier les données supplémentaires requises et s'assurer la possibilité de 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, et la granularité de ces faits.

Par exemple, l'unité du temps utilisée doit-elle être le jour, la semaine ?

Les produits sont-ils analysés par catégorie, par lot, par marque ?

- La variété des besoins, et leur modularité peut entraîner un découpage de l'entrepôt en plusieurs parties, parmi ces parties on trouve les datamart (les bases de métier), qui sont alimentés par un entrepôt et qui 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 datamart), soit de manière descendante.

b) - Coûts de déploiement :

Il faut utiliser une machine puissante, souvent parallèle, spécialisée pour cette tâche. Les tentatives de mixer à la fois l'informatique décisionnelle et l'informatique de production au sein d'une seule machine sont souvent des échecs. Car les utilisations sont trop différentes. La capacité de stockage doit être très importante. Les prévisions de montée en charge du système devront évaluer cette quantité.

Il faut noter que les prix de ces matériels ne cessent de décroître. La puissance de stockage , ainsi que de calcul est multipliée par deux tous les 2 ans et cette tendance s'accélère même, à prix constant.

Il faut aussi une équipe pour maintenir le système, le faire évoluer (administrateur, développeurs, concepteurs,...). sans oublier la mise en place du système, la formation, etc.

Pour les logiciels, ils peuvent être importants. Ils concernent les logiciels d'administration de l'entrepôt, ceux d'interrogation et de visualisation, ainsi que les coûts de l'environnement logiciel de data mining (fouille de données) proprement dit.

L'entrepôt devra être accessible par tout utilisateur et on se situe, généralement, dans un environnement client-serveur.

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 du système.

II.8.18.2. Modèles de données utilisés :

a) - Niveau conceptuel :

Le modèle conceptuel choisi doit être le plus simple possible pour permettre au plus grand nombre d'appréhender l'organisation des données et comprendre ce que l'entrepôt mémorise.

On parle de modèle multidimensionnel, souvent représenté sous forme de cube, car les données seront toujours des faits à analyser suivant plusieurs dimensions. Prenons l'exemple des ventes de produits à des clients dans le temps.

Les faits ici sont les ventes, les dimensions sont : les clients, les produits et le temps. Les interrogations s'interprètent géométriquement comme l'extraction d'un plan, d'une droite de ce cube (par exemple : lister les ventes du produit *A* sur période de temps *D*), ou l'agrégation de données le long d'un plan ou d'une droite (par exemple : Obtenir le total des ventes du produit *A* revient à sommer les éléments du plan indiqué en figure).

Un avantage évident de ce modèle est sa simplicité, même si dans un entrepôt important il peut exister plusieurs cubes, parce qu'il est nécessaire de suivre plusieurs faits dans des directions parfois identiques, parfois différentes.

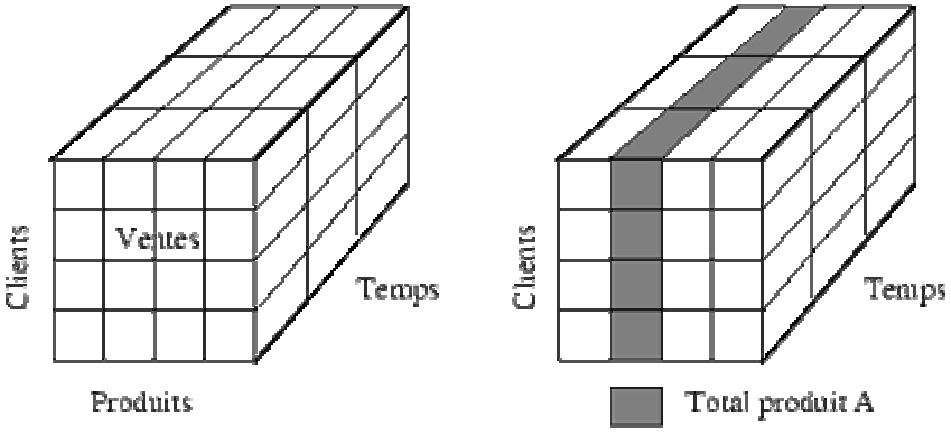


Fig II.34. Modèle en cube

b) - Niveau logique :

Au niveau logique, l'unité de base de représentation est la table comme dans le modèle relationnel. L'implantation classique consiste à considérer *un modèle en étoile* avec au centre on trouve la table des faits et les dimensions comme autant des branches à l'étoile. Les branches de l'étoile sont des relations de 1 à plusieurs, la table des faits est énorme contrairement aux tables des dimensions. Donc le modèle est très dissymétrique en comparaison avec les modèles relationnels des bases de production.

Encore, la simplicité du modèle obtenu rend la construction en étoile très pratique.

Les faits sont qualifiés par des champs qui sont le plus souvent numériques et cumulatifs comme des prix, des quantités et des clés pour relier les faits à chaque dimension.

Les tables des dimensions sont caractérisées par des champs le plus souvent textuels.

Dans l'exemple suivant, nous nous limitons au suivi d'un seul fait : le montant des ventes.

Un enregistrement dans la table des faits VENTE correspond à un total des ventes à un client dans une tranche horaire d'un jour précis, pour un produit choisi.

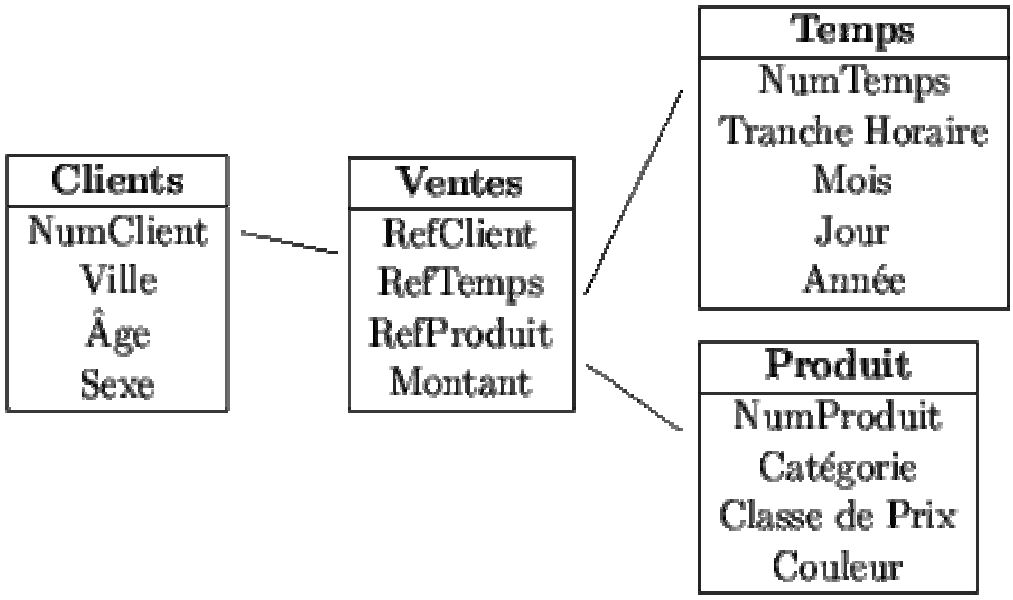


Figure II.35. Modèle en étoile

Il existe un autre modèle concurrent, c'est le modèle en flacon. L'avantage majeur de ce modèle est l'économie de place de stockage. Mais, un examen approfondi montre que ce modèle n'apporte rien, et complique même le modèle. Son principe est le suivant : Quand une hiérarchie apparaît dans une dimension, il est préférable d'enregistrer toute cette hiérarchie dans une seule et même table formant une grande dimension.

Exemple : pour une dimension produit avec des catégories produit et des sous-catégories (et ainsi de suite), toutes les échelles sont conservées dans la même table de dimension. Il sera alors possible de *naviguer* (ou *forer*) par des opérations de zoom dans cette échelle (D*rill up* et D*rill down*).

II.8.18.3. Alimentation :

L'alimentation de l'entrepôt de données est la procédure qui permet de transférer des données du système opérationnel vers l'entrepôt de données en les adaptant.

La conception de cette opération est une tâche 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.

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.

Généralement, ce sont des faits qui seront ajoutés. Les dimensions évoluent selon leur nature soit peu soit assez rapidement.

Dans notre exemple de vente, nous pouvons imaginer que la gamme des produits n'est pas très souvent modifiée. Par contre, de nouveaux clients seront certainement ajoutés mais bien moins que des dates.

a) - Transformations et vérifications :

l'opération d'alimentation de l'entrepôt rencontre le problème suivant :

La forme et le contenu des données de production ne convient pas toujours immédiatement au format choisi pour les données de l'entrepôt. Par conséquent, des transformations sont souvent nécessaires.

1 – Le format :

Le format physique des données provenant de la production peut ne pas être adéquat avec le système hôte de l'entrepôt. D'ou des transformations de type sont parfois nécessaires (Système IBM vers système Unix...).

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

2 - Consolidation :

Selon les choix des unités pour les différentes dimensions, des opérations de consolidation devront accompagner le chargement des données (par exemple sommer les ventes pour obtenir et enregistrer un total par jour et non pas pour toutes les transactions).

3 - Uniformisation d'échelle :

Pour éviter de trop grandes dispersions dans les valeurs numériques, une homogénéisation des échelles de valeurs est utile.

Si on ne réalise pas cette homogénéisation, cela peut pénaliser les outils d'analyse et de visualisation et peut aussi remplir inutilement les disques.

4 – Autres transformations :

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

Par exemple, il est fréquent qu'un même client soit mémorisé plusieurs fois sur différents serveurs.

Parmi les points à vérifier, on peut citer:

1 – Les erreurs de saisie :

Par exemple des doublons sont présents mais qui sont invisibles ; à cause des fautes de frappe : (Mahmoudi Ali ; 3, rue Didouche Mourad ; Alger), (Mahmoud Ali ; 3,rue Didouche Mourad; Alger) sont certainement un seul et même client.

2 – Erreur d'intégrité des domaines :

Un contrôle sur les domaines des valeurs permet de retrouver des valeurs douteuses .

Par exemple on a un doute pour la date 11 novembre 1911 (11/11/11) ou la date 1 janvier 1901 (01/01/01).

3 - Informations manquantes :

Des champs importants pour lesquels aucune valeur n'a été saisie peuvent pénaliser le processus de découverte d'information, ou bien pour avoir une signification particulière telle que la détection de fraudes.

Pour cela, il est parfois important d'insérer des valeurs par défaut significatives (comme NULL), plutôt que de laisser ces données vides.

Il convient aussi de noter le suivant :

- Les sources des données alimentant un entrepôt peuvent être hétérogènes.
- Les bases de production peuvent être nombreuses, différentes et délocalisées géographiquement.
- Des fichiers peuvent être achetées auprès d'entreprises spécialisées dans la constitution et la revente de fichiers qui vont aussi entrer dans le processus d'alimentation de l'entrepôt.
- Pour prendre en compte ces problème de vérification et de normalisation les suites logicielles d'accompagnement d'entrepôts de données contiennent des outils susceptibles d'aider à développer des procédures d'alimentation.

b) - Méta-données :

Il est évident que des choix conceptuels, logiques et organisationnels ont été opérés tout au long de construction de l'entrepôt de données.

La perte de l'information sur ces choix peut induire de mauvaises interprétations. Donc, un annuaire spécialisé conserve toutes les informations (les méta-données en clair) au sujet du système d'information qui régit l'entrepôt.

Cet annuaire contient entre autres choses :

- Les adresses et les descriptions des objets de l'entrepôt (tables, champs, ...) En particulier chaque donnée est définie par une description précise et claire, qui montre son origine, ... , surtout lorsque les données proviennent de bases différentes, et qu'elles peuvent avoir d'autres significations par ailleurs ;
- Les utilisateurs, ainsi que les autorisations, ...;
- Les règles de transformation et de vérification ;
- L'historique de l'entrepôt avec les dates de chargement, ...

Alors, Sans référentiel qui qualifie de façon précise ce que signifie chaque valeur dans la base, il n'est pas possible de conduire une analyse et interpréter les résultats. C'est exactement le rôle que joue l'annuaire des méta-données.

c) - Certification, publication :

Comme nous l'avons montré précédemment, la cohérence de l'entrepôt est globale et ce n'est qu'à la fin de la procédure de chargement qu'il est possible de la vérifier.

Une procédure de certification de la qualité des données chargées doit suivre le chargement. Cette procédure est spécifique à l'entrepôt et ne peut être décrite précisément. 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 longue) transaction.

II.8.19. Utilisation et exploitation de l'entrepôt de données :

L'utilisateur final doit pouvoir interroger les données *en ligne* à l'aide d'outils simples et conviviaux. Ces outils commencent à se généraliser.

Les éditeurs les nomment : reporting tools, managed queries, Executive Information Systems (EIS), OLAP tools (Online analytical Processing), ... etc

Bien que les différences entre tous ces systèmes ne soient pas toujours très nettes.
On suppose avoir défini un entrepôt de données pour lequel les données sont organisées selon un modèle en étoile. Nous allons préciser les outils attendus dans un tel environnement :

- Outils de requête (SQL) et d'analyse multi-dimensionnelle.
- Les opérations possibles pour l'utilisateur en analyse multi-dimensionnelle.
- Les outils de visualisation.

II.8.19.1. Requêtes :

Nous présentons ici les outils destinés à l'utilisateur final qui lui permettent d'extraire des données de l'entrepôt. Parmi ces outils on cite :

**** Les outils de création de rapport (reporting tools) :*** extraient les données et proposent une mise en forme destinée à leur diffusion : par impression ou par des services internet ou intranet. Lorsque leur intégration dans le système d'information est réussie, ils mettent en évidence :

- La structure multidimensionnelle.
- Présentent les agrégats.
- Supportent la navigation.

**** Des progiciels comme SAS :*** dans ce domaine ont a aussi réalisé une percée importante. Ils sont qualifiés de EIS tools (Executive Information System Tools) et ajoutent des analyses classiques et paramétrables pour les ventes, les achats ou la finance.

Les outils les plus adaptés sont certainement les outils OLAP, malheureusement, on pourra relever quelques défauts qui expliquent peut-être la difficulté qu'ils ont à s'imposer.

À l'origine, E.F. Codd (précurseur dans les bases de données relationnelles) a aussi écrit 12 règles pour définir ce que signifie OLAP. Ces règles portent sur l'évidence des différentes dimensions telles que, la transparence du système, l'accessibilité aux utilisateurs, la performance, l'intégration dans le client-serveur, l'extensibilité des dimensions, la navigation, la flexibilité,... etc.

Les éditeurs ont tenté de bâtir des outils se réclamant de ces règles. Malheureusement, il est difficile de vérifier les principes OLAP seulement avec le langage SQL.

De ce fait, la plupart des outils sont maintenant des systèmes propriétaires non ouverts et non standardisés. De plus, ces systèmes ne peuvent pas à l'heure actuelle traiter des bases de données volumineuses.

II.8.19.2. Agrégats et navigation :

L'opération de *navigation* (appelée aussi forage) permet d'obtenir des détails sur la signification d'un résultat en affinant soit une dimension ou en ajoutant une dimension.

Cette opération figure dans de nombreux outils et doit être intégrée dans le système (elle est souvent coûteuse).

Par exemple si un utilisateur final demande les chiffres d'affaires par produit, et s'étonne d'un résultat pour un produit donné. Il aura sûrement l'envie d'en analyser les raisons.

Une solution consisterait à ajouter la dimension temps, dans l'unité du temps trimestrielle pour trouver une variation saisonnière, ou dans l'unité hebdomadaire pour envisager l'effet week-end, ou encore la dimension magasin pour mettre en évidence un effet géographique.

Pour des raisons de performance, il est utile de pré-calculer et pré-enregistrer dans l'entrepôt des agrégations de données.

Dans notre exemple, si des requêtes fréquemment formulées impliquent le calcul de la somme des ventes par produit et par trimestre, il est dans ce cas simple d'optimiser le temps de réponse du système en mémorisant ce calcul.

On note que parfois que l'étude des besoins qui doit mettre en évidence la création des agrégats. Et pour insister sur l'importance de mémoriser les résultats de calculs, on peut noter que des machines sont parfois dédiées à leur exécution et leur diffusion : ce sont des *serveurs d'agrégats*.

La stratégie de mémorisation de ces agrégats peut prendre deux formes :

1 – Une première solution consiste à créer de nouvelles tables de faits pour les agrégats, de manière que toutes les dimensions inutiles ou incohérentes avec l'agrégat seront supprimées.

2 - La deuxième solution consiste à enrichir la table initiale des faits par les agrégats avec une information supplémentaire indiquant le niveau de regroupement.

Ces deux solutions ont quasiment la même incidence sur les volumes.

Pour expliquer un résultat donné, il est parfois nécessaire de le comparer avec d'autres faits.

Par exemple, la baisse des ventes pour le mois de janvier peut s'expliquer par une baisse des achats ou encore une rupture au niveau du stock.

Si l'entrepôt est conçu pour suivre les trois faits ensembles (ventes, achats, stock), et si les dimensions selon lesquelles ces trois faits sont suivis sont les mêmes, alors on doit

pouvoir réaliser un rapport unique et on parle dans ce cas de *forage transversal* (D*rill across)*.

C'est une opération qu'il faut réaliser avec beaucoup de soins, car mettre en oeuvre une requête sur plusieurs tables de faits peut se révéler irréalisable. D'ou, engagée sans précautions, la requête va générer une table intermédiaire énorme qui sera le produit cartésien entre les trois tables de faits.

II.8.19. 3. Visualisation :

Les outils de visualisation sont très importants dans le processus de décision et peuvent intervenir à plusieurs niveaux. Ils sont utiles pour :

1. Découvrir de nouvelles informations, parce qu'une représentation permet de repérer plus simplement des singularités, des anomalies ;
2. Présenter des résultats, dans l'optique d'une large diffusion, car un graphique est plus lisible et accessible qu'un tableau de chiffres ;
3. Représenter un modèle issu d'une opération de fouille de données (représenter un arbre de décision, un ensemble des règles, un réseau de neurones...).

Pour assurer le premier point, les outils de visualisation sont généralement intégrés dans les outils d'analyse et doivent supporter des opérations comme : comparer, modifier les échelles, retrouver les données correspondant à un point ou un objet tracé, zoomer sur des régions, permettre la navigation (Drill-up, Drill down).

La représentation d'un objet reposant sur plusieurs dimensions sur un écran ou une feuille de papier pose des problèmes énormes.

Les représentations 3D ne sont pas toujours adaptées, les objets sont déformés, certains sont cachés... Des animations sont nécessaires pour faire pivoter la représentation. Les couleurs, utilisées en grand nombre, sont difficiles à choisir, facile à confondre.

La visualisation scientifique reste un secteur de la recherche actif où vont se mêler des arguments informatiques, psychologiques, ergonomiques, esthétiques, ...etc.

Le choix du type de graphique utilisé même en représentation 2D est important surtout pour une large diffusion.

- Un graphique en camembert représente une fonction de répartition (représenter une partie par rapport le tout).
- Les courbes sont utiles dans le cas des échelles numériques continues.
- Les histogrammes sont préférés quand une dimension a des valeurs discrètes (une liste de régions, une liste de produits, …).

II.8.19. 4. Supports physiques et optimisations :

On essaye de discuter les éléments à mettre en considération lors de l'implantation d'un entrepôt de données.

a) - Calcul des volumes :

C'est une évaluation assez nécessaire et traditionnelle à réaliser au moment de la rédaction du cahier des charges lorsqu'on désire construire une base de données ou un entrepôt.

Dès que les études logiques sont terminées, il est nécessaire d'estimer la place que les données occuperont sur disque.

Par exemple le suivi des ventes d'une chaîne de 300 magasins dont les dimensions sont : le temps, magasin, produit, promotion. Et les faits suivis sont : le prix de vente, la quantité vendue, le prix d'achat, le nombre d'achats réalisé.

Un enregistrement (RECORD) de la table des faits aura une structure logique de la forme :

Fait(RefTps , RefProd , Refmag , Refprom , PrixVente , QtéVendue , PrixAchat , QtéAchat)

On estime conserver les enregistrements sur 02 ans jour par jour, c.à.d 730 jours.

Et les 300 magasins vendent en moyenne 3000 produits par jour parmi 30000 possibles, ce qui donne un total de 300 x 3000.

Les produits ne bénéficient que d'une seule promotion dans un magasin à un jour donné. On aura donc comme résultat final : 730× 300 × 3000 × 1 = 657 millions d'enregistrements dans la table des faits.

Suivant le système choisi, et la représentation physique des clés (04 clés) et des entiers (04 entiers), on peut estimer la taille de la table des faits à environ 21 Go.

On note ici qu'il reste à estimer la taille des index, la taille des tables de dimension,… etc.

b) - **Matrices creuses:**

Dans l'exemple précédent, on voit dans la table des faits que seulement 3000 produits parmi 30000 produits sont vendus chaque jour dans un magasin, c.à.d la table des faits ne contient pas un enregistrement pour chaque élément dans chaque dimension.

On interprète cette constatation en fonction du cube en disant que beaucoup de cubes unitaires ne contiennent aucune donnée, formant ainsi une matrice creuse (pleine de zéros).

Cette propriété peut-être exploitée pour améliorer la performance des algorithmes et optimiser le stockage.

c) - Le problème des clés et des index :

La gestion des clefs et des index est un grand problème pour ces grandes tables. La table des faits aura des index pour chaque clé étrangère, ainsi que pour les regroupements les plus fréquents de clés étrangères.

Dans notre exemple, la clé étrangère RefProd est très utile pour retrouver rapidement les achats d'un produit donné.

Un index sur la clé composée (RefProduit.RefTemps) est utile pour retrouver les achats d'un produit donné dans un mois donné.

Les tables de dimension sont souvent indexées suivant tous leurs champs. Il faut souligner que les gains les plus remarquables s'obtiennent par deux techniques :

- Utiliser des *machines parallèles.*
- *Pré-calculer des agrégats*.

d) - Les agrégats :

Le pré-calcul de certains groupes de données les plus fréquemment demandés est l'optimisation la plus performante. Les gains vont d'un rapport de 10 à 1000.

Dans certains cas, des *serveurs d'agrégats*, des machines dédiées à distribuer des calculs intermédiaires sur les groupes, sont intégrées dans l'architecture de l'entrepôt.

II.8.20. Entrepôt et fouille de données (Data Mining) :

II.8.20.1. Définition du data mining : il s'agit de trouver des informations dans un entrepôt allant au delà de ce que l'on peut aisément et efficacement exprimer avec une requête.

Autrement dit, une fouille de données ou data mining est un processus d'acquisition progressif de connaissances qui regroupe un ensemble de techniques d'exploitation intelligentes des données, afin de répondre à des questions complexes enfonies (cachées) dans les entrepôts de données (data warehouse) comme par exemple la fidélité des clients, ou l'évaluation de la qualité du service …etc.

Ce processus peut être soit interactif soit partiellement ou totalement automatisé.

Si le processus est totalement automatisé on parlera du système expert, si le système est interactif est partiellement automatisé on parlera du processus assisté par ordinateur (I.A.O = Information assistée par ordinateur).

Le data mining opère beaucoup plus sur les données détaillées que sur les données agrégées.

II.8.20.2. Méthodes et outils de data mining :

Ensemble de techniques d'exploration de données permettant d'envisager les liens sémantiques entre ces données par exemple par champs d'application tels que :
L'analyse de risque, le marketing direct, la grande distribution, la gestion du stocks, la maintenance, le contrôle de qualité, la médecine, l'analyse financière, …

Ces méthodes s'appuient sur la découverte des règles à partir des données afin d'améliorer grandement les processus, …

Les mécanismes de base sont les méthodes de déduction issues de la logique et permettant de déduire un théorème à partir d'axiomes. Donc les méthodes nécessitent la connaissance des règles à priori.

D'une autre part les méthodes d'induction permettent de tirer des conclusions à partir d'une série de faits.

Parmi les techniques employées dans le data mining on peut citer :

1– l'analyse statistique : qui consiste à choisir des variables et les analyser par rapport à des fonctions connues en calculant les variances, les écarts types, les corrélations entre les variables.
2 - La découverte des règles : permet la découverte des relations fines entre données.
3 - La recherche de modèle fonctionnel : consiste à induire un ensemble de valeurs en sortie à partir d'un ensemble de valeurs en entrée.

II.8.20.3. Quelques produits de Data Mining :

Afin d'appliquer ces techniques et méthodes, il existe une dizaine de produits de data mining sur le marché, parmi ces produits :

1 – Intelligent miner d'IBM : Une famille d'outils actuellement composée de :

- **Intelligent miner for data (IMD) :** permet la préparation des sources de données relationnelles et qui comporte : la sélection des données à explorer, le codage, la détermination des valeurs manquantes et l'agrégation des valeurs.
- **Intelligent miner for text (IMT) :** orienté vers l'analyse de textes libres afin de faciliter leur compréhension, il comporte à son tour trois composants : un moteur de recherche textuel avancé (TEXT MINER), un outil d'accès au web (moteur de recherche NETQUESTION, et un méta-moteur), un outil d'analyse de textes (TEXT ANALYSIS).

2 – SPSS : est le leader du marché de data mining offrant une large gamme de produits incluant l'outil principal SPSS historiquement réputé pour ses statistiques plus le système CLEMENTINE de la société anglaise intégral solution (ISL) .

ces solutions permettent l'accès et la gestion des données, la visualisation des données, la présentation des rapports multidimensionnels, les statistiques traditionnelles, la construction d'arbres de décision, l'utilisation de réseaux de neurones, la distribution de rapports électroniques statistiques ou graphiques, … etc .

3 – MARKETONE de right point software : le descendant de DATA MIND, un produit de data mining construit en France puis expatrié aux Etats-Unis utilise une technologie originale basée sur les réseaux d'agents, il montre l'intérêt du data mining pour les compagnes de marketing.

MARKETONE se compose d'une série de composants logiciels pour aider à l'implémentation de systèmes de marketing en temps réel.

4 – DARWIN de thinking machines : Intègre trois techniques de base :

- réseaux de neurones
- arbre de décision
- raisonnement basé mémoire.

L'outil fonctionne en client-serveur et propose des modules de transformation de données et de visualisation de résultats.

5 – SCENARIO DE COGNOS : basé sur la segmentation par arbres de décision étendus avec des techniques statistiques telles que l'analyse factorielle, la détection des cas singuliers et une présentation graphique intuitive.

Il permet aussi une intégration avec les techniques multidimensionnelles et les outils décisionnels associés tels que : IMPROMPTU, POWERPLAY.

6 – MINESET de silicon graphics : Un outil réputé pour ses fonctionnalités interactives et graphiques, il intègre les techniques de classification et de segmentation, ainsi que la génération de règles associatives. Il offre une interface graphique élaborée et de nombreux utilitaires, notamment pour préparer les données.

7 – NEUROSHELL de ward systems : basé sur les réseaux de neurones, il offre de nombreuses techniques avancées d'apprentissage et d'exécution en utilisant des algorithmes génétiques pour l'optimisation, il comporte les modules suivants :

- NEUROSHELL PREDICTOR : module de prédiction.
- NEUROSHELL CLASSIFIER : module de classification .
- Une version spécialisée pour la prédiction financière (NEUROSHELL TRADER)

II.8.21. Entrepôt de données et WEB :

une nouvelle technologie permettant le couplage entre les entrepôts de données et le web.
Un data webhouse est un data warehouse dont une partie des données est capturée par le web. Ce couplage est nécessaire pour au moins certaines applications telles que le client-serveur web ou la connexion des applications autour du data warehouse à l'intranet et l'extranet permet d'ouvrir les SI au personnel de l'entreprise et aux clients externes.

II.8.22. Impact du WEB sur les entrepôts de données :

1 – Accélérer le chargement, ainsi que le changement des données, car le WEB permet D'intégrer et d'analyser les données en temps réel.
2 – le WEB offre des meilleurs moteurs de recherche, que ce n'est pas le cas avec un entrepôt de données seul .
3 – plus de composants électroniques connectés au WEB.
4 – plus de données acquises (car des nano-ordinateurs dans les voitures, les maisons, les bureaux, …etc, sont connectés au réseau).
5 – Disponibilité du multimédia (audio – vidéo – média cartographique - …etc).
6 – l'interaction avec les utilisateurs se fait d'une façon électronique.
7– L'acquisition, le nettoyage, la transformation des données sont en temps réel.

II.8.23. Entrepôt de données temps réel :

Un entrepôt de données temps réel ce n'est qu'un entrepôt connecté par un WEB, il se compose de :

1 – Base de métier (Data Marts).
2 – l'entrepôt de données lui même.
3 – Capteurs.
4 – Distributeurs.
5 – Transformateurs.
6 – Systèmes opérationnels.
7 – Agrégateur incrémental.

II.8.24. Gestion d'un entrepôt de données issu du WEB :

Beaucoup de sites web sont composés par des fichiers HTML liés. Il est souvent difficile de les uniformiser, de garder une certaine cohérence entre eux, d'exploiter les informations capturées et de maintenir à jour.

Les entrepôts de données permettent la gestion de méta-données relatives aux informations capturées, la saisie interactive des informations, et le maintien de l'intégrité de la base. Ils apportent donc un support indispensable pour la gestion de données évolutives et la présentation homogène de données issues du web.

Utilisés en sens inverse, ils permettent aussi de publier des données homogénéisées et issues de sources multiples sur un intranet d'entreprise ou sur internet.

II.8.25. Perspectives d'un entrepôt de données (Recherches actuelles) :

Les recherches sur les ED menées actuellement portent essentiellement sur deux aspects :

1 – la maintenance des entrepôts de données.
2 – La découverte des connaissances à partir des données stockées.

- La maintenance de l'ED entraîne donc la consultation des sources sous-jacents et cette consultation peut induire des délais importants (pannes du réseau, indisponibilité des sources, délais de traitement des données, … etc).

Afin d'éviter tous ces inconvénients, et rendre l'ED moins dépendant des sources sous-jacentes, un certains nombre de données supplémentaires sont ajoutées et maintenues au niveau de l'ED lui même.

Dans ce contexte, les recherches portent sur l'étude des méthodes et des algorithmes pour la conception des entrepôts indépendants.

- Concernant la découverte des connaissances à partir des données accumulées de l'entreprise au fil du temps, les travaux de recherches portent sur l'extraction des règles dites d'association. Et comme l'extraction de telle règle est très coûteuse en temps et en espace, les travaux se concentrent sur l'accélération du processus d'extraction par ré-utilsation des résultats.

Concernant l'intégration des données, les prochaines grandes étapes à fixer sont :

1 – support des fonctions de planification collaborative.
2 – Unifier certains services marketing, finance, et opération.

Un autre facteur d'influence est le développement des services web. Ce qui permette la création des entreprises électroniques configurables dynamiquement.

D'une autre part, les concepts et les outils des entrepôts de données doivent évoluer pour inclure des mécanismes d'accès à des bases de données de ces services web.

Les informations obtenues devront pouvoir être intégrées et stockées dans des entrepôts de données fédérées, on peut aussi imaginer des agents intelligents interagissant avec des fournisseurs des services WEB, pour obtenir des informations pertinentes (disponibles tout le temps) pour des ED.

En conclusion l'entrepôt de données deviendra alors petit à petit un entrepôt de connaissances comportant des données issues des entrepôt traditionnelles, mais aussi des connaissances du domaine, des ontologies, et des méta-données.

II.8.26. Conclusion :

Les entrepôts de données sont des structures qui accueillent un flux important de données orientées sujets, organisées, coordonnées, intégrées, variables dans le temps et non volatiles. Ces données sont orientées beaucoup plus au processus d'aide à la décision.

Les ED adoptent une architecture spécifique basée sur :

- Des machines puissantes souvent parallèles.
- Des middlewares de collecte de données.
- Des outils d'analyse et de gestion de données.
- Des SGBD
- Des serveurs transactionnels.

Cette architecture permet l'extraction des connaissances et autorise des recherches complexes, rapides, et performantes.

La construction d'un tel ED utilise une approche multidimensionnelle favorisant l'analyse multiple des données afin de garantir la lisibilité des données, ainsi que la simplicité d'y accéder.

Lors de la construction d'un ED on est amené obligatoirement à une redondance des données, il sera aussi nécessaire d'agréger, de compter, de sommer, et de réaliser quelques statistiques élémentaires (moyenne, écart type,…).

Il faut noter aussi que dans un ED il est toujours possible de découper les données suivant les besoins des décideurs et les profils des utilisateurs afin de permettre des décisions pertinentes et rapides, ainsi que des recherches de connaissances flexibles.

L'approche d'entrepôt de données est devenu aujourd'hui une technologie très répondue pour la gestion stratégique de la plupart des entreprises économiques et financières.

II.9. Conclusion du chapitre (Critiques des approches) :

Au long de ce chapitre nous avons exposé quelques approches d'interopérabilité entre bases de données hétérogènes et réparties, ces approches sont :

1 – Interopérabilité par transfert de fichiers.
2 – Interopérabilité en temps réel.
3 – Interopérabilité en temps différé.
4 – Approche d'interopérabilité par médiation (de schémas).
5 - Approche d'interopérabilité par médiation de contexte (Approche ontologique).
6 - Approche d'interopérabilité par bases de données fédérées (Federated Data Bases).
7 - Approche d'interopérabilité par entrepôt de données (Data Warehouse).

Certes que ces approches ne sont pas les seules approches existantes dans le domaine d'interopérabilité des bases de données, il y en a d'autres, mais ces approches sont les plus connues, et des centaines de recherches ont été élaborées dans le but de les expliquer, les améliorer et les rendre plus répandues.

Malheureusement, et malgré que chaque approche possède ses partisans et ses domaines d'utilisation, chacune d'elle souffre de quelques faiblesses.

Dans cette conclusion, nous allons présenter les points de faiblesses les plus visibles pour chacune d'elles.

1 – Approche d'interopérabilité par transfert de fichiers :

- Le support de transfert des fichiers utilisé est : soit le protocole de transfert des fichiers FTP, soit un message. Le protocole FTP malgré qu'il peut transférer des fichiers entre n'importe quel client et serveur FTP, mais son ensemble limité de services limite son utilisation à un seul type de service de transfert, c'est toujours le transfert de fichiers, cela veut dire que les application ne peuvent pas échanger autre chose au dela des fichiers, par exemple des données, des objets, ou mêmes des programmes.
- L'interopérabilité par le protocole FTP est binaire. C.à.d chaque application coopérant voit les données dans son format natif, et c'est le protocole qui assure les conversions nécessaires ce qui complique le transfert et le rend coûteux en temps.

2 – Approches d'interopérabilité en temps réel et différé :

- L'interopérabilité en temps réel se réalise par l'intermédiaire d'un programme spécifique appelé programme appelant, qui élabore basant sur les indications de l'utilisateur une requête dont les composants peuvent être distribués entre plusieurs systèmes coopérant, d'ou l'efficacité de cette interopérabilité dépend directement des performances du programme appelant qui peut baser sur des indications mal ciblées.
- Le mécanisme de dialogue avec les systèmes applicatifs utilise des fonctions de synchronisation inter applicatives qui basent sur des fonctions de transfert fournies par les protocoles de communication ce qui pose le problème de limitation des services de transfert autorisés.
- Le mécanisme d'interopérabilité base sur une solution propriétaire (des protocoles spécifiques) ce qui pose un problème de standardisation.
- La requête élaborée par le programme appelant qui représente le moteur de transfert et d'échange entre systèmes applicatifs utilise des composants qui peuvent résider dans plusieurs systèmes ce qui pose le problème de complexité de la requête et l'optimalité de son exécution.

3 – Approches de médiation (de schémas) :

- L'ajout d'une nouvelle source de données entraîne la modification du schéma global adopté par la médiation.
- Le non déterminisme de l'ontologie adoptée par le médiateur pour générer le schéma global d'accès aux sources.
- Les requêtes formulées doivent être réécrites en fonction du langage commun adopté par le schéma intégré (langage médiateur) puis en langages propriétaires des sources concernées puis en langage commun avant qu'elles auront des réponses (03 réécritures). Toutes ces réécritures vont consommer un temps important.

4 – Approche par médiation de contexte (approche ontologique) :

- C'est une approche orientée beaucoup plus vers la sémantique, d'ou elle pourra résoudre surtout les conflits sémantiques, tandis que pour les autres types de conflits (conflits de modélisation, de classification, structurels, de schémas, …) elle prouve sa limitation.
- L'approche se réalise en plusieurs phases et implique trop de concepts, de notion, d'opérations et surtout de transformations.
- Le non déterminisme de la notion de l'ontologie qui joue un rôle fondamentale dans la littérature de l'approche. Jusqu'à maintenant les ontologies constituent une discipline qui reste à classifier et à développer en comprenant ses principes et ses utilisations.
- L'approche s'appuie sur certaines contraintes telles que :
  - l'interopération entre un client X et des sources S_i se réalise lorsque X et S_i adopte un modèle de représentation commun.
  - Compréhension mutuelle des éléments partageables entre X et S_i .
  - Compréhension de la signification des données reçues par X et des requêtes reçues par S_i .

Si l'une des ces hypothèses n'est pas rempli, cela peut perturber l'efficacité et la validité de toute l'approche.

5 – Approche d'interopérabilité par fédération de bases de données :

- Jusqu'à nos jours, il n'existe pas des méthodes d'intégration complète qui résolve tous les conflits, aussi moins de méthodes qui peuvent être automatisées.
- Ambiguïté et non clarté du but visé par la phase d'intégration (la simplicité, la complétude, l'exhaustivité).
- Un problème au niveau du choix du modèle pivot, la majorité préfère actuellement le modèle orienté objet à cause sa richesse, mais cela posera les sous problèmes suivants :
  - Lequel des nombreux modèles orientés objets existants est le meilleur pour être choisi, certains préfèrent choisir un modèle avec un minimum de sémantique et pour lequel il n'existe pas trop d'alternatives de modélisation pour la représentation des données.
  - Plus que le modèle choisi est riche des concepts, plus que le processus d'intégration sera plus complexe.

- Lorsque le nombre de schémas ou le nombre de types d'objets composants est important, l'identification de toutes les correspondances, ainsi que la détection des conflits devient une tache très difficile.

- La résolution des conflits de différents types et degrés demande un effort supplémentaire des recherches afin d'obtenir une solution généralisée (par exemple les conflits de structures d'attributs n'ont pas de solutions).
- Problème de compréhension de schémas pauvres en sémantique.
- Peu d'outils automatiques capables d'effectuer la traduction entre les modèles locaux et le modèle commun, sauf pour passer du modèle EA au schéma relationnel, les développements en cours se focalisent sur les traductions entre modèle orienté objet et relationnel.
- Problèmes d'intégration des objets complexes.
- Difficulté d'établir des correspondances complexes entre plus que deux types.
- Complexité de prendre en compte des contraintes d'intégrité et des méthodes.

6 – Approche d'entrepôt de données (Data Warehouse) :

- l'entrepôt de données est orienté beaucoup plus vers l'usage décisionnel.
- La mise en place d'un ED demande une architecture matérielle et logicielle spécifique et de coût très élevé.
- L'approche de l'ED est imposée dans le cas de très gros volumes de données stockées, analysées, traitées, et mises à la disposition des décideurs et des utilisateurs.
- La conception et la mise en place d'un ED demande un travail énorme et un temps important.
- Problème des données mal ciblées.
- Redondance non souhaitable de certaines données et redondance insuffisante de certaine d'autres.
- Période de mise à jour ou d'historisation mal choisie.
- Le transfert des données pour l'analyse est une procédure très coûteuse en temps (représente 80% de l'analyse).
- Difficulté de répercuter les changements des données au niveau des sources de données à intégrer.
- Il faut penser à une équipe pour : la conception, la mise en place du système, la maintenance, l'évolution du système, la formation (concepteurs, administrateurs, développeurs, … etc).

Vu à toutes ces faiblesses, on préfère penser à une autre approche plus générale qui soit : Répandue, performante, simple, standard, …
C'est l'approche d'interopérabilité par XML qui va prendre en charge tous les problèmes et les anomalies précitées, cette approche sera l'objet du prochain chapitre.

Chapitre 4

PROPOSITION D'UNE NOUVELLE APPROCHE D'INTEROPERABILITE DES BASES DE DONNEES

IV.1. Préface :

Après avoir exposer les différentes approches existantes de l'interopérabilité des bases de données hétérogènes et réparties, il était très important de faire une comparaison entre elles, en mettant l'accent sur leurs avantages et leurs inconvénients.

Le résultat de comparaison a favorisé l'approche XML qui est devenue aujourd'hui un standard répandu pour le problème d'interopérabilité et d'échange des données entre bases de données.

Malheureusement, malgré le grand succès de cette approche et sa souplesse, des problèmes d'interopérabilité persistent toujours, surtout pour ce qui concerne la résolution des conflits. Cela nous a amené à proposer une solution qui se bénéficie des avantages des approches étudiées, et qui mette en considération les facteurs suivants :

la clarté, la complétude, la standardisation, l'efficacité et la puissance de résolution des conflits.

IV.2. Principe de l'approche :

Dans notre approche, nous avons essayé de combiner entre les approches :

- Médiation de schémas (Architecture : Médiateur – Adaptateur).
- Médiation de contexte (Approche ontologique)
- L'approche XML.

Nous avons aussi appelé une architecture basée sur le paradigme agents, qui est une tendance nouvelle, et qui reste peu utilisée jusqu'à nos jours.

L'utilisation de l'architecture multi-agents se justifie par le fait que les agents ont des caractéristiques importantes telles que :

- ***L'autonomie :*** qui est la capacité d'un agent à exécuter un certain nombre de taches indépendamment des autres agents **[BEN 2004]**
- ***La coopération :*** qui est la capacité d'un agent à interagir avec d'autres agents en vue de la résolution coopérative d'un problème donné**. [BEN 2004]**

IV.3. L'architecture proposée :

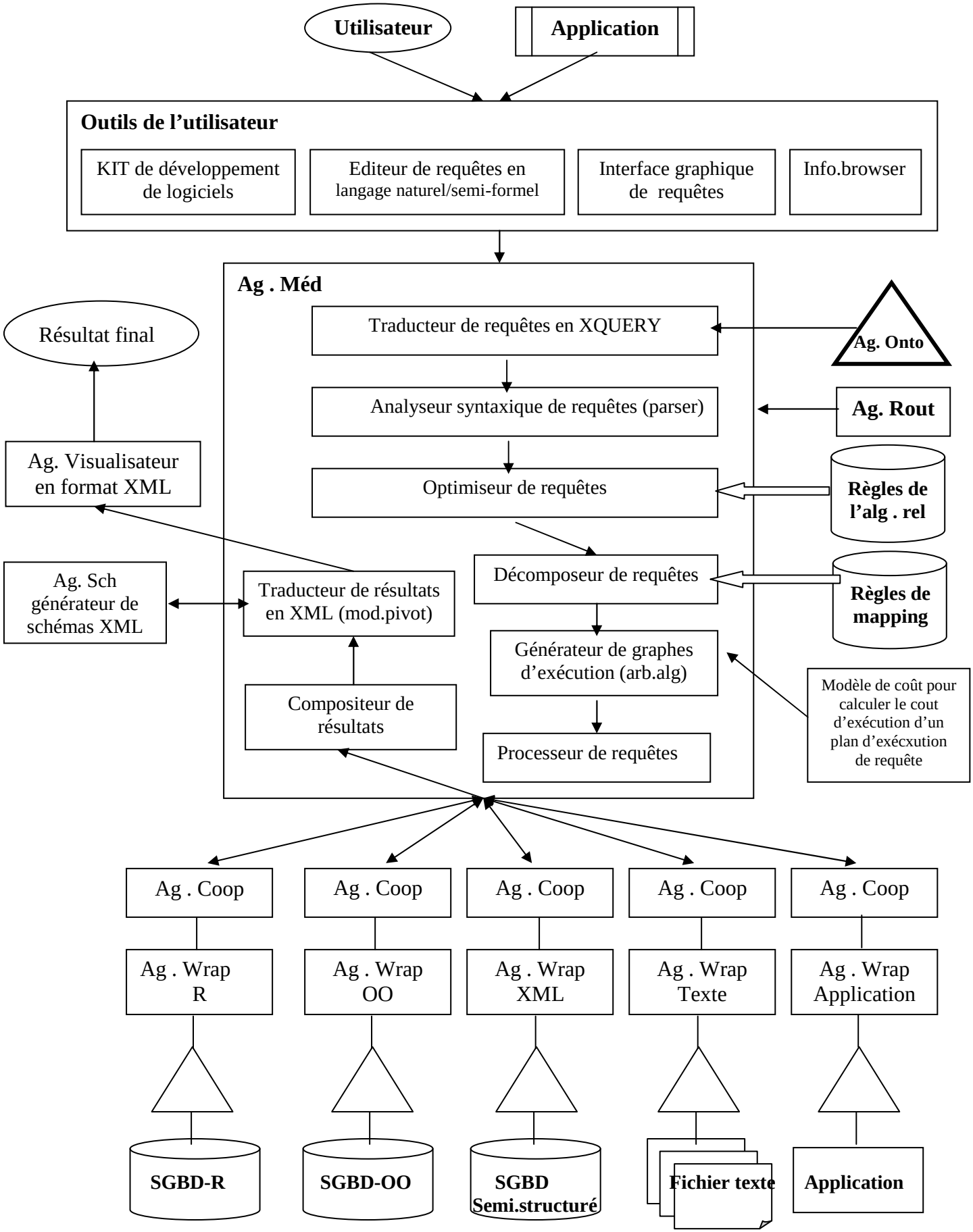


Fig IV.1. Architecture proposée

IV.4. Présentation de l'architecture :

IV.4.1. Définition d'un agent : est un composant logiciel autonome qui participe au fonctionnement global du système **[BEN 2004].** Chaque agent est spécialisé dans la résolution d'un problème ou l'exécution d'une tache particulière.

Nous décrivons chaque agent par :

- Son rôle général au sein de la coopération.
- Les services qu'il offre.
- Les connaissances qu'il possède.
- Les agents avec lesquels il communiquent.

IV.4.2. Les différent agents de l'architecture :

La collection des agents que nous avons proposé dans notre architecture sont :

1 – Agent Ontologie (Ag . Onto) : une ontologie envisage la description des concepts d'un domaine donné et les relations qui peuvent exister entre ces concepts. Les ontologies sont devenues aujourd'hui une solution très répandue au problème d'interopérabilité des bases de données.

Dans notre approche, cette mission importante est confiée à un agent spécifique qui réalise les différentes taches avec plus d'efficacité et de fiabilité.

2 – Agent Wrapper (Ag. Wrap) : il représente une interface entre l'agent médiateur et une source locale de données. Sa mission principale, est l'accès transparent aux données locales de la source. Ces accès sont déclenchés par des demandes d'exécution de sous requêtes mono-sources exprimées dans notre architecture en langage XQUERY.

Le traitement d'une sous requête par Ag. Wrap se réalise en plusieurs phases :

- Une transformation de la sous requête émise dans le langage local de la source.
- L'exécution de sous requête sur le SGBD local.
- La remise en forme des résultats obtenus avant leur envoi vers l'agent émetteur (Ag.Med).

3 – Agent de coopération (Ag. Coop) : on peut le considérer comme le représentant de l'agent adaptateur au niveau de la coopération, il a comme taches :

- Faire la correspondance entre les objets locaux et les concepts de l'ontologie pour leur apporter une sémantique compréhensible par les autres systèmes coopérants.
- Assurer les traductions sémantiques des données et des requêtes du contexte d'un système coopérant au contexte d'un autre système.
- Résoudre quelques conflits, notamment ; les conflits schématiques, les conflits sémantiques de valeurs (différence d'unité, d'échelle, de référentiel, …).

4 – Agent de routage (Ag. Rout) : son rôle est l'enregistrement dynamique des agents qui participent à la coopération. Il stocke des méta-données décrivant ces différents agents. Ces méta-données peuvent être également exploitées pour localiser les systèmes intervenants dans le traitement d'une requête globale.

5 – Agent Médiateur (Ag. Med) : offre une interface entre un utilisateur (ou application) et le système global de coopération en lui offrant une vue unique envers les différentes sources de données.

L'agent médiateur que nous devons utiliser dans notre architecture doit comporter les modules et les composants suivants :

1 – Traducteur de requête : il a comme tache principale l'acquisition d'une requête utilisateur écrite dans son propre contexte et exprimée en langage naturel ou un langage semi-formel proche de sa langue courante, et de la convertir en langage XQUERY de XML.

2 – Analyseur syntaxique de requêtes (Parser) : vérifie le texte de la requête du coté syntaxique et assure sa conformité avec les règles de syntaxe du langage XQUERY.

3 – Optimiseur de requêtes : chaque utilisateur exprime sa demande de sa manière, le rôle de l'optimiseur ici est d'uniformiser les requêtes utilisateurs et de les minimiser le maximum possible en basant sur les règles et les propriétés de l'algèbre relationnelle afin d'économiser le temps nécessaire pour leur exécution.

4 – Décomposeur de requêtes : décompose la requête globale en sous requêtes locales en utilisant l'ontologie du domaine (remplacer chaque concept par son équivalent dans l'ontologie), et en faisant appel aux règles de mapping.

5 – Générateur de graphes d'exécution (plans d'exécution) : son rôle est la génération d'un ensemble de plans d'exécution de requêtes. L'adoption d'un modèle de coût permet d'estimer le temps d'exécution de la requête associée à chaque plan.

6 – processeur de requêtes : permet d'envoyer chaque sous requête vers la source appropriée via un protocole de communication (XML/SOAP) et de l'exécuter selon un plan d'exécution choisi (souvent le plan le plus optimal).

7 – Compositeur des résultats : après avoir exécuter chaque sous requête sur la source invoquée, le compositeur regroupe les différents résultats retournés par les sources en un seul résultat final qui sera délivré à l'utilisateur.

8 – Traducteur de résultats en XML : permet de transformer le résultat final obtenu en format XML, la traduction est réalisée en collaboration avec le générateur de schémas XML.

6 – Agent générateur de schémas XML (Ag. Sch) : son rôle est tout simplement la génération d'un schéma XML qui sera la base pour créer un document XML, l'agent Ag.Sch adopte l'outil de génération de schéma de XML appelé XML schéma.

7 – Agent visualisateur (Ag. Vis) : son rôle est de fournir à l'utilisateur un résultat bien lisible, c'est enfin un document écrit en XML.

IV.4.3. Communication inter-agents :

On préfère réaliser la communication entre les différents agents par des envois de messages de type XML/SOAP.

IV.5. Quelques spécifications sur les ontologies :

Les ontologies sont initialement utilisées pour le partage et la réutilisation de bases de connaissances, puis utilisées dans les systèmes d'interopérabilité des bases de données pour résoudre la majorité des conflits, notamment les conflits sémantiques qui sont les plus difficiles à résoudre pour toutes les approches.

Il existe plusieurs types d'ontologies, le plus connu c'est le type générique. Les ontologies génériques portent sur des concepts généraux qui sont indépendants d'un domaine ou d'un problème particulier tel que les concepts : temps, espace, notions mathématiques, … elles peuvent être utilisées dans des situations diverses et variées.

L'ontologie générique que nous construisons permet :

1 – La mise en œuvre d'une terminologie propre au domaine concerné par

l'interopérabilité afin de pouvoir éviter les conflits sémantiques.

2 – La capitalisation des connaissances.

3 – Le partage sémantique et la communication entre les acteurs des sources de

données.

IV.6. Critères de conception d'une ontologie :

1 – clarté : toutes les ambiguïtés doivent être réduites.

2 – Cohérence : la cohérence des définitions, des concepts et leurs attributs.

3 – Extensibilité : l'ontologie doit être construite de telle manière qu'il soit possible de

l'étendre facilement sans besoin de mettre en cause (refaire) ce qui a été déjà fait.

4 – Engagement ontologique minimal : l'ontologie doit décrire le domaine sans faire

beaucoup d'hypothèses, autrement dit , elle doit contenir un vocabulaire partagé qui ne doit pas être en aucun cas une base de connaissances comportant des connaissances supplémentaires qui dépassent les besoins réels du monde à modéliser .

IV.7. Les composantes d'une ontologie :

1 – Les classes (les concepts) : les classes dans une ontologie représentent les différents concepts concrets ou abstraits liés à un domaine donné.

Une classe peut avoir des sous classes ou des classes descendantes qui représentent des concepts plus spécifiques.

2 – Les relations : Envisagent des liens logiques et sémantiques entre les différents concepts (classes), soit par exemple les relations : sous-classes, synonyme, …etc

3 – Les axiomes : généralement utilisées pour exprimer des contraintes sur les attributs des classes et leurs valeurs correspondantes.

4 – Les instances : représentent les éléments d'un domaine.

5 – Les attributs : représentent quelques propriétés d'une classe.

IV.8. Langages de représentation d'une ontologie :

Dans le cas général , on peut utiliser n'importe quel langage de modélisation de données pour décrire une ontologie. Cependant, ces langages ne sont pas très efficaces pour bien représenter une ontologie.

Les langages de spécification des ontologies offrent des outils spéciaux pour décrire les ontologies, parmi les langages de spécification les plus connus on peut citer :

Ontolingua, Frame Logic, OML/CKML, XML/RDF, UML, OKBC, KIF, … **[NAD 2005]**

Dans notre approche, et pour assurer un certain degré d'homogénéité de l'architecture proposée, nous allons utiliser le langage XML/RDF qui paraît conforme avec notre modèle pivot qui est le modèle XML.

IV.9. Démarche de construction de l'ontologie : [NAD 2005]

1 – Extraction des connaissances et création des concpets : afin de répondre aux besoins d'interopérabilité des bases de données surtout du coté sémantique, nous procédons à la collecte des données en faisant appel aux experts en domaine des bases

de données et d'analyse de données. On peut aussi réutiliser des ontologies déjà existantes.

2 – Conceptualisation : une phase qui consiste à recenser et à établir les liens logiques et sémantiques entre les concepts de l'ontologie.

3 – Ontologie semi-formelle UML : après avoir définir les concepts, établir les liens logique et sémantiques entre eux, on utilise UML pour représenter l'ontologie obtenue d'une manière semi-formelle, c.à.d sous forme de classes.

4 – Formalisation en XML : dans cette phase on procède à implémenter le diagramme des différentes classes de l'ontologie en utilisant le langage de balise XML (XSD).

IV.10. Règles de mapping (règles de correspondance) :

Si une requête est émise par un utilisateur au schéma intégré, elle ne peut pas être satisfaite, sans l'intervention d'un système de mapping qui fait la correspondance entre les structures de données des différentes bases de données réparties sur des sites géographiquement distants, et la structure de données du schéma intégré.

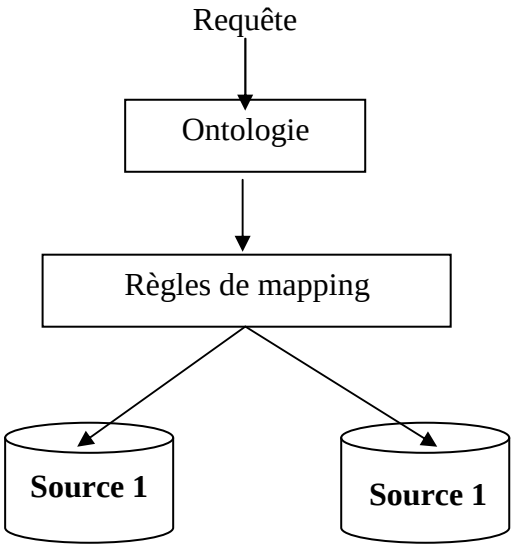


Fig IV.2. Ontologie et règles de mapping

IV.11. Résolution de quelques conflits :

1 – Conflits de nommage :

Si un même concept C est nommé dans le schéma d'une source par C1 et dans le schéma d'une autre source par C2. la solution consiste à le représenter dans l'ontologie par l'un d'eux (C1 ou C2) ou par un autre (soit O), et de définir les règles de mapping appropriées.

par exemple si on a deux concepts exprimés dans les schémas S1, S2 par O1, O2, et soit O l'objet correspondant dans le schéma intégré S.

les règles de mapping convenables peuvent être comme suit :

(S.O = S1.O1), (S.O = S2.O2)

2 – Conflits de généralisation/Spécialisation :

Si pour un concept O du schéma S sont associés plusieurs concepts du schéma S2

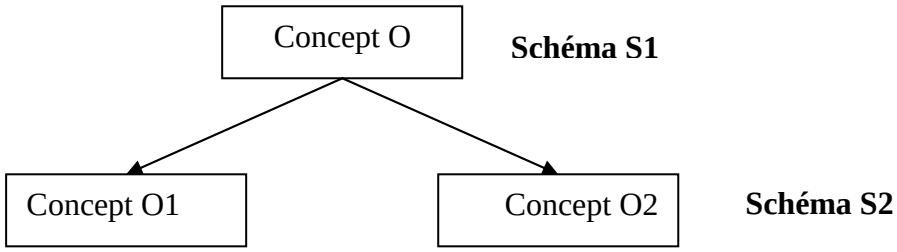


Fig IV.3. Généralisation/Spécialisation

La solution est d'inclure les trois concepts dans l'ontologie basant sur la notion de généralisation/spécialisation, puis on définit les règles de mapping associées.

Exemple :

Soit le concept O du schéma S1, qui subsume les deux concepts O1, O2 du schéma S2.

Cela veut dire que : $O1 \subset O$, $O2 \subset O$ (relation de généralisation/spécialisation).

Pour résoudre ce conflit, il suffit d'inclure les trois concepts O, O1, O2 dans l'ontologie et on définit les règles de mapping suivantes :

(S.O = S1.O1), (S.O1 = S2.O1), (S.O2 = S2.O2).

3 – Conflits de structure .

Deux concepts C1, C2 appartenant à deux schémas différents S1, S2 n'ont pas la même structure, c.à.d le nombre d'attributs qui composent les classes des deux concepts sont différents.

La solution consiste à présenter dans l'ontologie le concept C qui doit regrouper les attributs des deux classes de C1, C2 et de définir les règles de mapping qui facilitent le passage.

Exemple :

Soit un objet O1, constitué de deux attributs A1, A2, on peut écrire O1 = (A1, A2)

Et soit O2 constitué des attributs A1, A2, A3, on l'écrit comme suit : O2 = (A1, A2, A3)

Donc l'objet global sera exprimé dans l'ontologie comme suit : O = (A1, A2, A3)

Les règles de mapping sont alors de la forme :

(S.O = S1.O1), (S.O = S2.O2), (S.O.A1 = S1.O1.A1), (S.O.A1 = S2.O2.A1),

(S.O.A2 = S1.O1.A2), (S.O.A2 = S2.O2.A2), (S.O.A3 = S2.O1.A3).

4 – Conflits de données :

prenons l'exemple de la diversité des unités de mesure ou de l'échelle ou de monnaie.

La solution dans ce cas consiste à définir des fonctions de conversion entre unités ou échelles ou autres.

5 – Conflits d'identificateurs d'objet :

Un même objet peut être identifié d'une manière différente d'une source à l'autre (problème de synonymes).

Comme les autres conflits, ce conflit sera résolu basant sur la notion de l'ontologie et de règles de mapping.

6 – Conflit de modèle :

La résolution de ce genre de conflits est réalisé à travers l'utilisation d'un modèle commun qui est dans notre cas le modèle XML.

IV.12. Conclusion :

Au cours de ce chapitre, nous avons exposé notre solution basée sur la combinaison des approches : médiation de schémas, médiation de contexte, approche XML, et qui adopte une architecture multi-agents.

On a pu aussi exploité d'une façon efficace les notion de : l'ontologie générique, le médiateur, l'adaptateur, et l'agent coopérant, ainsi que d'autres concepts.

L'exploitation des capacités des ontologies génériques et du modèle XML comme modèle pivot nous a permis de résoudre la majorité des conflits envisagés, notamment : les conflits de nommage, de généralisation/spécialisation, de structure, de données, d'identificateur d'objets, de modèle, …

Et comme rien n'est parfait, notre solution pourra être améliorée pour aboutir à des résultats plus satisfaisants.

CONCLUSION

L'interopérabilité entre applications est la capacité de ces applications à se coopérer d'une façon transparente et efficace afin de réaliser certaines fonctionnalités.

Le besoin d'interopérabilité est imposé par :

- La vaste collection de données provenant des sources hétérogènes et réparties surtout avec l'interconnexion massive au réseau Internet ou intranet.
- Les nouvelles applications qui requièrent souvent des accès coordonnés à plusieurs sources de données à cause de leurs esprits collaboratifs.

Parmi les objectifs visés par une telle interopérabilité entre applications et surtout entre bases de données on cite les suivants :

- Permettre à un utilisateur (personne ou application) d'utiliser des données issues d'un ensemble de systèmes d'informations autonomes, répartis et hétérogènes.
- Développer des architectures (infrastructures, protocoles d'accès, outils logiciels) pour : le partage, l'échange et le contrôle de données.

La réalisation de systèmes d'informations interopérables nécessite l'accès à de multiples sources de données distribuées et hétérogènes. Cela rencontre plusieurs problèmes, dont on a pu résoudre certains d'eux au long de notre étude, il s'agit de :

1 – La modélisation et l'interrogation de multiple sources de données distribuées et hétérogènes.
2 - L'interopérabilité des bases de données hétérogènes et distribuées.
3 – L'extraction efficace des connaissances depuis de grandes bases de données.

Notons que l'objectif principal de notre étude était de trouver des solutions efficaces et performantes au problème de l'interopérabilité des bases de données hétérogènes et réparties. la résolution des autres problèmes posés par l'introduction pourront être l'objet de recherches futures dans le même domaine.

Notre étude nous a permis d'atteindre tous les objectifs fixés au préalable, notamment :

1 – Comprendre le problème de l'interopérabilité entre bases de données hétérogènes et réparties, et définir sa position et son importance dans le domaine de communication et d'échange de données entre systèmes d'informations.

2 – L'acquisition de compétences importantes dans le domaine de consolidation des bases de données hétérogènes et réparties à travers les divers approches permettant d'intégrer et par conséquent de mieux exploiter ces données.

3 –On est arrivé à exposer plusieurs architectures (à travers les différentes approches) qui permettent un très haut niveau d'interopérabilité des bases de données hétérogènes autonomes et distribuées, à noter :

- L'architecture (Médiateur – Adaptateur) ou bien l'architecture de médiation de schémas dont DARPA I3 est son exemple le plus idéal.
- L'architecture de médiation de contexte qui assure un haut niveau d'interopérabilité sémantique en basant sur les notions : ontologies, contextes, ainsi que des opérations de transformations et de coopération.

- L'architecture des bases de données fédérées qui propose une vue globale intégrée des différentes bases de données participant sous forme d'une base de données unique virtuelle.
- L'architecture entrepôt de données exigeant un environnement matériel et logiciel un peu spécifique et qui peut traiter des flux d'informations de volumes très importants d'une entreprise.
- L'architecture mise en jeu par la solution XML qui paraît selon notre étude la meilleure, vu sa simplicité, son efficacité et son habilité d'être très vite améliorée.

Toutes les architectures précitées permettent un accès transparent, unique, homogène, cohérent, et intégré aux bases de données hétérogènes et distribuées. C.à.d elles assurent une meilleure interopérabilité en tenant compte des profils des utilisateurs et des différents contextes d'exécution, et en conservant ainsi, pour chaque base de données son autonomie en terme : SGBD, structures de données, les données elles mêmes.

A la fin de notre étude, nous avons proposé une nouvelle approche basant sur la combinaison des approches les plus répandues à savoir : l'approche de médiation de schémas, l'approche de médiation de contexte, et l'approche XML.

Dans notre approche, nous avons aussi adopté une tendance nouvelle, peu utilisée qui est l'architecture multi-agents.

Notons qu'un agent est un composant logiciel capable de réaliser ses taches d'une façon autonome et avec un degré très élevé de performance. Il est aussi facile de faire coopérer les différents agents de notre architecture avec le moindre de coût.

Perspectives :

1 – Résoudre les autres problèmes qui font face à la réalisation d'un tel système interopérable, dont certains ont été exposé au cours de notre introduction, tels que :

- la gestion des transactions réparties sur plusieurs sites distants.
- l'évolution et l'optimisation des performances des middlewares invoqués par les différentes approches d'interopérabilité.

2 – Etudier d'autres nouvelles approches d'interopérabilité telles que :

- L'approches de services WEB.
- L'approche du WEB sémantique.

En présentant pour chacune ses avantages et ses inconvénients, et en traitant les solutions qu'elle apporte au problème posé dans notre travail.

3 – Etudier avec plus de détail l'architecture XML en mettant l'accent sur les points suivants :

- Les serveurs de données XML (EXCELON d'Object Design comme exemple).
- Le langage de requêtes XQUERY.
- Optimisation des requêtes et des transactions sur des sites hétérogènes et réparties.
- Etudier la possibilité de mettre en œuvre l'architecture XML pour des postes clients et serveurs mobiles (serveur mobile comme l'exemple de cartes à puces).

4 – Améliorer notre architecture proposée pour atteindre une meilleure interopérabilité.

5 – Définir des environnements qui permettent l'implémentation de notre solution.

Tableau récapitulatif des fautes et leurs corrections.

Erreur	N° page	N°Ligne	correction
Son précieux assistance	Page de remerciements		Sa précieuse assistance
La commerce électronique	02	02	Le commerce électronique
Sont exposés	02	42	Sont exposées
à collaborer	04	05	à se collaborer
XAURY	13	06	XQURY
Mise à jour	23	14	Mis à jour
répartie	26	03	répartis
collaborer	26	04	Se collaborer
Contexte 1,1,1	Schéma , page 32		Contexte 1,2,3
Source 2	Schéma 33		Source 04
II.5.3.2	37	23	II.6.3.2
II.5.3.3	38	01	II.6.3.3
Contexte globale	45	26	Contexte global
Mentionner	62	14	Mentionné
Et vise versa	72	39	A éliminer
Fouille	99	21	Feuille
répondues	106	14	répandues
E fonction	107	06	En fonction
Répondue	108	38	répandue
répondue	112	24	répandue
pourront	112	24	pourrant
protocole	123	21	protocole
In interpréteur	125	24	Interpréteur
Il est	128	03	elle est
Il contient	128	03	elle contient
mettent face	157	26	font face

TABLE DES FIGURES

TABLE DES APPROCHES ET DES TECHNIQUES

TABLE DES TABLEAUX

27 - 108

CHAPITRE II : DIFFERENTES APPROCHES D'INTEROPERABILITE

Résumé :

L'interopérabilité des bases de données et des SGBD, ou d'une manière générale entre systèmes d'informations hétérogènes et répartis, est devenue une nécessité pour répondre aux besoins d'échange et de communication. elle prend aujourd'hui une large place surtout avec l'interconnexion massive des systèmes d'informations via Internet et intranet ou extranet .

l'interopérabilité peut être définie par la capacité des systèmes d'informations à se collaborer, même s'ils ont des natures très différentes, afin de réaliser des fonctionnalités communes .

Notre étude est concentrée sur l'interopérabilité des bases de données hétérogènes et réparties. On est arrivé à présenter un état de l'art du domaine, à exposer les différentes approches conçues pour réaliser cette interopérabilité, dont les plus connues sont :

* l'approche de médiation de schémas qui met l'accent sur deux composants fondamentaux : le médiateur et l'adaptateur .
* L'approche de médiation de contexte orientée sémantique, qui exploite les capacités des ontologies, des contextes de coopération, et qui peut résoudre la plupart des conflits sémantiques .
* L'approche de fédération qui base essentiellement sur la notion de l'intégration des données suivant un modèle commun appelé modèle pivot .
* L'approche entrepôt de données orientée beaucoup plus aux besoins des systèmes décisionnels, surtout au niveau des entreprises qui reçoivent et traitent des flux très importants d'informations .

L'étude comparative sur ces différentes approches a aboutit à un résultat très évident en faveur de l'approche XML, comme étant un standard très répandu d'échange et d'interopérabilité .

L'étude est terminée par la proposition d'une nouvelle approche qui combine entre les trois approches, à savoir : l'approche de médiation de schémas, de médiation de contexte, et l'approche XML, et qui adopte une architecture multi-agents .

Mots clés : ***Bases de données hétérogènes, Bases de données réparties, Bases de données orientées objet, Bases de données relationnelles, Interopérabilité, Intégration, Fédération, Médiation, Médiateur, Adaptateur, Ontologie, Entrepôt de données, Langage XML, Optimisation de requêtes, Web service, Web sémantic.***

Abstract :

The data bases and DBMS interoperability, or in generally, the interoperability of heterogeneous and distributed information systems , became a necessity to answer to the needs of exchange and communication. It takes today a very large place, especially with the extensive interconnection of information systems via internet and intranet or extranet .

the interoperability can be defined by the ability of information systems to collaborate itself, even though they have some very different natures, in order to achieve some common functionalities .

Our study is concentrated on the heterogeneous and distributed data bases interoperability .we have focused our attention to present a state of the art of the domain, and to expose the different existing approaches conceived to achieve this interoperability, the most known approches are :

* The approach of diagram mediation that puts the accent on two fundamental components : the mediator and the adapter .
* The approach of context mediation oriented semantics, that exploits capacities of ontologies, of cooperation contexts, and that can solve most of the semantic conflicts .
* The approach of federation that is essentially based on the notion of the data integration following a common model called pivot model .
* The approach of data warehouse oriented to the needs of decisional systems, especially in companies that receives and treat a very important fluxes of information .

The comparative study on these different approaches has leads to a very obvious result in favor of the XML approach, as being the best standard for exchange and interoperability .

The study is finished by our proposition of a new approach that combines between three approaches : the approach of diagram mediation, of context mediation, and the XML approach, and that adopts a multi-agents architecture.

Key words : ***Heterogeneous data bases, distributed data bases, Oriented object data bases, Relational data bases, Interoperability, Integration, Federation, Mediation, Mediator, Adapter, Ontology, Data warehouse , XML language, Queries optimization, Web service, semantic web,,,.***