Contributions à la conception et l'exploitation de systèmes d'intégration de données

Ladjel Bellatreche

ENSMA : Ecole Nationale Supérieure de Mécanique et d'Aérotechnique
LISI : Laboratoire d'Informatique Scientifique et Industrielle

Mémoire

pour l'obtention du diplôme

D'HABILITATION A DIRIGER DES RECHERCHES

(Faculté des Sciences Fondamentales et Appliquées)

(Diplôme National — Arrêté du 7 août 2006)

Ecole Doctorale : Science Pour l'Ingénieur
Secteur de Recherche : INFORMATIQUE ET APPLICATION

Présenté par :

Ladjel BELLATRECHE

****

Contributions à la conception et l'exploitation de systèmes d'intégration de données

****

Soutenu le
devant la Commission d'Examen

****

Rapporteurs :	**Christine COLLET**	Professeur, Institut Polytechnique de Grenoble
	Stefano SPACCAPIETRA	Professeur, EPFL, Lausanne Suisse
	Michel SCHOLL	Professeur, CNAM, Paris
	Tamer ÖZSU	Professeur, Université Waterloo, Canada
Examinateurs :	**Jean-Marc PETIT**	Professeur, INSA, Lyon
	Chantal REYNAUD	Professeur, Université Paris Sud
	Guy PIERRA	Professeur, ENSMA, Poitiers, Directeur de Recherche
	Yamine AÏT AMEUR	Professeur, ENSMA, Poitiers

A mes Parents,
Hayette,
Houda, Loulou, Kenza,
Mes frères et soeurs,
ma famille,
mes amis.

Table des matières

Chapitre 1

Le contexte : problématique et domaines d'application

L'information disponible sous forme numérique a connu et connaît aujourd'hui encore un essor spectaculaire. Cette évolution est due à de nombreuses causes qui incluent : l'explosion de sources de données et d'applications disponibles sur le Web, le développement des solutions réseaux comme le haut débit, les autoroutes de l'information, des méthodes de construction et de validation de logiciels sécurisés, le changement des habitudes traditionnelles de gestion de données des entreprises, l'émergence de XML, etc. Cette évolution a fait naître un besoin crucial d'échanges et de partages de l'énorme quantité d'informations provenant de diverses sources de données éparpillées sur la Toile ou au sein de différentes organisations contenant des informations pertinentes et complémentaires sur des produits, les clients, les fournisseurs, les opérations de l'entreprise et les applications. Ces échanges et partages se matérialisent souvent par l'intégration des sources données. Un système d'intégration consiste à fournir une interface unique, uniforme et transparente aux données via un schéma global. Les domaines d'application de cette intégration incluent en particulier : l'Intranet des entreprises, les collaborations ponctuelles entre les entreprises, la fusion des entreprises (effet par exemple de la crise économique, etc.), le commerce électronique, l'informatique décisionnelle.

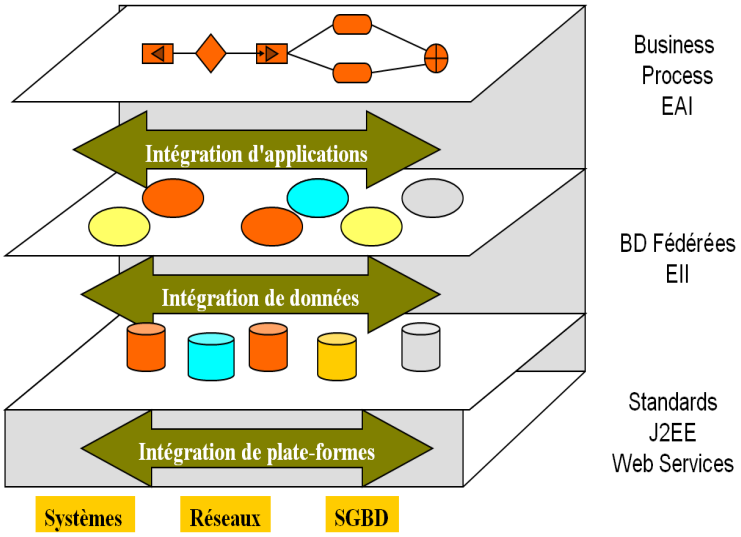


FIG. 1.1 – Différents niveaux d'intégration [96]

Un nombre important de projets de recherche sur les systèmes d'intégration de données a été proposé dans la littérature: on peut ainsi citer Padoue [142], TSIMMIS [70], Picsel [92], MOMIS [45]. En 2005, la conférence SIGMOD a organisé deux sessions ; une académique et une autre industrielle sur l'intégration de données. Ces travaux de recherche ont donné naissance à une nouvelle industrie de l'intégration. Selon une étude d'IBM, pour 1$ dépensé pour une application packagée, 5 à 9$ sont dépensés pour assurer son intégration. L'intégration a trois dimensions : l'intégration des données (*Entreprise Information Integration* (EII)), l'intégration des applications (*Entreprise Application Integration* (EII)) et l'intégration des plateformes (Figure 1.1). Nos travaux de recherche ont porté sur l'intégration de données.

1.1 Problèmes liés à l'intégration des données

Le processus d'intégration de données a pour entrée un ensemble de sources de données (schémas et populations), et produit en sortie une description unifiée des schémas initiaux (le schéma intégré) et les règles de correspondances qui permettent l'accès aux données sources à partir du schéma intégré.

La mise en oeuvre des systèmes d'intégration de données est une tâche difficile. Cette difficulté est due à plusieurs facteurs : (a) le nombre croissant des sources de données à intégrer, (b) le fait que la sémantique des sources soit peu explicitée, (c) l'hétérogénéité des sources, (d) l'autonomie des sources et (e) l'évolution des sources.

- *Le nombre croissant de sources* : le nombre de sources de données participant au processus d'intégration ne cesse d'augmenter, en particulier dans les domaines comme le commerce électronique, la biologie, l'environnement, l'ingénierie, etc. Cette augmentation a fait naître un besoin d'automatiser la construction de système d'intégration.
- *La sémantique de sources de données est peu explicitée*. La plupart des sources de données candidates à l'intégration n'ont pas été conçues pour être intégrées dans le futur mais pour satisfaire des applications de tous les jours. Souvent, le peu de sémantique contenu dans leurs modèles conceptuels (dans le cas, où chaque source est une base de données) est perdu par le fait que leurs modèles logiques sont utilisés par les applications. Le modèle conceptuel permet d'exprimer à la fois les besoins applicatifs et la connaissance du domaine sous une forme intelligible pour un utilisateur ultérieur. Malheureusement, c'est le modèle logique qui est exploité ; et celui-ci, résultant de la normalisation et de l'adaptation au système support, est en général très différent du modèle conceptuel. Ainsi, l'absence de représentation du modèle conceptuel ou de toute autre représentation sémantique dans les bases de données rend l'interprétation et la compréhension de celle-ci très difficile même pour les concepteurs ayant une bonne connaissance du domaine d'application.
- *L'hétérogénéité des données*. Elle concerne à la fois la structure et la sémantique. L'hétérogénéité structurelle provient du fait que les sources de données peuvent avoir différentes structures et/ou de différents formats pour stocker leurs données. L'autonomie des sources fait augmenter cette hétérogénéité d'une manière significative. En effet, les sources de données sont conçues indépendamment par des concepteurs différents ayant des objectifs applicatifs différents. Chacun a donc une approche structurelle différente. De nombreux travaux concernant ce type d'hétérogénéité ont été proposés dans les contextes des bases de données fédérées et des multi-bases de données. L'hétérogénéité sémantique, par contre, présente un défi majeur dans le processus d'élaboration d'un système d'intégration. Elle est due aux différentes interprétations des objets du monde réel.

– *Autonomie des sources*. La plupart des sources participant à l'intégration de données fonctionnent en mode totalement autonome. Autrement dit, les administrateurs de ces sources sont tout à fait libres de modifier leur schéma ou mettre à jour leur contenu, sans en faire état préalablement aux utilisateurs.
– *Dynamicité des sources*. Les fournisseurs des sources de données étant différents, chaque source doit pouvoir, en plus, se comporter indépendamment des autres (tout particulièrement dans des environnements dynamiques comme le Toile). En conséquence, la relation entre le système intégré et ses sources est faiblement couplée. Une source doit pouvoir modifier sa structure et sa population sans en informer les autres et sans que cela engendre des anomalies de maintenance du système d'intégration. Dans un tel contexte asynchrone, l'évolution du système d'intégration concerne à la fois les schémas et les populations intégrées.

La construction d'un système d'intégration doit prendre en compte ces différents aspects. Une fois construit, le système d'intégration, comme tout système informatique, doit fournir des outils pour faciliter son exploitation. Cette dernière peut être assurée par la proposition des structures d'optimisation. Vu le nombre important d'utilisateurs qui accèdent aux systèmes d'intégration, chacun avec un profil, des préférences et un dispositif d'affichage spécifiques, il serait souhaitable de leur offrir une interface selon leur profil afin de réduire la masse de données délivrées à l'utilisateur et augmenter la pertinence des résultats produits.

1.2 Vers une démarche de développement des systèmes d'intégration de données

La majorité des travaux sur les systèmes d'intégration se concentre principalement sur une seule étape de leur cycle de vie : la construction, l'exploitation efficace ou la personnalisation. Pour fournir des solutions d'intégration de données englobant les trois phases, nous avons travaillé sur chacune des phases afin de proposer une démarche de développement de systèmes d'intégration comportant deux phases de conception ; une *en amont* incluant la phase de construction et une *en aval* comprenant les phases d'exploitation et de personnalisation. La nécessité de proposer une démarche de développement d'un système d'intégration est motivée par le fait que les phases de construction d'un système d'intégration sont fortement dépendantes. Par exemple, l'une des entrées de la phase d'exploitation est le système d'intégration.

1.2.1 Conception en amont d'un système d'intégration

La construction est la première étape de notre démarche de développement d'un système d'intégration (Figure 1.2). Elle concerne quatre étapes que nous avons identifiées dans la thèse de Dung NGUYEN XUAN [217] : (1) la définition du schéma global et la mise en correspondance entre les schémas locaux et le schéma global, (2) le choix de l'architecture du système intégré, (3) la résolution des conflits et (4) la gestion de l'évolution.

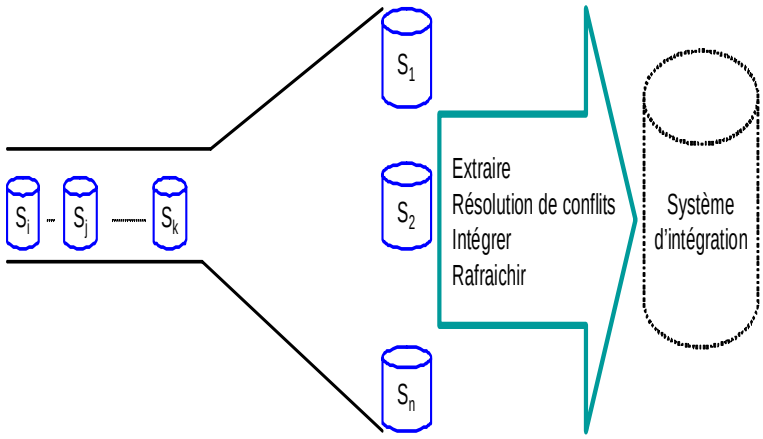


FIG. 1.2 – Les composants de la phase en amont de conception

1.2.1.1 Définition du schéma global et mise en correspondances avec les schémas locaux

Cette tâche consiste à définir, ou construire, le schéma global et à définir les relations qui existent entre le schéma global et les schémas locaux. La correspondance des schémas a été largement étudiée par la communauté des bases de données (voir [182] pour un état de l'art sur ces travaux). Deux principales approches ont été proposées pour assurer la correspondance entre les schémas : GaV (Global as View) et LaV (Local as View).

L'approche GaV a été la première à être proposée pour intégrer des informations. Elle consiste à définir à la main (ou de façon semi-automatique) le schéma global en fonction des schémas des sources de données à intégrer puis à le connecter aux différentes sources. Pour cela, les prédicats du schéma global, aussi appelés relations globales, sont définis comme des vues sur les prédicats des schémas des sources à intégrer. Comme les requêtes d'un utilisateur s'expriment en termes des prédicats du schéma global, on obtient facilement une requête en termes de schéma des sources de données intégrées, en remplaçant les prédicats du schéma global par leurs définitions. Parmi les systèmes utilisant Ga, on peut citer TSIMMIS [70] et MOMIS [45].

L'approche LaV est l'approche duale, elle suppose l'existence d'un schéma global et elle consiste à définir les schémas des sources de données à intégrer comme des vues du schéma global. Les principaux systèmes développés autour de cette approche sont: Infomaster [99], PICSEL [102], Information Manifold [139]. Lors de l'ajout d'une nouvelle source, le système d'intégration ajoute des nouvelles règles représentant la vue de la nouvelle source.

D'autres approches combinant GaV et LaV ont également été proposées, notamment GLAV, BaV (both as view), etc.

On considère souvent que l'adjonction d'une nouvelle source est plus facile dans l'approche LaV que dans l'approche GaV [111]. En fait, cela dépend de l'hypothèse faite concernant (1) les concepts contenus dans le schéma global, et (2) les concepts existants (ou auxquels le système d'intégration doit donner accès) dans la nouvelle source. Dans l'hypothèse où une nouvelle source ne doit pas modifier le schéma global (hypothèse faite dans les systèmes LaV), soit qu'elle ne contienne aucun nouveau concept,

soit que le schéma global ne représente que partiellement les différentes sources, l'effet d'adjonction est tout à fait similaire. Le coût de modification de l'implémentation de la vue globale, dans l'approche GaV, est contre balancée, dans l'approche LaV, par le coût de définition du schéma local comme vue du schéma global, puis par le coût de réécritures de requêtes en fonction des vues.

1.2.1.2 Le choix de l'architecture d'un système d'intégration

Deux principales architectures existent pour supporter un système d'intégration : médiateur [119] et matérialisée [120]. Dans l'architecture médiateur, l'intégration des données s'effectue en deux étapes : (i) le système d'intégration génère, à partir d'une requête d'un utilisateur, autant de sous requêtes qu'il y a de sources de données à interroger, (ii) il construit alors la réponse finale à partir des résultats de sous requêtes et la présente à l'utilisateur. L'architecture virtuelle consiste à développer une application chargée de jouer le rôle d'interface entre les sources de données locales et les applications d'utilisateurs. Ce type d'approche a été utilisé dans un nombre important de projets [70, 99, 139, 148, 185, 211, 143]. Il repose sur deux composants essentiels: le *médiateur* et l'*adaptateur* [119].

Dans l'architecture matérialisée, l'intégration des données s'effectue également en deux étapes : (i) les données issues de différentes sources sont dupliquées dans un entrepôt de données, (ii) les requêtes des utilisateurs sont posées directement sur ce dernier. L'avantage principal d'une telle approche est que l'interrogation se fait directement sur les données de l'entrepôt et non sur les sources originales. On peut donc utiliser les techniques d'interrogation et d'optimisation des bases de données traditionnelles. Par contre cette approche exige un coût de stockage supplémentaire et surtout un coût de maintenance causé par les opérations de mises à jour au niveau de sources de données (toute modification dans les sources locales doit être répercutée sur l'entrepôt de données). Quelques projets spécifiques d'entrepôts de données servent actuellement de références, en particulier, le projet européen DWQ (Data Warehouse Quality) [124] et le projet WHIPS à l'Université de Stanford - USA [112, 136]. Cette architecture a également été adoptée par les applications décisionnelles.

1.2.1.3 La résolution de l'hétérogénéité

On peut distinguer trois principales générations d'approches pour la résolution de l'hétérogénéité. Ce sont les approches manuelles, semi automatiques et automatiques. Ce critère prend une importance croissante avec intégration d'un nombre de plus en plus important de sources de données indépendantes.

1.2.1.3.1 La génération d'intégration manuelle Les méthodes d'intégration de données manuelles correspondent à la première génération de systèmes d'intégration. Cette génération est apparue dans les années 90 [61, 150, 172, 195]. A cette époque, les travaux d'intégration de données visaient essentiellement à automatiser l'interopérabilité syntaxique de données. Cependant, les conflits sémantiques étaient résolus d'une façon manuelle, puisque seul un observateur humain pouvait interpréter la sémantique des données, ces approches laissaient donc à l'administrateur (ou l'utilisateur) la responsabilité du processus d'intégration (Figure 1.3).

D'après Parent et al. [172], les approches d'intégration manuelles visent à fournir des langages de manipulation de schémas que l'administrateur ou l'utilisateur peuvent utiliser pour construire lui-même

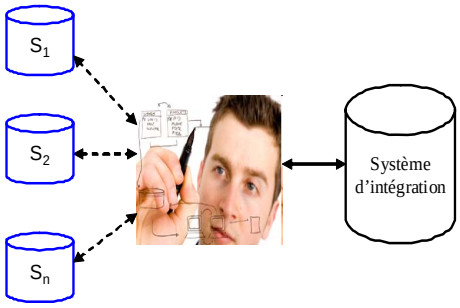


FIG. 1.3 – Résolution manuelle de l'hétérogénéité

(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 alors automatiquement les règles de traduction entre les schémas initiaux et le schéma intégré. Les langages déclaratifs sont plus faciles à utiliser, mais l'établissement des règles de traduction par le système est plus délicat. Les stratégies manuelles supposent la connaissance par l'administrateur de la structure du schéma intégré. Elles étaient largement utilisées dans les *système de multi-bases de données* [61], les bases de données fédérées, etc.

Par contre, dans les environnements qui nécessitent d'intégrer un nombre important de sources de données réparties qui évoluent fréquemment, l'intégration de données manuelle devient *très coûteuse et souvent même impossible*. Les traitements plus automatisés ont donc été conçus pour faciliter la résolution des conflits sémantiques. Ceci correspond aux deux générations suivantes.

1.2.1.3.2 Intégration semi-automatique à l'aide d'ontologies linguistiques (ou thésaurus) Afin d'atteindre une intégration, au moins partiellement automatique de différentes sources de données, plusieurs groupes de recherche ont développé des techniques d'intégration basées sur des ontologies, issues de la communauté de l'intelligence artificielle. Le rôle d'une ontologie est de servir de pivot pour définir la sémantique des différentes données à l'aide de concepts communs et formalisés compréhensibles et admis par tous les participants (utilisateurs).

La deuxième génération de systèmes d'intégration utilise des ontologies linguistiques (qui sont également appelées des thésaurus) pour identifier automatiquement ou semi-automatiquement d'abord les termes clés du domaine puis quelques relations sémantiques entre les termes utilisés dans les sources de données. Nous appelons Ontologies Linguistiques (OL), les ontologies qui décrivent l'ensemble des termes apparaissant dans la description langagière d'un domaine et les relations qui les relient. Dans cette catégorie d'ontologies, outre des relations entre concepts représentées par des termes (par exemple, "subsumée par" pour la relation de subsomption), des relations entre termes (par exemple, la synonymie ou l'homonymie) sont également définies. Les relations entre les termes étant souvent fortement contextuelles, les inférences automatiques basées sur ces ontologies nécessitent, toujours, la supervision d'un expert. Les OL aident à reconnaître des similarités conceptuelles entre des phrases même si différents termes sont utilisés. Néanmoins, puisque la signification des termes est contextuelle et que les relations entre termes sont approximatives, de fausses similarités peuvent être produites et les résultats ne peuvent

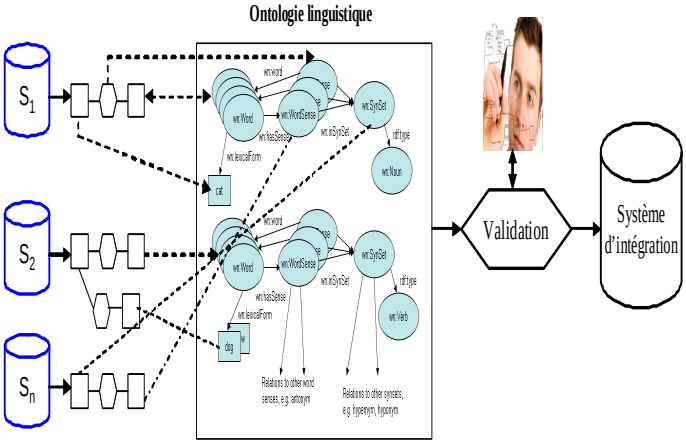


FIG. 1.4 – Résolution semi automatique de l'hétérogénéité

jamais être exploité en l'état.

On peut cependant automatiquement comparer les noms d'objets, de relations ou d'attributs et essayer d'identifier les éléments similaires en utilisant des OL. Les OL restent cependant orientés "terme" et non "concept". Cela pose notamment des problèmes d'homonymie et de dépendance vis-à-vis d'un langage. Ce type d'ontologies est également lourd à développer et à mettre à jour. Certains travaux d'intégration à base OL utilisent des mesures d'affinité et de similarité afin de calculer la vraisemblance de relations entre concepts [48, 45, 137]. Ces mesures sont associées aux seuils définis par les concepteurs pour décider si la relation est retenue. De telles procédures compromettent la qualité du système d'intégration. C'est pourquoi ce type d'approche ne peut être complètement automatique que pour l'interprétation, automatique, mais approximative de documents. Son travail pour concevoir un système d'intégration de données est nécessairement supervisé par un expert et ne peut être que partiellement automatique (Figure 1.4).

Deux thésaurus assez connus sont utilisés dans ce genre d'approche: WORDNET et MeSH. MeSH (Medical Subject Heading)[1] est un thésaurus médical. Le projet MOMIS [45] est un exemple de projet d'intégration utilisant des OLs qui vise à intégrer semi-automatiquement des données de sources structurées et semi structurées.

1.2.1.3.3 Intégration automatique à l'aide d'ontologie conceptuelle Dans les deux générations précédentes, la sémantique des sources n'est pas explicitée. La troisième génération des systèmes d'intégration consiste à associer aux données une ontologie conceptuelle (*OC*) qui en définit le sens. Une ontologie conceptuelle est "*une spécification explicite et formelle d'une conceptualisation faisant l'objet d'un consensus*" [176]. Dans cette conceptualisation, le monde réel est appréhendé à travers des concepts représentés par des classes et des propriétés. Les mots d'un langage naturel peuvent être associés aux concepts pour contribuer à leurs définitions. Mais c'est l'ensemble des caractéristiques associées à un concept ainsi que ses liens formels avec les autres concepts qui en définissent le sens. Une *OC* regroupe ainsi les définitions d'un ensemble structuré de concepts. Ces définitions sont traitables par machine et

[1] http://www.chu-rouen.fr/cismef/

partagées par les utilisateurs du système. Elles doivent, en plus, être explicites, c'est-à-dire que toute la connaissance nécessaire à leur compréhension doit être fournie dans l'OC.

La référence à une telle ontologie est alors utilisée pour éliminer automatiquement les conflits sémantiques entre les sources dans le processus d'intégration de données. L'intégration de données est donc considérée comme automatique. Nous pouvons citer dans cette approche les projets PICSEL [92], OBSERVER [148], OntoBroker [79], KRAFT [211], COIN [100]. En comparaison avec les approches d'intégration utilisant des ontologies linguistiques, ces approches sont:

1. plus rigoureuses: car les concepts sont définis avec plus de précision, l'appariement peut être plus argumenté,
2. facilement outillables: si les différentes ontologies sont basées sur le même modèle, comme OWL [146], PLib [179], etc. Une correspondance entre ontologies peut être exploitée par des programmes génériques pour intégrer les données correspondantes.

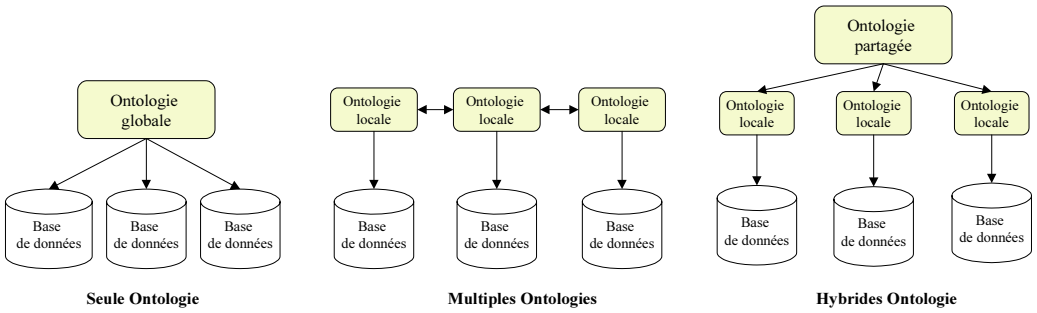


FIG. 1.5 – Différentes architectures d'intégration à base ontologique proposées par Wache et al. [212]

Plusieurs approches d'intégration à base ontologique ont été développées [212]. Ces dernières peuvent être divisées en trois catégories: approches avec une seule ontologie, approches avec ontologies multiples et approches hybrides (figure 1.5). Dans l'approche avec une seule ontologie, chaque source référence la même ontologie globale de domaine. Les systèmes d'intégration SIMS [6] et COIN [100] sont des exemples de cette approche. La conception de tels systèmes ressemble à la conception de bases de données réparties homogènes [169]. En conséquence, une nouvelle source ne peut ajouter aucun nouveau concept sans exiger le changement de l'ontologie globale. Dans l'approche à multiples ontologies (exemple du projet OBSERVER [148]), chaque source a sa propre ontologie développée indépendamment des autres sources. Dans ce cas, les correspondances inter-ontologies sont difficiles à mettre en oeuvre. L'intégration des ontologies est donc faite d'une façon manuelle ou semi-automatique. [148]. Pour surmonter les inconvénients des approches simples ou multiples d'ontologies, l'approche hybride a été proposée. Dans cette dernière chaque source a sa propre ontologie, mais toutes les ontologies utilisent un vocabulaire partagé commun (exemple du projet KRAFT [211]). Dans cette catégorie, l'intégration de données peut tendre à être automatique. C'est dans cette catégorie que s'inscrivent nos travaux.

L'ontologie locale peut être soit à l'intérieur (stockée avec les données sources), soit à l'extérieur de la source. Récemment, un nombre important de projets propose d'embarquer l'ontologie locale dans chaque source. Chaque ontologie locale référence une ontologie partagée. Ce type de source est appelée source de données à base ontologique. Les données de ce type des sources sont appelées *les données à base ontologique*.

Les ontologies conceptuelles permettent aux sources participant au processus d'intégration d'expliciter leurs sémantiques dans le but d'automatiser le processus d'intégration sémantique de données.

1.2.1.3.4 Bilan sur les ontologies conceptuelles La présence d'ontologie locale dans chaque source augmente la richesse sémantique des données et la rend explicite.

Les ontologies, comme les modèles conceptuels, conceptualisent l'univers du discours au moyen de classes hiérarchisées par subsomption associées à des propriétés. Les ontologies présentent cependant quatre différences par rapport aux modèles conceptuels. Ces différences sont à la base de la contribution que peuvent apporter les ontologies dans la problématique des bases de données.

- **Objectif de modélisation**. Une ontologie *décrit les concepts d'un domaine* d'un point de vue assez général, indépendamment de chaque application précise et de tout système dans lequel elle est susceptible d'être utilisée. A l'opposé, un modèle conceptuel de données *prescrit l'information* qui doit être représentée dans un système informatique particulier pour faire face à un besoin applicatif donné. Les éléments de la connaissance du domaine représentés par une ontologie et pertinents pour un système particulier pourront donc être extraits de l'ontologie par le concepteur du système sans que celui-ci ait besoin de les redécouvrir. Ainsi par exemple, la norme IEC 61360-4 (IEC 61360-4 1998) constitue un dictionnaire formel et consensuel de la plupart des catégories de composants électroniques existants, ainsi que leurs propriétés et des relations qui les relient. Ce domaine constitue donc une description de référence du domaine des composants électroniques dont un concepteur de bases de données peut aisément extraire, en interaction avec les utilisateurs, les entités, propriétés et relations devant être représentées au sein d'une base de données d'ingénierie d'une entreprise de conception électronique. Une démarche tout à fait analogue pourrait exister dans le domaine bancaire pour autant qu'une ontologie des produits bancaires usuels soit formalisée.
- **Atomicité des concepts**. A la différence d'un modèle conceptuel de données, où chaque concept ne prend son sens que dans le contexte du modèle dans lequel il est défini, dans une ontologie, chaque concept est identifié et défini explicitement de façon individuelle et constitue une unité élémentaire de connaissance. Un modèle conceptuel de données peut donc extraire d'une ontologie seulement les concepts (classes et propriétés) pertinents pour son objectif applicatif. Il peut également, sous réserve de respecter leur sémantique (par exemple, subsomption), les organiser de façon assez différente de leur organisation dans l'ontologie, la référence au concept ontologique permettant de définir avec précision la signification de l'entité référençante.
- **Consensualité**. Une ontologie de domaine représentant les concepts sous une forme consensuelle pour une communauté, l'exploitation de l'ontologie tant dans une phase de conception, par les concepteurs, qu'ultérieurement dans une phase d'exploitation, par un utilisateur, permet un accès naturel et ergonomique aux informations du domaine dès lors que concepteur et utilisateur relèvent de la communauté visée par l'ontologie. De même, l'intégration sémantique de tous les systèmes basés sur une même ontologie pourra facilement être réalisée si les références à l'ontologie sont explicitées.
- **Non canonicité des informations représentées**. A la différence des modèles conceptuels qui utilisent un langage minimal et non redondant pour décrire les informations d'un domaine, les ontologies utilisent, en plus des concepts primitifs, des concepts définis qui fournissent donc des accès alternatifs à la même information. La représentation de ces concepts non canoniques sous

forme de vue sur la base de données permet d'étendre encore l'ergonomie d'accès à l'information.

1.2.1.4 Utilisation d'ontologies conceptuelles pour l'intégration de données

Nous présentons dans cette section comment une ontologie conceptuelle est utilisée dans un système d'intégration de données. L'intégration de données à base ontologique peut être divisée en trois étapes :

1. la **représentation de sémantique de données** qui vise à interpréter le sens de chaque source en l'associant à une ontologie locale. Ce processus est effectué d'une façon manuelle ou semi-automatique. En réalité, la sémantique de chaque source peut être découverte semi-automatiquement grâce aux techniques de fouilles de données ou aux machines d'apprentissage, etc. Mais son résultat est approximatif, il nécessite donc la présence de l'administrateur pour valider ce résultat. Cette étape peut être éliminée si chaque source est *une base de données à base ontologique.*
2. l'**intégration sémantique** qui vise à intégrer les ontologies des sources. Elle consiste à établir les relations sémantiques (équivalence, subsomption) entre les concepts des ontologies. L'automatisation du processus d'intégration sémantique dépend des méthodes d'utilisation d'ontologies qui sont précisées ci-dessous.
3. l'**intégration de données** qui vise à peupler les données dans un entrepôt pour les systèmes matérialisés ou à construire des interfaces de requêtes pour les systèmes virtuels qui fournissent une vue unique sur les données. Cette étape peut être faite par des programmes génériques qui exploitent la correspondance ontologique établie dans l'étape précédente.

Nous avons identifié dans la thèse de Dung NGUYEN XUAN [217] un facteur important qui peut influencer les solutions d'intégration de données qui concerne la relation temporelle entre les ontologies locales et partagée. Ce facteur classifie les systèmes d'intégration à base ontologique en deux classes: (1) intégration sémantique *a posteriori*, et (2) intégration sémantique *a priori*.

1.2.1.4.1 Intégration sémantique *a posteriori*

L'approche d'intégration sémantique *a posteriori* est caractérisée par les principes suivants:

- les ontologies locales sont indépendantes;
- l'étape d'intégration sémantique est donc effectuée d'une *façon manuelle* ou *semi-automatique*. Elle consiste à établir la correspondance entre les concepts primitifs des ontologies.
- dans ce contexte, deux structures d'intégration d'ontologies sont possibles:
  1. la structure "**réseau**" (voir la figure 1.6-A) :
     - la correspondance entre deux ontologies locales est établie directement de l'un à l'autre. Supposons qu'il existe n ontologies locales, il faut alors créer $[n*(n-1)/2]$ correspondance.
     - l'interface utilisateur peut être créée directement à partir d'une ontologie locale quelconque.
  2. la structure "**en étoile**" (voir la figure 1.6-B):
     - la correspondance entre deux ontologies locales est établie indirectement à travers une ontologie référencée. Cette ontologie est également appelée *l'ontologie partagée* du système. Une ontologie locale n'est mise en correspondance qu'avec l'ontologie partagée. Et seuls les concepts locaux que l'on souhaite intégrer sont mis en correspondance avec

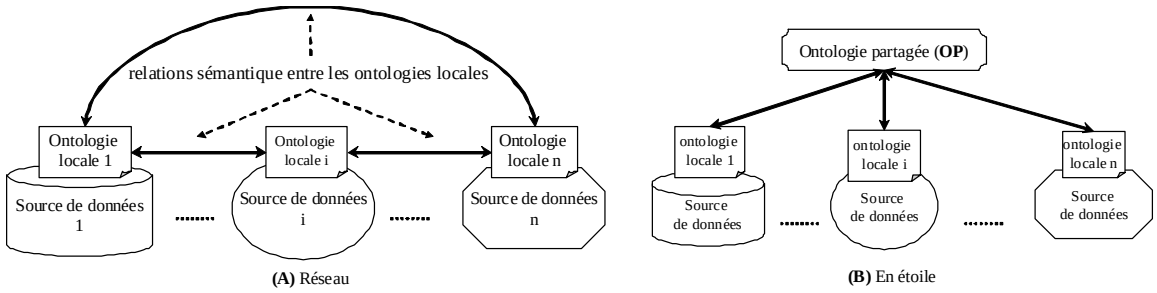


FIG. 1.6 – Utilisation d'ontologies conceptuelles dans l'approche d'intégration sémantique *a posteriori*

les concepts partagés. Supposons qu'il existe *n* ontologies locales, il faut alors créer [*n*] articulations d'ontologies.
- l'ontologie partagée est soit une ontologie spécifique au système, soit une ontologie normalisée indépendante du système.
- l'interface utilisateur est créée à partir de l'ontologie partagée.

L'avantage d'une telle approche est l'indépendance des sources de données intégrées. Par contre, l'intégration sémantique n'est pas automatique. Pour la structure "réseau", le nombre de mise en correspondance entre les ontologies est important. Pour la structure "en étoile", ce nombre est diminué; et l'interface utilisateur correspond à l'ontologie partagée.

Parmi les systèmes d'intégration qui suivent cette approche, nous pouvons citer OBSERVER [148] pour la structure "réseau", et KRAFT [211] pour la structure "en étoile".

1.2.1.4.2 Intégration sémantique *a priori* L'approche d'intégration sémantique *a priori* correspond à une problématique où l'administrateurs de chaque source souhaite faciliter l'accès à ses données par de nombreux clients, éventuellement inconnus. Elle suppose l'existence d'une ontologie de domaine. Elle est caractérisée par les principes suivants (figure 1.7) :
- Chaque source reprend *a priori* des concepts dans une ontologie de domaine préexistante pour construire son ontologie locale. Cette ontologie de domaine est considérée comme *l'ontologie globale* du système. L'ontologie locale peut être soit être un sous-ensemble de l'ontologie de domaine soit être une spécialisation c'est-à-dire un sous ensemble éventuellement étendu par des classes plus spécialisées et/ou des propriétés additionnelles; aucune de ces extensions ne durant exister dans l'ontologie de domaine. Cette intégration est dite *intégration par articulation d'ontologies* [217].
- l'intégration sémantique est donc naturellement *automatique* grâce aux relations sémantiques *a priori* entre les ontologies locales (ces relations sont celles entre les concepts dans l'ontologie globale).
- l'interface utilisateur est créée à partir de l'ontologie globale.
- un système d'intégration de données suivant cette approche n'intègre que les données dont la sémantique est présentée dans l'ontologie globale. Elle peut éventuellement, comme nous l'avons montré dans la thèse de Dung NGUYEN XUAN être étendue automatiquement par les extensions figurant dans les diverses ontologies locales.

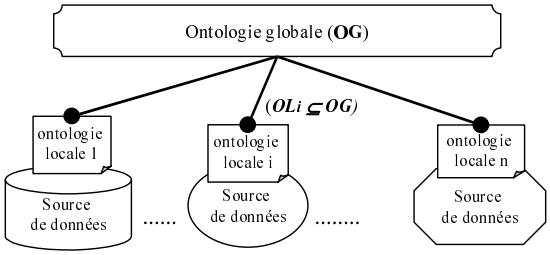


FIG. 1.7 – Utilisation d'ontologies conceptuelles dans l'approche d'intégration sémantique *a priori*

Cette approche permet d'intégrer facilement une nouvelle source dans le système si la sémantique de cette source est couverte par *OG*. Par contre, cet aspect limite les sources de données au niveau de leurs indépendances. Parmi les projets utilisant cette méthode, nous pouvons citer SIMS[6], PICSEL [92, 102], OntoBroker [79], BUSTER [210], COIN [100].

1.2.1.4.3 Domaines d'utilisation des solutions d'intégration a priori L'intégration a priori de données prépare les sources en explicitant leurs sémantiques. Cette approche a été largement motivée par l'émergence des ontologies de domaine. Elles occupent désormais une place importante dans la recherche en informatique et sont utilisées ou proposées, dans de nombreux domaines (intégration de données, e-ingénierie, Web sémantique, Web service, bases de données, etc.). Le caractère formel et consensuel d'une ontologie permet de la diffuser dans une communauté et contribue à pouvoir rendre interopérable les applications en représentant explicitement la sémantique des données.

L'utilisation croissante d'ontologies dans divers domaines fait qu'aujourd'hui, certaines données sont représentées comme des instances de classes d'ontologies. Leur sémantique est donc définie à partir des ontologies. C'est en particulier le cas (1) pour les annotations de documents sur le Web, (2) pour les objets figurant dans les catalogues de composants [179].

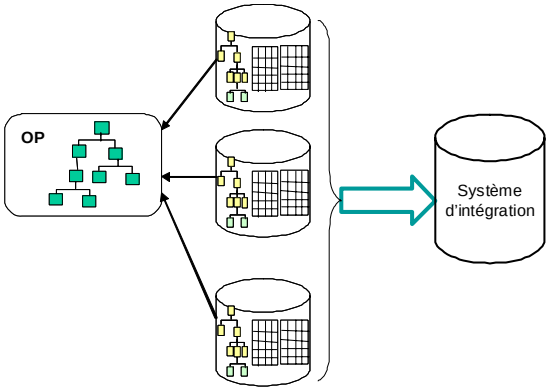


FIG. 1.8 – Utilisation d'ontologies conceptuelles dans l'approche d'intégration sémantique *a priori*

1.2.1.4.4 Discussion sur l'intégration sémantique a priori de données Une intégration a priori sémantique peut assurer l'automatisation complète du processus d'intégration si les deux conditions suivantes sont satisfaites (Figure 1.8) :

1. chaque source doit représenter explicitement la signification de ses propres données ; c'est la notion d'ontologie locale qui doit exister dans chaque source ; et

2. il doit exister une ontologie partagée de domaine, et chaque ontologie locale doit référencer explicitement l'ontologie partagée pour tous les concepts qu'elle définit et qui sont déjà définis dans l'ontologie partagée. Ceci définit la totalité des relations sémantiques existant entre les concepts des ontologies locales et partagée (articulations).

Une approche a priori d'intégration à base ontologique n'élimine pas la nécessité d'une réflexion humaine pour identifier deux conceptualisations différentes d'une même réalité. Mais, elle exige une réflexion faite a priori, lors de la mise à disposition de la source de données, et non a posteriori, pendant la phase d'intégration sémantique. Elle exige donc l'existence a priori d'une ontologie de domaine consensuelle. Les approches d'intégration sémantique a priori existant dans la littérature présentent néanmoins deux inconvénients majeurs :

1. elles interdisent de représenter des concepts qui n'existent pas dans l'ontologie partagée, or, par définition, une ontologie partagée ne définie que ce qui est partagé par toute une communauté. Or, chaque membre de cette communauté, et c'est trivialement vrai pour le commence électronique, souhaite aussi définir des concepts non partagés.

2. elle impose de calquer la structure de la base de données sur la structure de l'ontologie alors que chaque utilisateur peut souhaiter utiliser des systèmes de gestion différents (par exemple relationnel quand le modèle d'ontologie est orienté objet) ou des structures de schémas différents.

Ce sont les problèmes que nous avons résolu dans le travail de thèse de Dung NGUYEN DUNG [217]. Ce travail est associé à trois hypothèses :
- il existe une ontologie de domaine recouvrant la totalité des termes consensuels, et
- les administrateurs (fournisseur) des sources de données veulent communiquer entre eux, mais
- les besoins sont différents et ils souhaitent une grande autonomie schématique.

1.2.1.5 La gestion de l'évolution

La présence des ontologies (partagées et locales) permet une automatisation du processus d'intégration, mais rend la gestion de l'évolution plus difficile. Les fournisseurs des sources étant différents et chaque source se comportant indépendamment des autres, la relation entre le système intégré et ses sources indépendantes est faiblement couplée. Une source pouvant modifier sa structure et sa population sans en informer les autres. L'ontologie commune (partagée) peut évoluer indépendamment des sources. Ceci engendre des anomalies de maintenance. Dans un tel environnement (où les changements sont asynchrones), gérer l'évolution consiste à :

1. au niveau du contenu, reconnaître un composant même s'il est décrit par des propriétés un peu différentes (des informations locales peuvent lui être attachées), et se souvenir (éventuellement) à quelles périodes un composant était disponible.

2. au niveau de l'ontologie, permettre une évolution asynchrone des différentes ontologies tout en conservant les relations inter-ontologies.

1.2.2 Conception aval d'un système d'intégration

Une fois le système d'intégration construit, il doit être exploité par des utilisateurs ou des décideurs quelque soit son architecture : médiateur, matérialisée, pair à pair, etc. Lorsqu'il est implémenté dans une architecture de médiation, les données ne sont pas stockées au niveau du médiateur. Elles restent dans les sources de données et ne sont accessibles qu'à ce niveau. L'intégration d'information est fondée sur l'exploitation de vues abstraites décrivant de façon homogène et uniforme le contenu des sources dans les termes du médiateur. Les sources pertinentes, pour répondre à une requête, sont calculées en fonction de la définition de ces vues. Le problème consiste à trouver une requête équivalente à la requête de l'utilisateur. Les réponses à la requête posée sont ensuite obtenues en évaluant cette requête sur les extensions des vues. De nombreux travaux, dans le but de faciliter l'exploitation d'un système d'intégration virtuel, ont été proposés. Des études ont porté sur les langages pour modéliser le schéma global du système d'intégration, pour représenter les vues sur les sources à intégrer et pour exprimer les requêtes provenant des utilisateurs humains ou d'entités informatiques. D'autres travaux ont porté sur la conception et la mise en oeuvre d'algorithmes de réécriture de requêtes en termes de vues décrivant les sources de données pertinentes. D'autres encore sur la conception d'interfaces intelligentes assistant l'utilisateur dans la formulation de requêtes, l'aidant à affiner une requête en cas d'absence de réponses ou en cas de réponses beaucoup trop nombreuses en personnalisant les requêtes.

Dans le cas où le système d'intégration est implémenté dans une architecture matérialisée, son exploitation est similaire à celle d'une base de données centralisée développée dans le cadre des applications OLTP (Online Transaction Processing) ou d'un entrepôt de données décisionnel développé pour les applications OLAP (Online Analytical Processing).

Depuis 1996, nous travaillons sur le problème d'optimisation de requêtes. Nos premiers travaux ont concerné la fragmentation horizontale et verticale dans les bases de données orientées objet. Nous avons proposé des nouveaux types d'algorithmes de fragmentation. Nous avons entamé ces travaux au TIMC-IMAG et approfondi à Hong Kong University of Science and Technology. Avec l'arrivée des entrepôts de données décisionnels, nous nous sommes concentrés sur leur conception physique. Rappelons que dans les bases de données traditionnelles, la tâche d'un administrateur était principalement concentrée sur la gestion des utilisateurs et d'un nombre restreint de structures d'optimisation comme les index mono table ou les différentes implémentations de l'opération de jointure (boucles imbriquées, tri fusion, hash, etc.). Dans les applications décisionnelles, la conception physique est devenue un enjeu important, comme l'indique Chaudhuri dans son papier intitulé *Self-Tuning Database Systems: A Decade of Progress* qui a eu le prix de 10 Year Best Paper Award à la conférence VLDB'2007 : "*The first generation of relational execution engines were relatively simple, targeted at OLTP, making index selection less of a problem. The importance of physical design was amplified as query optimizers became sophisticated to cope with complex decision support queries.*". Cette amplification est due aux caractéristiques suivantes liées aux entrepôts de données : (1) le volume de données, (2) la complexité des requêtes et les (3) les exigences des décideurs sur le temps de réponse de requêtes. Pour offrir une meilleure utilisation d'un entrepôt de données, la conception physique est devenue un enjeu important [69].

Durant la phase d'exploitation d'un système d'intégration ; l'administrateur doit effectuer quatre tâches principales : (1) choix des structures d'optimisation, (2) choix de leur mode de sélection, (3) développement des algorithmes de sélection et (4) validation et déploiement des solutions d'optimisa-

tion.

1.2.2.1 Choix de structures d'optimisation

Une large panoplie des structures d'optimisation a été proposée et la plupart supportée par les systèmes de gestion de bases de données commerciaux (SGBD). Nous pouvons ainsi citer les *vues matérialisées*, les *index avancés*, la *fragmentation*, l'*allocation*, le *traitement parallèle*, le *groupement*, la *compression*, etc. Certaines structures sont l'héritage des bases de données traditionnelles; comme la fragmentation, l'allocation, certains types d'index, le regroupement, etc. Notons que chaque structure a ses avantages, ses limites et certaines ont de fortes similarités. Afin de mieux les cerner, nous les avons classées en deux principales catégories [20, 37] : les *structures redondantes* et les *structures non redondantes*. Les structures redondantes optimisent les requêtes, mais exigent des coûts de stockage et de maintenance. Les vues matérialisées [108], les index [110], la fragmentation verticale[2] [158, 191] sont trois principaux exemples de cette catégorie. Les structures non redondantes ne nécessitent ni coût de stockage, ni coût de maintenance. Deux exemples de cette catégorie sont la fragmentation horizontale [191, 17] et le traitement parallèle (dans le cas où l'allocation est faite sans réplication) [200]. La figure 1.10 illustre cette classification.

1.2.2.2 Modes de sélection des structures d'optimisation

La sélection et la gestion des structures d'optimisation sont au coeur de la conception physique. La sélection d'une structure d'optimisation donne un schéma contenant une ou plusieurs instances de cette structure. Pour satisfaire, sa charge de requêtes, l'administrateur peut choisir une ou plusieurs structures. Dans le cas, où il choisit une seule structure, sa sélection est appelée une sélection isolée. Dans le cas, où il choisit plusieurs structures, leur sélection est appelée sélection multiple.

1.2.2.2.1 La sélection isolée des schémas d'optimisation Cette sélection a connu beaucoup d'intérêt de la part des communautés de bases et entrepôts de données [4, 26, 33, 109, 67, 128, 71, 101, 165, 158, 94]. Elle peut concerner une structure redondante ou non redondante. La plupart des problèmes de sélection d'un schéma d'optimisation ont été reconnus comme des problèmes NP-Complet. Dans les bases de données traditionnelles, cette sélection a souvent concerné les index mono attributs [67], la fragmentation horizontale et verticale [158, 26, 33], l'allocation [5, 147], etc.. Dans le contexte des entrepôts de données, en plus des structures traditionnelles, elle concerne les vues matérialisées [109], les index binaires et multi tables [71], la compression des index binaires, etc.

La sélection isolée n'est pas toujours suffisante pour optimiser toute la charge de requêtes du fait que chaque structure d'optimisation est bien adaptée pour un profil particulier de requêtes, d'où le recours à la sélection multiple [200, 20]).

1.2.2.2.2 La sélection multiple des schémas d'optimisation Comme la sélection isolée, elle peut concerner des structures redondantes, non redondantes ou les deux. Elle est plus complexe que la sélec-

[2]Dans la fragmentation verticale, la clé de la table fragmentée est dupliquée sur tous les fragments. Pour cela, elle est considérée comme une structure redondante

tion isolée vue la complexité de son espace de recherche englobant ceux des structures concernées. En plus de cette complexité, une autre s'ajoute qui concerne la gestion des similarités (ou dépendances) entre certaines structures. Prenons l'exemple de deux structures d'optimisation qui sont les vues matérialisées et des index. Elles génèrent deux schémas d'optimisation redondants, partageant la même ressource qui est l'espace de stockage et nécessitent des mises à jour régulières. Une vue matérialisée relationnelle peut être indexée afin d'augmenter ses performance et vice versa. En conséquence, la présence d'un index peut rendre une vue matérialisée plus avantageuse. Cette similarité pourrait influencer la sélection de leurs schémas [192, 225].

Le degré de dépendance entre les structures d'optimisation n'est pas toujours le même et dépend principalement de la nature de ces dernières. Deux degrés de dépendance ont été définis [225]: *dépendance faible* et *dépendance forte*.

Définition 1

Une structure d'optimisation SO_1 dépend fortement d'une autre structure d'optimisation SO_2 si un changement dans la sélection de SO_2 se traduit souvent par un changement de celle de SO_1. Dans les autres cas, SO_1 dépend faiblement de SO_2 [225].

Définition 2

Une structure d'optimisation SO_1 dépend faiblement d'une autre structure d'optimisation SO_2 si un changement dans la sélection de SO_2 n'affecte pas un changement de celle de SO_1 [225].

La relation de dépendance n'est pas symétrique. A partir de cette définition et étant donné que la relation de dépendance est binaire, nous pouvons énumérer trois combinaisons de dépendances: *Forte-Forte (ou mutuellement forte*), *Forte-Faible* et *Faible-Faible (ou mutuellement faible)* que la sélection multiple doit prendre en considération. Le tableau 1.1 montre des dépendances entre les vues matérialisées, les index (définis sur une seule table) et la fragmentation horizontale identifiées par [225]. D'autres dépendances

SO_1/SO_2	Index	Vue matérialisée	Fragmentation horizontale
Index	–	Forte	Faible
Vue matérialisée	Forte	–	Faible
Fragmentation horizontale	Faible	Forte	–

TAB. 1.1 – Dépendances entre techniques d'optimisation

existent et qui n'étaient pas identifiées par les travaux du groupe de recherche d'IBM [225].

Pour utiliser la sélection multiple dans la pratique, trois implémentations sont offertes aux administrateurs (concepteurs) [56]: (a) *une implémentation indépendante*, (b) une *implémentation conjointe* et (c) une *implémentation séquentielle* que nous détaillons dans les sections suivantes.

1.2.2.2.3 Implémentation indépendante Dans cette implémentation, les schémas de structures d'optimisation choisies par l'administrateur sont sélectionnés indépendamment, aucun ordre n'est imposé. Elle est intéressante si les structures d'optimisation concernées ne sont pas dépendantes [225], ce qui est rare dans le contexte des bases et entrepôts de données, où un nombre important de structures présente des dépendances.

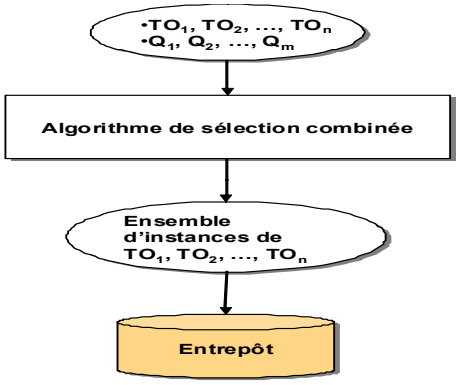


FIG. 1.9 – Approche de sélection conjointe

1.2.2.2.4 Implémentation conjointe Si le concepteur opte pour cette implémentation, les schémas de structures d'optimisation sont sélectionnés simultanément. Elle n'est possible que pour les structures ayant une forte similarité (dépendances fortes et mutuelles) et partageant la même contrainte [225]. Dans cette implémentation, un seul algorithme parcourant l'espace global de recherche correspondant aux structures considérées est utilisé. Son avantage est sa prise en compte des interdépendances entre les structures d'optimisation utilisées. Son inconvénient majeur est sa complexité, car l'espace de recherche augmente de manière combinatoire en fonction du nombre important des combinaisons possibles [225, 206]. Ainsi, les problèmes de sélection de chaque structure sont connus comme des problèmes NP-Complet et leur combinaison ne fait qu'augmenter la complexité du problème. Un autre inconvénient concerne son extensibilité. Pour ajouter une nouvelle structure d'optimisation ayant des similarités entre les structures existantes, une modification de l'algorithme global et peut-être les structures de données représentant le problème combiné sont envisagés [225]. [206, 32] sont deux exemples de travaux optant pour cette implémentation pour sélectionner les vues matérialisées et les index. Microsoft Tuning Wizard utilise cette implémentation pour recommander des index et les vues matérialisées aux utilisateurs [192]. Figure 1.9 illustre cette sélection.

1.2.2.2.5 Implémentation séquentielle Cette implémentation est une alternative pour réduire la complexité de l'implémentation conjointe. Elle prend en considération les dépendances entre les structures concernées. Les schémas de structures d'optimisation choisies par l'administrateur sont sélectionnés l'un après l'autre en respectant l'ordre d'application des algorithmes de sélection. Cette implémentation peut être vue comme une *succession de sélection isolée*. Le problème de la conception physique $\mathbb{PSP}$ dans cette implémentation est alors décomposé en plusieurs sous-problèmes traités dans un ordre bien établi. Une des entrées du sous problème i ($i > 1$) sont les sorties du sous problème $(i-1)$. Cette sélection est souhaitable lorsque la dépendance entre les deux structures d'optimisation SO_1 et SO_2 n'est pas mutuellement forte [171, 191, 200]. En d'autres termes, si une structure d'optimisation SO_1 dépend fortement de SO_2, mais SO_2 dépend faiblement de SO_1, alors cette implémentation est appliquée; en respectant l'ordre de sélection de schémas de deux structures: SO_1 doit être sélectionnée après la sélection de SO_2.

Cette implémentation est facile à mettre en oeuvre, chacun de ses problèmes peut être résolu indépendamment des autres [225], ce qui facilite l'ajout de nouveaux modules implémentant de nouvelles structures d'optimisation. [200] est un exemple de cette implémentation où la fragmentation, l'indexation

et l'allocation sont établis d'une manière séquentielle.

1.2.2.3 Développement des algorithmes de sélection

Le problème de sélection de structures d'optimisation est complexe, d'où la nécessité de développer des algorithmes offrant des solutions quasi optimales, en allant des algorithmes simples comme les hill climbing aux algorithmes complexes comme les génétiques et le recuit simulé. Les algorithmes proposés doivent prendre en compte la ou les structures d'optimisation utilisées et leur mode de sélection.

1.2.2.4 Validation de schémas d'optimisation

Afin de quantifier la qualité des solutions proposées par chaque algorithme, la présence d'une métrique est nécessaire. Deux types de métrique sont possibles : (i) mathématique et (ii) validation réelle sur le système de base de données cible implémentant le système d'intégration. La validation réelle est difficile à mettre en place vue sa complexité, car elle exige que le système d'information soit opérationnel. Etant donné que nous proposons une démarche de développement d'un système d'intégration, il est primordial d'avoir des modèles de coût mathématiques simulant l'exécution de la charge de requêtes sur le SGBD cible supportant le système d'intégration. Un modèle de coût estime le coût d'exécution d'une charge de requêtes. Cette estimation peut concerner le nombre d'entrées sorties nécessaires pour exécuter cette charge, ou le temps CPU, etc. Notons que les optimiseurs de requêtes (Oracle, SQL Server, etc.) proposent des optimisations basées sur des modèles de coût (Cost-based Optimisation). Dans le cas, où le concepteur est satisfait de ses structures d'optimisation, il peut générer des scripts correspondant aux schémas d'optimisation qui peuvent attaquer le SGBD cible implémentant le système d'intégration.

La Figure 1.10 résume notre vision sur les différentes étapes de la phase d'exploitation d'un système d'intégration matérialisé.

1.2.2.5 Vers des outils d'assistance et simulation de la conception physique

Comment nous sommes arrivés à la simulation?

Durant la thèse de Kamel BOUKHALFA, nous avons étudié plusieurs structures d'optimisation, pour chacune d'elle un nombre important d'algorithmes a été proposé. Certains algorithmes de sélection comme le génétique et le recuit simulé ont besoin d'un réglage de leurs paramètres (taux de mutation, taux de croisement, la température, etc.), etc.

En 2006, nous (Kamel BOUKHALFA et moi) avons proposé un sujet de Master 2 au laboratoire LISI. Son but était d'implémenter des algorithmes basés sur les techniques de fouille de données pour la sélection des index de jointure binaire pour optimiser les requêtes OLAP. A la fin de leur stage, les étudiants ont proposé une interface intéressante permettant aux utilisateurs (concepteurs) de choisir les tables à index et les différents paramètres liés à l'algorithme utilisé. Cette interface est à l'origine de la proposition d'un outil graphique assistant les concepteurs durant leurs tâches (Figure 1.11) au niveau du : (1) choix des structures d'optimisation, (2) de la nature de sélection (isolée ou multiple), (3) des algorithmes de sélection utilisés, (4) des paramètres des algorithmes, (5) des tables et des attributs concernés par les structures de sélection (le cas des index et de la fragmentation), etc. Une autre motivation de ce

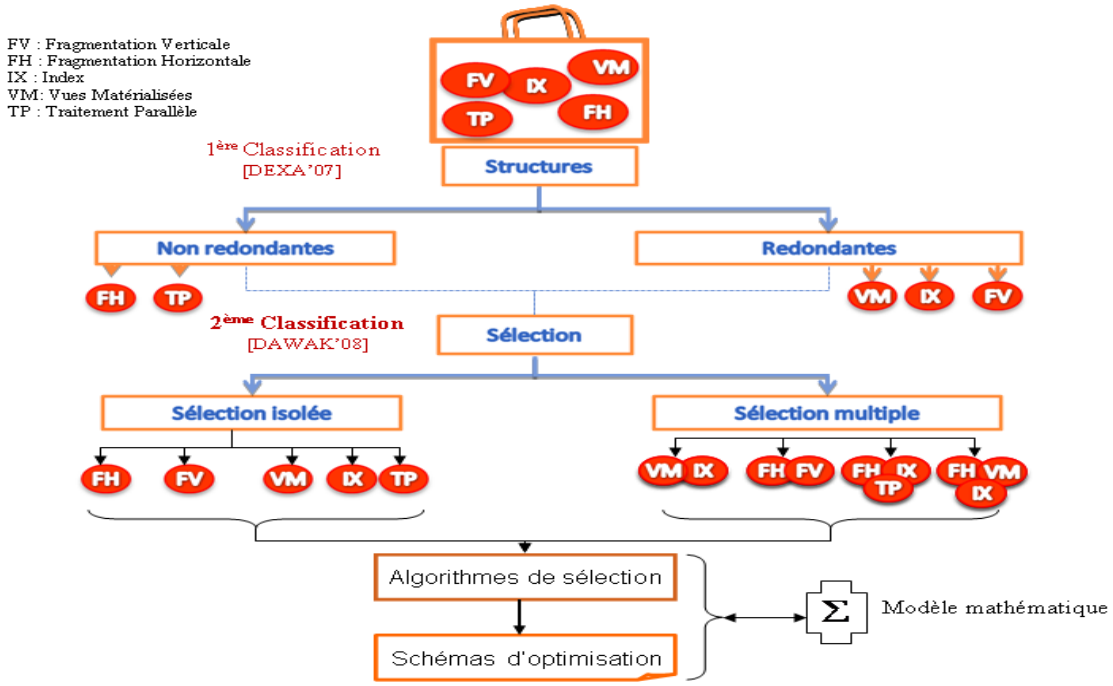


FIG. 1.10 – Les différentes phases de l'étapes d'exploitation

travail était l'étude effectuée par Fabio et al. [164] sur le temps passé par l'administrateur pour assurer les tâches d'administration. La figure 1.12 illustre cette étude et montre que la tâche de conception physique et de tuning consomme 17% du temps d'administration.

Notre outil d'administration a été présenté lors des 5èmes journées francophones sur les entrepôts de données et analyse en ligne à Toulouse. Prof. Teisseire MAGUELONNE du laboratoire LIRMM de Montpellier, nous a suggéré de l'utiliser comme un simulateur de la conception physique. La notion de simulateur suit parfaitement notre vision concernant la proposition d'une démarche de développement d'un système d'intégration. La simulation peut être utilisée durant la phase d'exploitation de système d'intégration en offrant aux concepteurs une plateforme de tests de différentes structures d'optimisation et leur paramétrage afin de satisfaire une charge de requêtes.

1.2.2.6 Personnalisation de système d'intégration de données

Ces dernières années, on assiste à une forte inflation du volume d'information en raison de l'accroissement des capacités de calcul et de stockage. Dans ce contexte de surcharge d'informations, il y a lieu de développer des outils informatiques pour faciliter la recherche et l'extraction rapide des informations pertinentes et les rendre disponibles et exploitables par un utilisateur. Un de ces types d'outils est la personnalisation qui consiste à répondre aux mieux aux attentes de chaque utilisateur. La personnalisation a été largement étudiée dans plusieurs domaines : recherche d'information, interface homme machine, bases de données, entrepôts de données. Dans le cas de la recherche d'information (RI) sur le Web, un utilisateur interroge un moteur de recherche à travers une liste de mots clés pour recevoir comme réponse un ensemble de pages Web. Dans le cas des bases de données, un utilisateur accède à l'instance de base

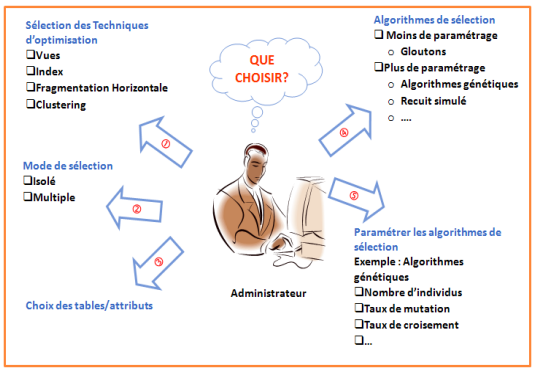


FIG. 1.11 – Tâches liées à l'administration

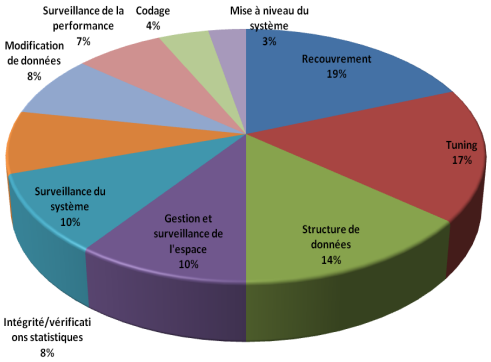


FIG. 1.12 – Répartition de l'effort de l'administrateur

de données à l'aide d'une requête dont la réponse est composée d'un ensemble de tuples. Notons que dans le cas des bases de données et contrairement au cas de la recherche d'information par exemple, la requête d'extraction de données peut être caractérisée par deux parties : (a) la partie de sélection pour ne retenir que les données recherchées et (b) la partie de mise en forme spécifiant comment seront affichées ces données. La mise en forme, en SQL par exemple, est spécifiée par l'ordre des attributs dans la clause SELECT.

Les utilisateurs d'un système d'intégration cherchent à retrouver des informations pertinentes à leur problème. L'extraction de telles informations pose un certain nombre de problèmes. Le premier est le temps de réponse élevé. Ce problème est d'autant plus crucial que l'utilisateur ne sachant pas a priori ce qu'il cherche, pose plusieurs requêtes pour trouver ce qui l'intéresse. Le deuxième est la taille importante des réponses aux requêtes utilisateur. Le nombre d'informations trouvées est généralement très important. L'exploitation de ces réponses n'est pas toujours possible. On peut se retrouver dans des situations, où des réponses ne peuvent pas être toutes explorées pour être utilisées. C'est pourquoi on fait appel aux outils de personnalisation ou de recommandation pour trouver rapidement un ensemble réduit de réponses intéressantes et pouvant satisfaire l'utilisateur [153]. Les deux processus de recommandation et de personnalisation de l'information sont souvent fortement liés. Cependant, ils diffèrent sur le contenu de la réponse :

- la personnalisation permet de réduire le contenu du résultat qui consiste en un sous ensemble de la réponse à la requête initiale posée par l'utilisateur,
- la recommandation permet de compléter le résultat personnalisé. Elle consiste à enrichir le contenu du résultat avec de nouvelles informations qui ne sont pas demandées par la requête initiale de l'utilisateur. Ainsi, elle garantit une réponse non vide qui est possible dans la personnalisation.

Dans la thèse de Hassina MOULOUDI dirigée par Prof. Arnaud GIACOMETTI et Patric MARCEL du laboratoire d'Informatique de Tours, nous nous sommes intéressés à la personnalisation de l'information dans le contexte de l'interrogation de bases de données par requêtes OLAP.

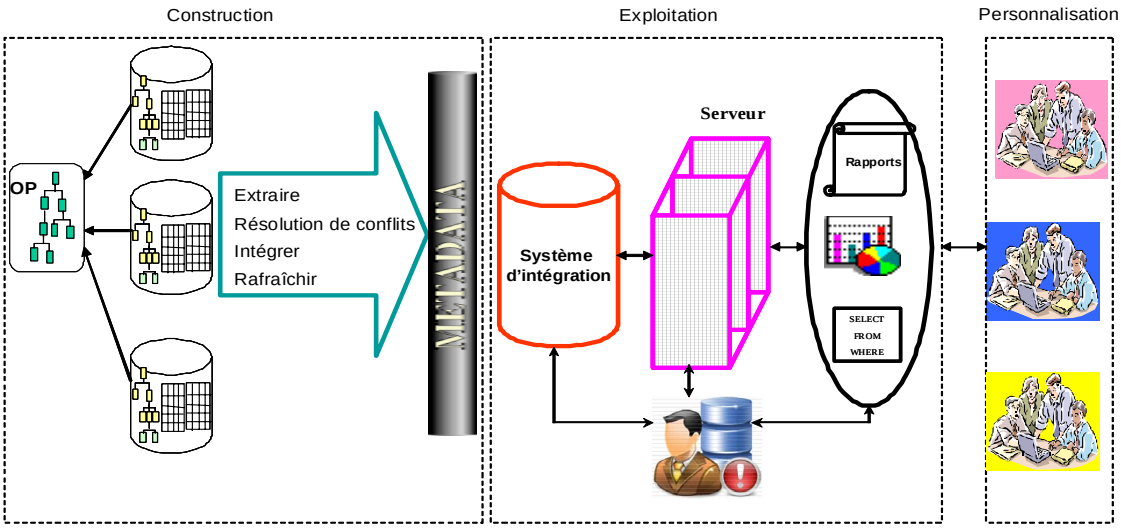


FIG. 1.13 – La proposition de notre démarche de développement d'un système d'intégration

1.3 Notre démarche de développement de système d'intégration

La figure 1.13 illustre l'architecture de notre démarche de développement de système d'intégration. Elle comporte trois phases principales: (a) la construction du système d'intégration, (b) son exploitation et (c) sa personnalisation.

Chronologiquement, nous avons commencé la phase en aval de conception de systèmes d'intégration, où nous nous sommes intéressés à la sélection des structures d'optimisation dans le contexte des bases de données traditionnelles et les entrepôts de données décisionnels. A notre arrivée au laboratoire LISI, nous nous sommes intéressés à la dimension intégration de sources de données hétérogènes par articulation d'ontologies (les deux thèses de Dung NGUYEN XUAN et Hondjack DEHAINSALA). Nous avons également continu à proposer des solutions pour la phase d'exploitation (la thèse de Kamel BOUKHALFA). En 2004, nous nous sommes intéressés à la problématique de la personnalisation dans les entrepôts de données avec les collègues du laboratoire d'informatique de Tours. Mon point d'entrée était l'exploitation des profils des utilisateurs pour gérer des structures d'optimisation. Dans le cadre de la thèse de Dilek TAPUCU encadrée par Yamine AIT AMEUR et effectuée au sein du laboratoire LISI, nous nous sommes intéressés au stockage et manipulation des profils des utilisateurs au sein d'une base de données à base ontologique.

1.4 Contributions et organisation de ce mémoire

Nos contributions concernent chaque phase de notre démarche de développement d'un système d'intégration, comme le montre la Figure 1.14.

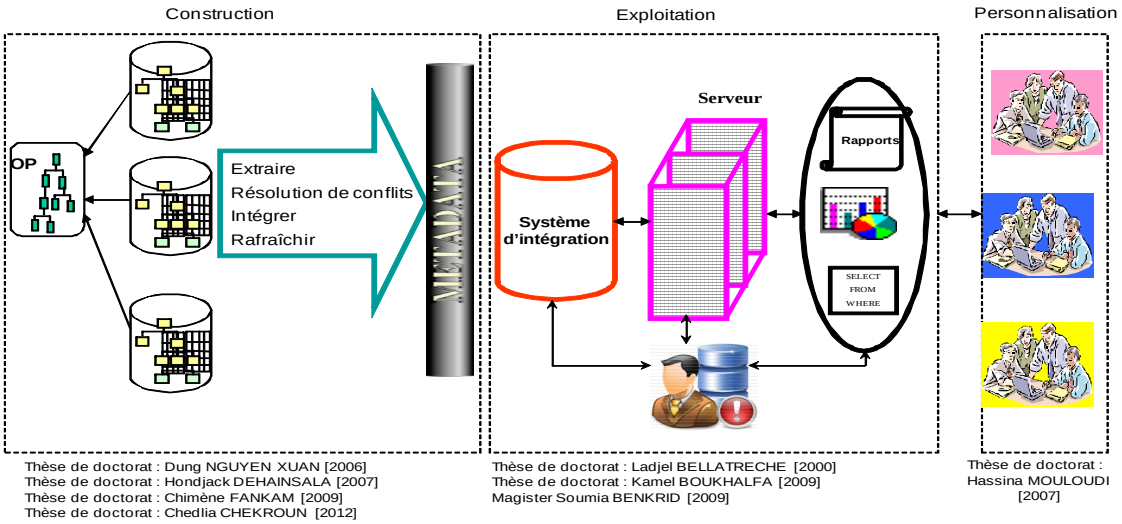


FIG. 1.14 – Nos contributions

1.4.1 Phase en amont

En ce qui concerne la phase en amont, nous avons proposé des solutions d'intégration de bases de données à base ontologique (BDBO). Une BDBO présente deux caractéristiques : (1) ontologie et données sont toutes deux représentées dans la base de données et peuvent faire l'objet des mêmes traitements (insertion, mise à jour, requêtes, etc.); (2) toute donnée est associée à un élément ontologique qui en définit le sens. Dans la thèse d'Hondjack DEHAINSALA [80] encadrée conjointement par Prof. Guy PIERRA et moi même au sien du LISI, nous avons proposé une architecture d'implémentation des BDBOs et une validation. Une formalisation du problème d'intégration par articulation a priori d'ontologies est décrite avec des scénarii d'intégration. Ces derniers sont validés sur des catalogues industriels dans le cadre d'un projet avec l'entreprise automobiles Renault.

Dans le cadre de travaux de thèse de Dung NGYUEN XUAN [217] encadrée conjointement par Prof. Guy PIERRA et moi même au sien du LISI, nous avons proposé une approche automatique d'intégration par articulation a priori d'ontologies dans un entrepôt de données. Plusieurs scénarii d'intégration ont été proposés et validée sur des ontologies réelles. Nous avons également proposé des solutions pour gérer l'évolution asynchrone de l'ensemble de sources, tout en maintenant la possibilité d'intégration automatique de ces dernières au sein d'un entrepôt de données. Deux problèmes ont été résolus: (i) la gestion des évolutions des ontologies afin de maintenir la cohérence des relations entre ontologies, et entre ontologies et données, et (ii) la gestion de cycle de vie des instances, c'est-à-dire la représentation de la connaissance des instances qui existaient dans l'entrepôt à un instant donné.

1.4.2 Phase en aval

La troisième contribution propose des solutions pour l'étape d'exploitation d'un système d'intégration matérialisée. Ce dernier est modélisé par un schéma en étoile relationnel caractérisé par une table des faits volumineuse et plusieurs tables de dimension de taille inférieure. Ces travaux ont été effectués dans le cadre de ma thèse et celle de Kamel BOUKHALFA, effectuée au sein de l'équipe ingénierie de

données du laboratoire LISI. Nous avons étudié plusieurs structures d'optimisation redondantes (vues matérialisées, les index, la fragmentation verticale) et non redondantes (la fragmentation horizontale et le traitement parallèle) et que j'ai dirigé de façon autonome. Certaines sont l'héritage des bases de données traditionnelles, comme la fragmentation horizontale. Contrairement aux structures d'optimisation sélectionnées après la création de l'entrepôt de données (ou la base de données), la décision d'utiliser la fragmentation horizontale doit se faire avant sa création. Cette situation rend l'étude du problème de la fragmentation plus sensible.
Nous avons remarqué que la plupart des travaux sur la sélection des structures d'optimisation traite chaque structure d'une manière indépendante. Cette façon de faire, ignore les similarités entre les structures. Pour exploiter les similarités et avoir une vue globale sur le processus de sélection des structures d'optimisation, nous avons donc proposé une formalisation globale du problème de l'optimisation physique de l'entrepôt de données. Selon les besoins d'exploitation, l'administrateur peut choisir une ou plusieurs structures pour répondre à sa charge de requêtes. Dans le cas, où plusieurs structures sont choisies, nous avons identifié les différents scénarii de la résolution le problème global, où chacun est évalué. Tous les algorithmes proposés pour sélectionner les structures d'optimisation étudiées ont été validés par un modèle de coût mathématique que nous avons développé, estimant le coût d'exécution d'une charge de requêtes et par une validation réelle en utilisant le SGBD Oracle.

Le travail effectué lors de la phase d'exploitation nous a permis d'identifier plusieurs difficultés liées aux tâches d'un administrateur d'un système d'intégration. Ce dernier doit faire face à plusieurs choix et chaque choix peut avoir une incidence forte sur la performance du système : (i) choix des structures, (ii) choix du mode de sélection des schémas d'optimisation, (iii) choix des tables et des attributs concernés par la structure d'optimisation, etc. En plus de ces choix, il doit quantifier l'intérêt de chaque structure, prendre en considération leurs feedbacks pour raffiner les solutions proposées. Dans le souci d'assister les administrateurs, nous avons suggéré le développement des outils de simulation de la phase physique des systèmes d'intégration.

1.4.3 La personnalisation de système d'intégration

Cette contribution est la plus récente. Elle a débuté en 2004. Pour la personnalisation, nous avons proposé une approche de sélection de structures d'optimisation en se basant sur les profils d'utilisateur. Notre travail sur la personnalisation est considéré comme le premier, à notre connaissance, dans le contexte des entrepôts de données décisionnels. Dans le cadre de la thèse de Hassina MOULOUDI, nous avons proposé une extension des travaux sur la personnalisation développés dans les bases de données traditionnelles, en tenant compte des caractéristiques des entrepôts de données : (a) Le stockage du gros volume de données : organisation complexe du gros volume de données sous forme de cubes et (b) le langage de manipulation : opérateurs de manipulation spécifiques permettant la mise en forme. Notre approche intègre ces caractéristiques comme suit : (i) personnalisation du contenu : elle permet d'accéder rapidement à une partie du gros volume de données pouvant intéresser un utilisateur. Pour cela, notre approche permet la transformation de la requête utilisateur sans accès à toutes les données stockées du cube. Nous avons proposé un algorithme qui utilise les dimensions qui constituent la partie du cube stockée en mémoire centrale et donc n'a pas d'accès à la table des faits qui représente la partie volumineuse du cube. (ii) Personnalisation de la mise en forme : elle permet de mieux présenter l'information à l'utilisateur. Pour cela, notre algorithme permet de calculer la mise en forme de la réponse à la requête

utilisateur qui est adaptée à ses attentes et en fonction de la capacité de son dispositif d'affichage.

Ce mémoire est organisé en quatre parties.

La première partie décrit la première étape de notre démarche de conception d'un système d'intégration qui est la construction.

Le chapitre 2 décrit le modèle d'architecture de bases de données à base ontologique : le modèle OntoDB. Nous présentons, dans un premier temps le modèle d'ontologie PLIB considéré comme le noyau de nos travaux d'intégration. Contrairement aux modèles d'ontologies classiques comme OWL et F-logic qui sont des modèles orientés inférences, PLIB est orienté intégration de données et caractérisation et modularité. Dans le deuxième temps, nous présentons les objectifs et les fonctionnalités que nous nous sommes fixés pour ce modèle, puis le cadre d'hypothèses que nous avons défini. Nous décrivons chacune des quatre parties qui composent notre architecture OntoDB : méta-base, données, méta-schéma et ontologie. Une formalisation des bases de données à base ontologique est proposée. Une validation de cette proposition est également décrite.

Le chapitre 3 présente notre approche d'intégration de données qui nous permet d'intégrer de façon automatique les bases de données à base ontologique ayant des articulations sémantiques a priori avec une ontologie normalisée. Trois scénarii d'intégration sont proposés : FragmentOnto où on suppose que l'ontologie locale de chaque source est un sous ensemble de l'ontologie partagée; (2)*ProjOnto* où chaque source définit sa propre ontologie en référençant "autant que cela est possible" l'ontologie partagée, mais où les instances de chaque source sont exportées comme des instances de l'ontologie partagée; (3)*ExtendOnto* où chaque ontologie locale est définie comme dans le scénario *ProjOnto*, mais où l'on souhaite enrichir automatiquement l'ontologie partagée; toutes les instances de données peuvent alors être intégrées, sans aucune modification, au sein du système intégré. Pour chaque scénario, nous décrivons d'abord son contexte appliqué, puis son algorithme d'intégration et enfin ses applications réelles possibles.

L'objet du chapitre 4 est de présenter une approche permettant l'évolution asynchrone de l'ensemble de sources, tout en maintenant la possibilité d'intégration automatique de ces dernières au sein d'un entrepôt. Nous introduisons d'abord quelques travaux antérieurs concernant l'évolution des données et des ontologies. Nous proposons ensuite un modèle pour la gestion des évolutions ontologiques, appelé *modèle des versions flottantes*. L'hypothèse fondamentale autour de ce modèle, appelée *le principe de continuité ontologique*, stipule qu'une évolution d'une ontologie ne peut infirmer un axiome antérieurement vrai. Ce principe a pour effet de simplifier considérablement la gestion de l'évolution des ontologies. Une méthode de gestion du cycle de vie des instances qui évite toute duplication de données est ensuite présentée. Le système d'intégration résultant, validé par plusieurs implémentations, présente alors la double originalité de permettre l'autonomie et l'évolution asynchrone des sources, tout en assurant une intégration automatique de leurs contenus.

La deuxième partie de ce mémoire est dédiée à l'exploitation et personnalisation du système d'intégration.

Le chapitre 5 présente une formalisation du problème de sélection de structures d'optimisation dans le cadre des entrepôts de données décisionnels, modélisés par un schéma en étoile. Il présente deux exemples pour la sélection isolée : la sélection de schéma de fragmentation horizontale et la sélection des index de jointure. Pour chaque sélection, une formalisation et une étude de complexité de son problème

sont décrites. Finalement, deux validations, une théorique et autre réelle sous le SGBD Oracle sont proposées.

Le chapitre 6 présente une étude de similarité entre la fragmentation horizontale dérivée et les index de jointure binaires. A l'issue de cette étude, il propose une approche permettant de combiner les deux structures. Sa principale caractéristique est qu'elle utilise la fragmentation horizontale pour réduire la complexité du problème de la sélection des index de jointure. Cette approche peut être utilisée pour tuner les entrepôts de données.

Le chapitre 7 est consacré à deux parties : (1) développement d'u un outil assistant les administrateurs dans leurs tâches de conception physique et de tuning. Les principales fonctionnalités de cet outil sont présentées à travers une utilisation sur un entrepôt de données issu du banc d'essai APB1 [75]. (2) La personnalisation du système d'intégration qui consiste à exploiter les profils des utilisateurs afin de réduire la quantité des informations manipulées.

La troisième partie présente les conclusions générales de ce travail et esquisse diverses perspectives.

La quatrième partie présente notre curriculum vitea.

Première partie

Phase en amont de conception

Chapitre 2

L'intégration automatique est-elle possible ? Explicitation de la sémantique dans les sources de données

Publications : [180, 126, 82, 81].

Dans le chapitre précédent, nous avons présenté notre vision d'intégration a priori de sources de données, où nous avons souligné que pour permettre une intégration automatique, les sources doivent être préparées en explicitant la sémantique de ses données. Une source ainsi préparée est appelée *source de données à base ontologique*, lorsqu'il s'agit d'une base de données, on parle de base de données à base ontologique (BDBO). Cette explicitation suppose l'existence d'une ontologie, ce qui est le cas dans plusieurs domaines comme le montre le portail swoogle et les bases de données techniques [74, 118, 155, 160]. Cette émergence des ontologies a fait naître un besoin de gestion de volumes importants de données à base ontologique. Un nombre croissant de projets académiques ont été proposés pour répondre à ce besoin : Sesame [62], RDFSuite [3], Jena [52, 215, 63], DLDB [170], RStar [141], KAON [60], 3Store [113] et PARKA [199]. Récemment, les éditeurs commerciaux de systèmes de gestion de bases de données comme Oracle (dans les versions 10g et 11g) et DB2 ont proposé des solutions pour incorporer et gérer des données représentées en RDF éventuellement décrites par des ontologies OWL dans leurs systèmes afin de permettre à leurs clients de bénéficier d'une plateforme de gestion de données sémantiques.

Dans la thèse d'Hondjack DEHAISALA présentée au sein du laboratoire LISI, encadrée conjointement par Pr. Guy PIERRA et moi et soutenue en 2007, nous avons proposé un modèle et une architecture nommées OntoDB pour supporter les bases de données à base ontologique [80]. Avant de la décrire, nous introduisons quelques concepts et définitions.

2.1 Ontologie conceptuelle

Deux types d'ontologies sont habituellement distingués. Les ontologies linguistiques (OL) dont le but est de représenter le sens des mots utilisés dans un univers de discours particulier et les ontologies conceptuelles (OC) dont le but est de représenter les catégories et propriétés d'objet d'un domaine particulier. Dans ce chapitre nous nous intéressons principalement aux OC dont nous commençons par indiquer les définitions qui en ont été données [179, 127].

2.1.1 Définitions

L'origine de la notion d'ontologie se trouve dans une branche de la philosophie, initiée par Aristote, traitant de la science de l'être : "Partie de la métaphysique qui s'applique à l'être en tant qu'être indépendamment de ses déterminations particulières". En informatique, cette notion semble être apparue, pour la première fois, dans la communauté Intelligence Artificielle par le projet ARPA Knowledge Sharing Effort [159].

La définition communément admise d'une ontologie est énoncée par T. Gruber [105] comme la "spécification explicite d'une conceptualisation". Le but d'une ontologie informatique est de permettre de représenter formellement la sémantique d'un domaine et de supporter certains mécanismes de traitement automatique.

Avec le développement important des études sur les ontologies au cours des dix dernières années, de nombreuses autres définitions ont été proposées. Chaque communauté adopte sa propre interprétation selon l'usage qu'elle en fait et le but qu'elle vise. Ainsi le terme ontologie recouvre en fait des réalités assez différentes. Pour beaucoup d'auteurs, néanmoins, tels par exemple Berners-Lee [49] à propos du Web sémantique, une ontologie réunit à la fois des éléments, concepts ou mots, et des règles permettant de manipuler ces éléments ou d'effectuer un certain nombre d'inférences.

Nous présentons maintenant plus en détails les caractéristiques communes des ontologies.

2.1.2 Caractéristiques communes des ontologies

Même si les terminologies et objectifs de chaque modèle d'ontologie peuvent différer, pratiquement tous les modèles sont basés sur les mêmes notions. Ces notions sont les suivantes :

- Les **classes** (ou concepts ou entités). Ils représentent des groupes d'individus partageant les mêmes caractéristiques. Ils correspondent aux entités "génériques" d'un domaine d'application. Les concepts d'une ontologie sont organisés hiérarchiquement par une relation d'ordre partiel qui est la relation de subsomption (*"est-une"*) permettant d'organiser sémantiquement les concepts par niveau de généralité. Intuitivement, un concept C_1 *subsume* un concept C_2 si C_1 est plus général que C_2 au sens où l'ensemble d'individus représenté par C_1 contient l'ensemble d'individus représenté par C_2. Par exemple, le concept *Personne* subsume le concept *Femme*. Formellement, $C1$ subsume $C2$ si dans tout contexte: $x \in C2 \Rightarrow x \in C1$.
- Les **relations**. Elles constituent des types d'association pré-définis entre les concepts. La relation commune qui est supportée par n'importe quel formalisme d'ontologie est *la subsomption: "est-un"*. Elle organise les concepts en une hiérarchie, où tout concept se compose d'une description propre définie par des propriétés *locales* et d'une description *partagée* avec ses subsumants comme c'est le cas entre classes dans un langage à objets [156]. La relation de subsomption est *transitive* (c'est-à-dire si C_1 subsume C_2 et C_2 subsume C_3 alors C_1 subsume C_3), *réflexive* (C_1 subsume C_1 lui même), et *antisymétrique* (si C_1 subsume C_2 et C_2 subsume C_1, alors $C1 = C2$).
- Les **instances** (ou individus ou objets). Elles représentent des individus du domaine d'ontologie.
- Les **axiomes**. Ils explicitent les énoncés conceptuels toujours vrais dans le contexte de l'ontologie. Ils peuvent être utilisés pour contrôler la correction des concepts ou des relations, ou pour déduire de nouveaux faits.

Toutes ces entités ontologiques, c'est-à-dire les classes, les propriétés, les relations, les instances et axiomes, doivent être définis explicitement à l'aide d'un langage ou d'un modèle d'ontologie. Cette sémantique peut être plus ou moins formelle, le degré de formalisation dépendant du but des applications qui utilisent l'ontologie. Il existe plusieurs modèles d'ontologies proposés dans la littérature comme F-Logic [166], RDF/RDF Schéma [145], DAML+OIL [90], OWL [146], PLIB [179], etc.

Un point commun à tous les modèles d'ontologies est la distinction entre les concepts *primitifs*, dont l'ontologie ne fournit pas de définition complète mais seulement quelques conditions nécessaires, la définition complète " reposant sur une documentation textuelle et un savoir préexistant partagé avec le lecteur " [105], et les concepts *définis* par l'ontologie, dont elle fournit des conditions nécessaires et suffisantes de reconnaissance en termes de concepts primitifs.

Nous présentons ici une taxonomie des ontologies conceptuelles. Nous distinguons ici deux catégories d'ontologies conceptuelles: (1) les **ontologies canoniques** qui ne contiennent que des concepts primitifs, et (2) les **ontologies non canoniques** qui contiennent à la fois des concepts primitifs et des concepts définis.

2.1.2.1 Ontologie canonique

Pour pouvoir représenter tous les concepts existants dans un certain domaine, une ontologie conceptuelle n'a besoin de décrire que les concepts qui ne peuvent pas être dérivés d'autres concepts. En effet, les concepts définis à partir d'autres concepts n'augmentent nullement le domaine couvert par l'ontologie puisque, par définition, ces concepts peuvent déjà se représenter à l'aide de la terminologie définie par les concepts primitifs. Comme dans le domaine des bases de données (où toute redondance est exclue), dans le vocabulaire technique (où un et un mot seul doit être utilisé pour chaque notion) et dans les formats d'échange (où il existe une seule représentation possible pour chaque information échangée), l'existence d'un langage *canonique* (une unique représentation pour chaque notion) constitue un avantage par rapport au langage contenant des synonymies (pour l'échange et l'intégration de données, etc.).

Nous appelons *ontologie conceptuelle canonique* (*OCC*) une ontologie conceptuelle ne contenant que des éléments primitifs.

PLIB [181] est un exemple de modèle d'ontologies conceptuelles canoniques sur lequel nous avons travaillé. Partant du constat qu'on ne peut définir de nouveaux termes qu'à partir de termes primitifs, et que les termes primitifs d'un domaine technique sont eux-mêmes très nombreux et difficiles à appréhender, l'objectif essentiel d'une ontologie PLIB est de définir, mais de la façon la plus précise et la plus concise possible, les classes et propriétés primitives qui caractérisent les objets d'un domaine du monde réel et les abstractions (ou modèles) que les différentes communautés peuvent en construire. L'objectif de concision signifie que PLIB ne définit de nouvelles classes que lorsque celles-ci ne peuvent complètement se définir en termes d'autres classes et de valeur de propriétés : il s'agit de classes primitives. L'objectif de précision signifie trois choses. Face à un objet matériel ou un artefact donné appartenant au domaine ciblé par une ontologie, un utilisateur humain de l'ontologie doit savoir décider : (1) à quelles classes l'objet appartient et n'appartient pas, (2) quelles propriétés s'appliquent à l'objet, et (3) quelles grandeurs ou valeurs caractéristiques correspondent à chaque propriété applicable. Par ailleurs, l'ontologie doit définir de façon formelle toutes les conditions nécessaires qui peuvent être vérifiées par un agent

informatique lors de l'échange d'une ontologie.

Pour PLIB, une ontologie est donc une collection de descriptions *explicites*, *formelles* et *consensuelles* de l'ensemble des concepts d'un domaine [176] :

- dans le contexte le plus large où ces concepts ont un sens précis, et,
- sans aucune restriction ou règle correspondant à une utilisation particulière.

Dans ce contexte,

- *explicite* signifie que les types de concepts utilisés et les contraintes sur leurs usages sont explicitement définis.
- *formel* signifie exploitable par la machine pour exécuter certains traitements.
- *partagé* signifie que le contenu est consensuel, accepté par un groupe; ceci est assuré, par exemple, par l'existence d'un processus normatif.

Le modèle PLIB présente les caractéristiques suivantes :

1. Il vise à permettre l'échange de données, PLIB n'offre pour l'instant aucun opérateur pour exprimer les équivalences conceptuelles. Les ontologies PLIB sont toujours canoniques. Ce sont des ontologies de caractérisation et non de déduction.
2. Il propose un modèle riche pour décrire avec précision les concepts primitifs
3. Il vise le développement d'ontologies très vastes (pour couvrir tout le domaine technique). PLIB offre, en effet, des mécanismes originaux de modularité d'ontologies qui interfacent une ontologie à une autre ontologie et importent les propriétés de cette dernière dans la première. Cet opérateur de modularité s'appelle *"is_case_of"* : il s'agit d'une subsomption sans héritage implicite, mais avec importation explicite des propriétés pertinentes.
4. Il possède plusieurs primitives apparues nécessaires dans certains domaines (en particulier les domaines techniques) et non disponibles dans les autres modèles, par exemple la définition contextualisée de valeurs, les contrainte d'intégrité ou la gestion des points de vue.

2.1.2.2 Ontologie non canonique

Des relations d'équivalence entre concepts peuvent également être représentées dans une ontologie canonique. Ce type de relations permet de représenter les concepts définis. Ceci peut être fait en utilisant une ou des relations formelles de classe, que nous appelons également des *opérateurs de classes*, comme les opérateurs orientés ensemble (union, intersection et différence) et les opérateurs de restrictions sur les valeurs de propriétés. Ce peut également être fait en définissant des relations entre propriétés, que nous appelons aussi des *opérateurs de propriétés*, telles que les relations algébriques ou logiques.

Une ontologie où des équivalences conceptuelles sont représentées est appelée *ontologie conceptuelle non-canonique* (OCNO). Ce type d'ontologie permet de représenter à la fois différentes conceptualisations d'un même domaine, et les correspondances entre celles-ci. Ceci permet, en effet, aux différents partenaires d'un échange d'associer leurs données à leurs propres ontologies. Puisque les partenaires ont exprimé leurs points de vue sur le même domaine d'étude, ils doivent pouvoir exprimer leurs concepts en termes de concepts de l'ontologie partagée.

La figure 2.1 illustre un modèle en couches nommé le modèle en oignon d'une ontologie de domaine. Un modèle montre comment les différentes catégories d'ontologies peuvent être combinées [127, 125].

Pour conclure sur les différents types d'ontologie existants, une OCC fournit une base formelle pour modéliser et échanger efficacement la connaissance d'un domaine. Une OCNC fournit les mécanismes pour lier différentes conceptualisations faites sur ce domaine. Finalement, les ontologies linguistiques (OL) fournissent une représentation en langage naturel des concepts de ce domaine, éventuellement dans différents langages.

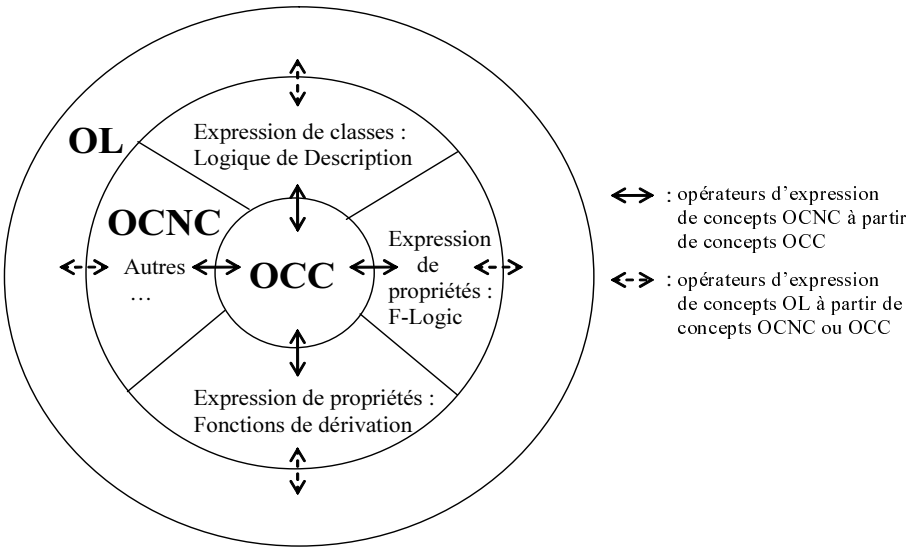


FIG. 2.1 – Le modèle en oignon pour les ontologies de domaine

2.1.3 Représentation formelle d'une ontologie PLIB

Formellement (voir [179] pour un modèle plus complet), une ontologie PLIB peut être définie comme un 4-uplet : $O :< C, P, Sub, Applic >$, avec:

- C : l'ensemble des classes utilisées pour décrire les concepts d'un domaine donné. Chaque classe est associée à un identifiant universel globalement unique (BSU) .
- P : l'ensemble des propriétés utilisées pour décrire les instances de l'ensemble des classes C. P définit toutes les propriétés susceptibles d'être présentes dans une base de données dont le modèle adhère à l'ontologie O. Chaque propriété est associée à un identifiant universel globalement unique (BSU). P est factorisé en trois sous-ensembles :
  1. P_{val} représentant l'ensemble des propriétés caractéristiques.
  2. P_{func} représentant l'ensemble des propriétés dépendantes de contexte.
  3. P_{cont} représentant l'ensemble des paramètres de contexte.
- $Sub : C \rightarrow 2^C$, est la relation de subsomption [3] qui, à chaque classe c_i de l'ontologie, associe ses classes subsumées directes. Sub définit un ordre partiel sur C.
  1. $OOSub$ ("is-a"): représente la relation de subsomption avec héritage; elle est seulement utilisée entre deux classes d'une même ontologie et doit définir des hiérarchies simples.

[3] 2^C désigne l'ensemble des parties de C

2. *OntoSub* ("is-case-of"): représente la relation de subsomption sans héritage; elle est en particulier utilisée entre deux classes de deux ontologies. *OntoSub* permet alors d'articuler *a priori* une ontologie locale avec une ontologie partagée et d'importation des propriétés pertinentes.

– $Applic : C \rightarrow 2^P$, associe à chaque classe de l'ontologie les propriétés qui sont applicables pour chaque instance de cette classe. Les propriétés qui sont applicables sont héritées à travers la relation *OOSub* et elles peuvent être importées de façon explicite à travers la relation de *OntoSub*.

Soulignons que les définitions ontologiques sont intentionnelles. Le fait qu'une propriété soit applicable pour une classe signifie qu'elle est rigide [106] c'est-à-dire essentielle pour chaque instance de la classe. Cela ne signifie pas qu'une valeur sera explicitement représentée pour chaque instance dans la base de données. Dans notre approche, le choix des propriétés effectivement représentées qualifiées de propriétés utilisées est fait au niveau du choix du schéma parmi les propriétés applicables. La section suivante discutera précisément de la représentation des instances.

Ayant présenté la notion d'ontologie et de modèle d'ontologies, nous présentons les BDBO.

2.1.4 Bases de données à base ontologique

Les bases de données permettant de gérer simultanément des ontologies et des données qui les référencent sont appelées bases de données à base ontologique (BDBO). Une BDBO possède deux caractéristiques :

– les ontologies et les données sont toutes deux représentées dans une unique base de données et peuvent faire l'objet des mêmes traitements (insertion, mise à jour, interrogation, versionnement, etc.);
– toute donnée est associée à un élément ontologique qui en définit le sens et vice versa.

De nombreuses BDBO ont été proposées dans la littérature incluant Sesame [62], RDFSuite [3], Jena [52, 215, 63], OntoDB [82, 180], DLDB [170], RStar [141], KAON [60], 3Store [113] et PARKA [199]. Ces BDBO se différencient suivant :

– le modèle d'ontologies supporté ;
– le schéma de base de données utilisé pour stocker des ontologies représentées selon ce modèle ;
– le schéma de base de données utilisé pour représenter les données à base ontologique (instances) ;
– les mécanismes utilisés pour définir le lien entre les données (instances) et les éléments des ontologies (classes).

2.2 Schémas de représentation des ontologies et des données

La problématique de la représentation des ontologies dans les bases de données est :

(1) de définir un schéma de tables pour le stockage des données de chacune des parties (ontologie et données). En effet, les ontologies et les instances des classes d'ontologies ont une structure objet, c'est-à-dire qu'elles représentent l'information sous forme de classes avec des relations (de composition et de subsomption) entre elles et sont constituées d'un ensemble de propriétés carac-

téristiques. La représentation des objets dans des bases de données relationnelles ou relationnelles-objets pose des difficultés car tous les concepts objets ne se traduisent pas directement en relationnel et les procédures de gestion (ajout, mise à jour, suppression, versionnement) des instances sont entièrement à définir [189, 197, 202].

(2) de définir un mécanisme pour lier les instances (de la partie "données") avec les éléments ontologiques (de la partie "ontologie") qui en définissent le sens et vice versa.

2.2.1 Représentation des ontologies

Les différentes approches de représentation des ontologies peuvent être classées en deux catégories : approche de représentation verticale et approche de représentation spécifique.

Dans l'approche verticale, les descriptions des classes et des propriétés des ontologies (découpées en triplets RDF) sont stockées naturellement sous forme de triplets (sujet-prédicat-objet) dans une unique table nommée *triplets* [52, 1, 215, 63]. La table *triplets* est constituée de trois colonnes : (sujet, prédicat, objet).

Dans l'approche de représentation *spécifique*, la structure des tables se rapproche de la structure du (ou des) langage(s) de définition d'ontologie(s) à intégrer. Le schéma de ces tables est défini de façon ad hoc (c'est-à-dire, de façon non systématique) d'une implémentation à une autre. Les implémentations définissent la structure de toutes ces tables suivant les détails des informations qu'ils veulent capturer et du lien fait avec la partie *données*. Par exemple pour des ontologies RDF Schema, leur schéma contient en général les tables suivantes : *class*, *subclass*, *property*, *subproperty*, *domain*, *range*.

2.2.2 Représentation des données

Les différentes approches de représentation des données à base ontologique peuvent être classées en trois catégories : approche de représentation *générique* ou *verticale*, approche de représentation *binaire* et approche de représentation *hybride*.

Dans l'approche de représentation *verticale* les données à base ontologique sont représentées sous forme de triplets (subject, predicate, object) comme dans l'approche verticale de la représentation des ontologies. Une variante de cette approche consiste à stocker respectivement l'identifiant (URI) des ressources et toutes les valeurs littérales dans des tables spécifiques : *ressources* et *littéral*, référencée ensuite par les triplets.

Dans l'approche de représentation *binaire*, les instances des classes et les valeurs de leurs propriétés sont stockées dans des tables séparées (les instances dans une table unaire et les valeurs de chaque propriété dans une table binaire).

L'approche *hybride* combine l'approche de représentation par triplets et l'approche de représentation *binaire*. Les valeurs des propriétés selon leur co-domaine (Entier, String, etc.) sont représentées dans des tables séparées. On aura une table *triplets* pour chacun des types de bases (INT, VARCHAR, FLOAT, etc.) : regroupant toutes les propriétés du même type.

Dans toutes les approches de représentation des données à base ontologique, les instances et les valeurs des propriétés sont représentées séparément soit sous forme de triplets dans une grande table soit

dans des tables de propriétés. Ces choix de représentation sont justifiés par le *faible typage* qu'offrent les langages de définition d'ontologies usuels (RDF Schéma, DAML+OIL, OWL). En effet, ces langages de définition d'ontologies sont caractérisés par le fait qu'ils supportent (1) la multi-instanciation, (2) les propriétés dont le domaine ou co-domaine n'est pas défini et (3) le fait qu'une instance initialise des propriétés non définies dans le contexte de sa classe.

Notons également que ces approches de représentation des données à base ontologique sous forme *éclatée* peuvent être efficaces pour les requêtes de mises à jour, où les ontologies utilisées ne subissent pas des évolutions constantes (l'ajout d'une nouvelle propriété, la suppression d'une propriété).

2.2.3 Caractéristiques et fonctionnalités des systèmes de gestions des bases de données à base ontologique

Afin de mieux comprendre les systèmes de bases de données à base ontologique, nous avons étudié l'architecture des principaux systèmes : Sesame [62], ICS-FORTH RDFSuite [3], Jena[4] et KAON [60]. Cette étude nous a, à la fois, permis d'identifier des caractéristiques et fonctionnalités communes à tous ces systèmes que l'on devra retrouver dans notre système, mais aussi ce qui manquait aux systèmes existants par rapport à nos besoins. Concernant ce qui était commun on trouve :

1. **Respect d'un standard.** Les systèmes de gestion de BDBOs sont tous basés sur un ou plusieurs langages de définition d'ontologies principalement : RDF, RDFS, DAML+OIL, OWL.

2. **Échange de données.** Les systèmes de gestions de BDBOs offrent la possibilité d'importer dans la base de données des ontologies et des données à base ontologie définies dans l'un des langages du point précédent. Et inversement, ils permettent d'extraire un sous-ensemble des ontologies et/ou un sous-ensemble des populations d'instances. Cette fonctionnalité permet le partage des données entre des systèmes.

3. **Langage de requêtes.** Les systèmes de gestion des BDBOs offrent des langages de requêtes (RQL ou SPARQL) qui permettent (1) la création des concepts (classes, propriétés, etc.), (2) la création d'instances des concepts et (3) l'interrogation pour retrouver des concepts ou/et des instances.

4. **API d'accès aux ontologies et leurs données**. Les systèmes de gestion des BDBOs offrent des interfaces de programmations pour l'accès aux ontologies et leurs données représentées dans les bases de données. Ceci en vue d'abstraire les schémas de données sous-jacents pour les applications qui accèdent aux données.

Notons toutefois qu'il existe d'autres fonctionnalités et caractéristiques *souhaitables* que ces systèmes ne supportent malheureusement pas et qui seraient utiles pour les applications des domaines qui nous intéressent particulièrement. Ces caractéristiques et fonctionnalités sont entre autres :

1. **La notion de schéma.** Tous les systèmes, que nous avons présentés, représentent les données à base ontologique dans l'une des approches que nous avons présentées (sous forme de *verticale* ou

[4] http://jena.sourceforge.net/

sous forme *binaire*). Une instance donnée dans ces systèmes ayant un certain nombre de propriétés est décomposée en plusieurs tuples dans différentes tables (dans l'approche *binaire*) ou dans la grande table *triplets* (dans l'approche verticale). L'accès à une instance nécessitera donc de faire de nombreuses jointures (ou auto-jointures) qui sont coûteuses. La notion de schéma de données au sens usuel des bases de données n'est pas présente. Les schémas de données sont importants car ils permettent de traiter de façon rapide de très gros volume de données, et car ils offrent la possibilité d'appliquer les différents mécanismes (indexation, regroupement (clustering), etc.) offerts par les SGBDs pour l'optimisation des requêtes.

2. **La gestion des versions et des évolutions des ontologies**. Cette fonctionnalité est nécessaire pour un certain type d'applications utilisant des ontologies susceptibles d'évoluer en fonction des besoins. En effet, l'évolution d'une ontologie occasionne la définition de nouvelles versions de cette ontologie. Ceci particulièrement, le cas des ontologies de domaines techniques où leur processus de normalisation peut prendre plusieurs années; des versions préliminaires sont définies progressivement jusqu'à la version finale. Leur réintégration dans les BDBOs doit être faite de manière à préserver les relations sémantiques existantes entre les différents concepts des ontologies. Il est nécessaire d'offrir également des primitives pour accéder et manipuler les versions des concepts.
3. **Le cycle de vie des instances.** Comme les ontologies, les données à base ontologique évoluent également. Deux types d'évolutions peuvent se produire. Le premier type d'évolution des données à base ontologique est directement lié à l'évolution des ontologies. En effet, d'une version à une autre de concepts des ontologies (les classes exemples), des modifications surviennent systématiquement sur la structure des nouvelles instances de ces concepts. Le deuxième type d'évolution est lié aux opérations (l'ajout, la suppression, la mise à jour) sur les instances, leurs propriétés ou leur schéma qui peuvent être appliquées sur les données à base ontologique. Pour les applications n'utilisant pas les mêmes versions des concepts d'une ontologie, il est souhaitable d'offrir d'une part, un mécanisme pour pouvoir accéder à toutes les instances des concepts existantes dans une version donnée dans la BDBO, et d'autre part, des mécanismes pour une gestion du cycle de vie des instances des classes. On devrait pouvoir interroger la BDBO pour retrouver entre autres :
   – la version à laquelle une instance d'une classe a été insérée dans la base de données ou supprimée,
   – les différentes propriétés utilisées pour une classe à une certaine version,
   – les versions auxquelles une propriété donnée a été initialisée ou supprimée.

Notons que la mise en œuvre de la plupart de ces fonctions exige qu'elle soit prise en compte directement au niveau des schémas de tables où sont stockées les ontologies et les données à base ontologique. Nous avons considéré ces caractéristiques et fonctionnalités comme des exigences à prendre en compte dans les schémas de représentation des ontologies et de leurs données.

2.3 Objectifs et Hypothèses

Le constat réalisé dans la section précédente nous a conduit à concevoir un nouveau système de gestion de BDBOs. Nous commençons par décrire les objectifs de ce nouveau système.

2.3.1 Objectifs

Notre système de gestion de BDBOs doit permettre la représentation des ontologies, des données et le lien entre eux de façon à pouvoir accéder à l'une des parties à partir de l'autre. Ontologies et données doivent toutes deux être représentées dans une unique base de données et doivent pouvoir faire l'objet des mêmes traitements (insertion, mise à jour, interrogation, versionnement, etc.). En plus de ces exigences, nous avons imposé à notre système d'atteindre les quatre objectifs supplémentaires suivants :

O_1- L'intégration automatique et la gestion homogène des populations d'instances dont les données, le schéma et les ontologies sont chargés dynamiquement.

O_2- L'adaptation facile (1) à des évolutions du modèle d'ontologie utilisé et (2) au changement de modèle d'ontologie. L'adaptation (1) est particulièrement importante pour le modèle d'ontologie PLIB utilisé dans le cadre de nos projets et qui était en cours de normalisation à l'ISO. Ce modèle subissait, en moyenne tous les 6 mois, des modifications. L'adaptation (2) visait, en particulier, à permettre de représenter des ontologies basées sur les logiques de description (exemple, *OWL*). Cette modification devrait être possible par une simple extension du schéma logique de la base de données.

O_3- Les accès génériques tant aux ontologies qu'aux instances, en permettant à l'utilisateur de faire abstraction de l'implémentation particulière sur un SGBD particulier (relationnel, relationnel objet ou objet par exemple).

O_4- La gestion des évolutions des ontologies tout en fournissant des mécanismes permettant la gestion du cycle de vie des instances.

2.3.2 Hypothèses

Compte tenu de la grande diversité des modèles d'ontologies existants, il est peu vraisemblable de trouver un modèle d'architecture de BDBO qui s'avère efficace dans tous les contextes. Nous définissons donc à la lueur de l'application que nous visons à traiter un certain nombre de restrictions en considérant certaines hypothèses sur : (1) les modèles d'ontologies qui peuvent être utilisés pour la mise en œuvre de notre architecture *OntoDB* et (2) les ontologies et les données à base ontologiques qui peuvent être représentées dans une base de données *OntoDB*.

H_1 - Nous avons repris tout d'abord les exigences de l'architecture RDFSuite [3] :

R_1 **-Typage fort des propriétés.**

- toutes les propriétés des classes doivent avoir obligatoirement un domaine et un co-domaine.
- le domaine et le co-domaine d'une propriété doivent être uniques.

R_2 **-Complétude de définition.**

- les descriptions complètes de tous les concepts qui contribuent à la définition d'un concept doivent pouvoir être représentées dans le même environnement (e.g., fichier) que celui où ce concept est défini. Ceci suppose en particulier que les super-classes d'une classe, si elles existent, soient connues, de même que les domaines et le co-domaine de ses propriétés.

H_2 - Les ontologies considérées sont celles qui peuvent s'exprimer sous forme de modèle, au sens de Bernstein [51, 50], c'est-à-dire d'un ensemble d'objets accessibles à partir d'un objet racine, et qui décrit un univers sous forme de classes et de propriétés.

H_3 - Les données considérées sont celles qui représentent des instances des classes de l'ontologie (aussi appelées par certains auteurs *individus*).

H_4 - On exige que ces instances respectent l'hypothèse de *mono-instanciation* définie par la règle suivante :

R_1 **-Mono-instanciation**

– toute instance du domaine est décrite par son appartenance à une et une seule classe, dite classe de base qui est la borne inférieure unique pour la relation de subsomption de l'ensemble des classes auxquelles elle appartient (elle appartient bien sûr également à ses super-classes).

Notons que, de l'hypothèse $H_1(R_1)$ et $H_4(R_1)$, on peut déduire la règle R_2 : *typage fort des instances*.

R_2 **- Typage fort des instances**

– toute instance du domaine ne peut être décrite que par les propriétés applicables à sa classe de base, c'est-à-dire celles dont le domaine subsume cette classe de base.

H_5 - Enfin, pour permettre la gestion automatique des différentes versions d'une même ontologie, on exige que l'évolution des ontologies obéisse au principe de continuité ontologique [219, 218] (détaillé dans le Chapitre 4).

2.4 Nos propositions pour la représentation des ontologies

2.4.1 Schéma utilisé pour stocker les ontologies

Nous avons vu (hypothèse H_2) que les ontologies auxquelles nous nous intéressons sont celles susceptibles d'être représentées sous forme d'un modèle au sens de Bernstein [51], c'est-à-dire d'un ensemble d'objets accessibles à partir d'un objet racine par des relations de composition et qui décrivent un univers sous forme de classes et de propriétés. Cette définition correspond à la plupart des modèles d'ontologies récents tels que OWL [78], PLIB [214, 122, 177]. Une telle ontologie est donc représentée comme une instance d'un schéma objet (souvent appelé méta-modèle) dans un formalisme de modélisation particulier (XML-Schema pour OIL et OWL, EXPRESS [193] pour PLIB). Cette représentation, à son tour, fournit à la fois un modèle conceptuel et un format d'échange pour les ontologies visées (document XML pour OWL, fichier physique d'instances EXPRESS pour PLIB).

Pour définir notre modèle, nous avons commencé, à partir des objectifs évoqués dans la section 2.3.1, par analyser les besoins que nous souhaitons qu'il satisfasse. Ces exigences sont définies sous forme d'un ensemble de fonctions devant être supportées par notre système. L'analyse de ces fonctions nous a permis de discuter et de décrire avec précision les différents éléments ou composantes du modèle d'architecture nécessaire.

– **F1 : capacité de stockage interne des ontologies au sein d'un schéma logique adapté au SGBD cible**. Celui-ci doit pouvoir être de type relationnel, relationnel-objet ou orienté objet.

– **F2 : interface générique d'accès par programme aux entités définissant une ontologie**. Cette interface est dite *générique* parce qu'elle doit être définie indépendamment de tout modèle d'on-

tologie particulier, ceci pour permettre à notre architecture de BDBO de pouvoir supporter la représentation d'ontologies de modèles évolutifs ou issus de modèles différents.

– **F3 : interface spécifique d'accès par programmation orienté-objets aux entités de l'ontologie**. Il s'agit ici de proposer une interface qui soit (1) orientée objet, (2) spécifique d'un langage de programmation particulier et (3) spécifique d'un modèle d'ontologie donné (par exemple OWL, PLIB, RDF Schéma ou DAML+OIL) mais qui permet de contrôler la correction syntaxique de tous les accès aux entités de l'ontologie (interface à typage fort ou "early binding"). Le caractère spécifique de cette API est lié au fait que les ontologies sont des instances de modèles objets particuliers et bien définis, et que les fonctions de l'API sont définies de façon à retourner les données selon la structure *objet* spécifique de ces modèles.

– **F4 : capacité de lire simultanément les ontologies et des instances représentées dans leur format d'échange et de les stocker dans la BDBO**. Notons que lorsque les ontologies et leurs instances sont susceptibles d'évoluer, cette fonction est particulièrement délicate car elle doit permettre de gérer lors de la lecture, l'évolution des ontologies. Les concepts et les instances des ontologies pouvant être utilisés par des applications dans différentes versions, il est indispensable que la BDBO possède des mécanismes pour garder une trace de toutes les versions des concepts successives de façon à assurer une *compatibilité ascendante* pour les applications.

– **F5 : capacité d'exporter des concepts des ontologies avec éventuellement leurs instances**. Cette fonction est duale de la précédente fonction de lecture. Il s'agit ici d'être capable d'extraire de la BDBO un sous-ensemble de concepts de l'ontologie (dans une certaine version) ainsi qu'un sous-ensemble d'instances associées à ces concepts. Les concepts extraits de la base de données doivent être cohérents, i.e., leurs définitions doivent pouvoir être complètes (cf. hypothèses H_2). Un concept est dit *complet* lorsque tous les éléments qui participent à sa définition sont présents dans le référentiel où il se trouve. Par exemple, pour une classe C, on doit trouver dans son référentiel, la définition de ses super-classes et de ses propriétés applicables et non uniquement leur identifiant (URI, BSU). Si l'utilisateur le souhaite, la fonction d'extraction doit permettre d'extraire *sémantiquement* les concepts et d'analyser toutes les dépendances de sorte à les extraire également et de façon récursive.

2.4.2 Besoin de représentation du méta-modèle

Nous avons identifié que pratiquement toutes les fonctions identifiées (F_1 à F_5) suggéraient la représentation, dans la base de données, d'un méta-modèle d'ontologie, et d'une représentation des modèles d'ontologies en tant qu'instances de ce méta-modèle. La représentation du modèle d'ontologie dans la base de données permet de rendre générique, par rapport aux modèles d'ontologies utilisés dans la BDBO, toutes les fonctions que nous avons identifiées et qui devront faire partie du système. De plus, ce méta-modèle peut être utilisé pour représenter les modifications possibles du modèle d'ontologie considéré. Ainsi, la représentation d'un méta-modèle d'ontologie dans les BDBOs répond aux mêmes besoins que la méta-base pour les bases de données classiques.

Dans le cas de la fonction F_1, nous avons préconisé d'utiliser les techniques de correspondance objet-relationnel pour la définition de la structure des tables permettant la représentation des ontologies. Vu que le méta-modèle est défini dans le même formalisme que le modèle d'ontologie, on pouvait envisager

d'utiliser la même technique pour définir la structure de tables permettant la représentation des modèles d'ontologie dans la base de données.

Notons que si le formalisme utilisé pour définir le modèle d'ontologie et le méta-modèle est le même, *alors le même code générique pourra être réutilisé* pour la mise en œuvre des mêmes besoins concernant les modèles d'ontologies et le méta-modèle. Ceci s'applique aux fonctions F_1 (pour définir des tables), F_2 et F_3 (pour accès aux modèles d'ontologies), F_4 (pour lire les modèles d'ontologies).

Notons que la représentation du modèle d'ontologie dans la base de données offre la possibilité de représenter au sein d'une même base de données plusieurs ontologies issues de modèles d'ontologies différents. Il suffit dans ce cas d'appliquer, sur chacun des modèles d'ontologies, les traitements mis en oeuvre pour le déploiement d'un modèle d'ontologie particulier (PLIB sur la BDBO OntoDB). Ceci n'est néanmoins possible que sous deux conditions :

1. Tous les modèles d'ontologies doivent être définis dans un même et unique formalisme de modélisation (celui sur lequel les programmes de déploiement se basent). Pour le prototype que nous avons implémenté, le formalisme utilisé fut le langage EXPRESS [193]. Ainsi si nous souhaitons créer/insérer des ontologies PLIB, RDF Schema, DAML+OIL et OWL dans OntoDB, alors les modèles d'ontologies RDF Schema, DAML+OIL, OWL doivent être "re-modélisés" dans le formalisme EXPRESS.
2. Pour pouvoir importer des ontologies RDF Schema, DAML+OIL, OWL, un *convertisseur*, i.e., un programme, qui servira à faire migrer les ontologies dans leur format d'origine (XML pour RDF Schema, DAML+OIL, OWL) dans le format d'instances du formalisme sur lequel se base les programmes de déploiement, doit être défini. Dans notre prototype, ce sera donc des instances de fichiers physiques EXPRESS [121].

Cette approche automatique a néanmoins l'inconvénient de ne pas factoriser les éléments communs de tout modèle d'ontologie (i.e., les classes, les propriétés et les types). Une autre approche possible proposée dans le cadre des travaux de thèse de Stéphane JEAN sur le développement d'un langage de requêtes spécifique à notre base de données à base ontologique (OntoQL), consiste à étendre manuellement dans le le modèle d'ontologie existant (PLIB) pour y introduire de nouvelles spécifications.

2.4.3 Méta-schéma réflexif

Le besoin identifié dans la section précédente nous a conduit à définir une partie dans notre BDBO nommée méta-schéma pour stocker le modèle d'ontologie utilisé. Les exigences définies jusqu'alors sur le *méta-schéma*, étaient d'être capable de représenter l'ensemble des variantes des modèles d'ontologies envisagés. Si nous imposons alors au *méta-schéma* d'être, de plus, réflexif, c'est-à-dire de pouvoir se représenter lui-même, on pourra représenter le *méta-modèle* lui-même comme instance de sa propre structure.

L'autoreprésentation du méta-modèle réflexif au sein de sa propre structure constitue donc un avantage significatif pour l'écriture de programmes plus simple. L'accès à un niveau "méta" en base de données pouvant avoir pour effet de ralentir les programmes, il convient de noter que la représentation explicite du méta-modèle dans la base de données n'empêche pas d'optimiser les programmes afin qu'ils y accèdent le moins possible.

2.5 Nos propositions pour la représentation des données à base ontologique

2.5.1 Position du problème et hypothèses

Une ontologie vise à représenter la sémantique des objets d'un domaine en les associant à des classes, et en les décrivant par des valeurs de propriétés. Selon les modèles d'ontologies utilisés, plus ou moins de contraintes existent sur ces descriptions. A titre d'exemple, si on n'introduit pas de restrictions particulières sur les descriptions OWL, un objet peut appartenir à un nombre quelconque de classes et être décrit par n'importe quelles propriétés. Ceci donne à chaque objet du domaine une structure qui peut lui être spécifique. A contrario, un schéma de base de données vise à décrire des ensembles d'objets "similaires" par une structure logique identique de façon à pouvoir optimiser les recherches de tels ensembles par des techniques d'indexation.

En l'absence de toute hypothèse particulière sur la description à base ontologique des objets du domaine, une structure de gestion de l'ensemble des objets doit permettre d'associer à chaque objet :

- un sous-ensemble quelconque de l'ensemble des classes,
- un nombre quelconque de propriétés.

Cette exigence entraîne alors soit un coût de stockage supplémentaire, avec une représentation systématique des appartenances ou des propriétés non existantes pour une instance, soit des temps de traitements importants résultant de la nécessité de réaliser un nombre important de jointures, si seules les appartenances et les propriétés existantes sont représentées.

En fait, dans bien des cas, et c'est en particulier le cas dans le domaine de l'ingénierie et du commerce électronique [175] :

1. Les propriétés sont typées : chacune est associée à une classe qui définit l'ensemble des objets auxquels la propriété peut être appliquée et a un domaine de valeurs.
2. Chaque objet est caractérisé par l'appartenance à une et une seule classe, même s'il peut, par ailleurs être décrit par des instances d'autres classes pour prendre en compte une multiplicité de points de vue de modélisation.

Dans le cadre de notre modèle d'architecture *OntoDB*, nous imposerons deux restrictions désignées sous le terme d'hypothèse de *typage fort*.

- **R1-** tout objet du domaine est *caractérisé* par son appartenance à une et une seule classe, dite classe de base qui est la borne inférieure unique, pour la relation de subsomption, de l'ensemble des classes auquel il appartient (il appartient bien sûr également à ses super-classes).
- **R2-** tout objet du domaine ne peut être décrit que par les propriétés applicables à sa classe de base (ceci signifie conformément à notre modèle formel, si C est la classe de base d'un objet o seule les propriétés appartenant à $applic(C)$ peuvent être utilisées pour décrire o).

Soulignons que l'hypothèse R_2 n'impose en aucun cas que *toutes* les propriétés applicables d'une classe soient effectivement utilisées pour en décrire les instances. Le choix des propriétés à utiliser effectivement pour les instances d'une classe d'une BDBO, ce que l'on peut appeler son *schéma* ou son *modèle*

conceptuel, dépend de l'objectif particulier de la BDBO. Il est, par exemple, possible qu'une propriété définie comme applicable au niveau d'une classe A soit effectivement représentée dans le schéma de deux sous-classes (A_1 et A_2) de A, non dans celui d'une troisième sous-classe A_3. Notons que ceci constitue une différence majeure avec une base de données orientée objet, toutes les classes ont un schéma d'instances et où les propriétés qui caractérisent les instances sont constituées *automatiquement* de *toutes* les propriétés applicables des classes. La seule possibilité pour que deux sous-classes partagent une *même* propriété est que cette propriété soit *héritée* d'une super-classe commune. Elle est alors *également héritée par toutes les sous-classes* de cette super-classe.

2.5.2 Notre proposition de schéma pour les instances des classes

Notre hypothèse de typage fort R_1 et R_2 permettant de définir, pour chaque classe, un schéma maximum permettant de décrire toutes les instances de cette classe. Il s'agit du schéma comportant toutes les propriétés applicables de la classe.

Néanmoins, comme nous l'avons souligné, le choix des propriétés applicables effectivement utilisées dans une BDBO dépend de l'objectif applicatif de la BDBO. Pour chaque classe donnée, il se peut que seul un sous-ensemble des propriétés soit effectivement utilisé pour décrire tout ou partie des instances. Le schéma final peut donc être traduit lorsqu'on lit l'ensemble des instances : c'est le sous-ensemble des propriétés applicables qui sont valuées pour au moins une instance.

Il convient de remarquer que ces schémas *ne sont pas nécessairement liés par une relation d'héritage*. Ainsi, comme nous l'avons noté ci-dessus, deux sous-classes A_1 et A_2 d'une classe A peuvent utiliser toutes deux une propriété P, définie comme applicable pour A, quand une troisième sous-classe A_3 peut ne pas l'utiliser. Utiliser l'héritage, comme cela est fait dans la représentation verticale usuelle dans les bases de données orientées objet (où chaque propriété est représentée dans la table correspondant au niveau où la propriété est définie) s'avère peu adaptée. Cela reviendrait, en effet, à représenter également la propriété P pour la classe A_3.

Nous proposons d'opter pour une technique de représentation se basant sur l'approche de *représentation horizontale*. Dans cette approche :

- chaque classe qui se trouve être une classe de base d'au moins une instance est associée à une table,
- le schéma de cette table comporte toutes les propriétés applicables de la classe qui sont utilisées par *au moins une des instances de la classe*.

Notre approche est donc caractérisée par les points suivants.

- **Tables pour les classes.** On ne crée des tables que pour un sous-ensemble de classes de l'ontologie : celles associées à des instances.
- **Colonnes pour les propriétés.** On ne crée des colonnes que pour un sous-ensemble de propriétés applicables des classes : celles utilisées par au moins une instance de la classe.

Le choix des classes et des propriétés est normalement fait par le concepteur de la base de données. Il sélectionne dans l'ontologie les classes pour lesquelles il souhaite créer des instances, ainsi que les propriétés à initialiser pour chaque table de classe. L'ensemble des classes et des propriétés sélectionnées forme le schéma de la partie donnée de la base de données et constitue donc son modèle conceptuel.

Lorsque la BDBO est destinée à charger automatiquement le contenu d'une autre BDBO basée sur

(a)

Objet1:student
noSS:string=02489 name:string=Woopy lastname:string=Golder age:int=35 gender:char=F

Objet2:employee
noSS:string=13457 name:string=Jean lastname:string=Claude age:int=40 salary:double=1500 service:string=CRCFAO

(b)

id_student					
OID	**id_noSS**	**id_name**	**id_lastname**	**id_age**	**id_gender**
100	02489	Woopy	Golber	35	F

id_employee						
OID	**id_noSS**	**id_name**	**id_lastname**	**id_age**	**id_salary**	**id_service**
103	13457	Jean	Claude	40	1500	CRCFAO

FIG. 2.2 – (a) Exemple d'une population d'instances de classes et (b) leur représentation dans la table de leur classe.

la même ontologie normalisée, le schéma peut également être défini dynamiquement en fonction des classes et des propriétés effectivement référencées dans les données à base ontologiques lues.

2.5.3 Besoin de représentation du modèle conceptuel des données

Supposons la situation suivante:

1. Lors d'une requête de type polymorphe c'est-à-dire, réalisée au niveau d'une classe non feuille, comment pourrait-on identifier les tables des classes à interroger ?

   Exemple. Retrouver toutes les instances de la classe *Person*. Comment savoir que seules les tables des classes *Student* et *Employee* sont concernées par la requête et que celle-ci doit se traduire sous la forme :

   SELECT ID **FROM** Student
   UNION
   SELECT ID **FROM** Employee

2. si l'on admet que l'utilisateur doit pouvoir faire des requêtes sur les propriétés applicables d'une classe, comment, lors d'une *requête polymorphe* au niveau d'une classe non feuille, peut-on savoir qu'une propriété est utilisée ou non pour certaines sous-classes ?

   Exemple. Retrouver le *nom*, l'*âge* et le *sexe* (*gender*) de toutes les instances de la classe *Person* (y compris celles de ses sous-classes). Comment savoir que la table *Employee* ne définit pas la propriété *gender* et que la requête doit s'écrire :

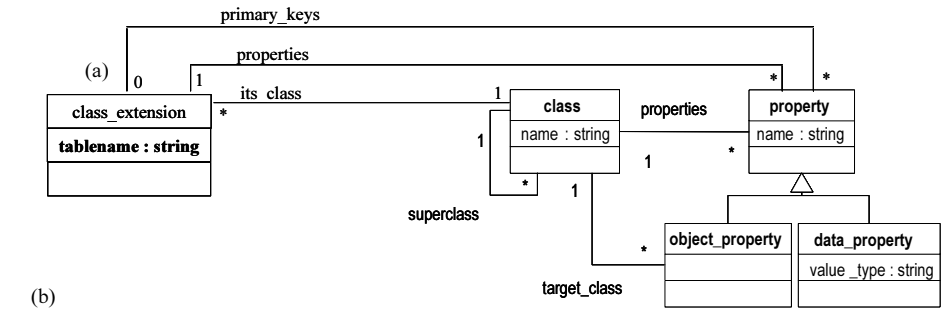
   SELECT ID, name, age, gender **FROM** Student
   UNION
   SELECT ID, name, age, NULL as gender **FROM** Employee

Les problèmes évoqués ci-dessus, résultent de l'absence d'information concernant le schéma des instances des classes d'ontologie nécessaire pour la génération des requêtes. Deux solutions sont possibles pour accéder au schéma des instances.

– **Par l'accès à la méta-base** de la base de données. La *méta-base* est la partie traditionnelle des bases de données dans laquelle sont stockées des méta-données utiles pour le SGBD pour la gestion de données. Elle contient, entre autres, une table des tables et une table pour les colonnes des tables de la base de données. Ces tables peuvent être exploitées pour l'identification des tables et des colonnes impliquées dans les requêtes précédentes.
L'inconvénient de cette solution est que la structure de la *méta-base* n'est pas la même d'un fournisseur de bases de données à un autre. Par exemple, en PostgreSQL, la table des tables est nommée *pg_table* et en SQL Server *sysobjects*. Du point de vue portabilité, l'utilisation de la *méta-base* n'est donc pas la solution idéale.

– **Par la représentation du modèle conceptuel des instances** dans la base de données. A défaut d'utiliser la *méta-base*, on peut représenter le modèle conceptuel des instances des différentes classes. La figure 2.3a, présente un diagramme de classes UML décrivant un modèle simplifié de schéma des instances des classes. Dans cette représentation, on peut remarquer que chaque extension de classe référence sa classe à travers l'attribut *its_class*. L'ensemble des propriétés initialisées par la classe est donné par la relation *properties*. La relation *primarykeys* permet de spécifier les propriétés de la classe qui formeront la clé qui permettront d'identifier une instance. La figure 2.3b montre un exemple de schéma de base de données issu du diagramme UML.

  Notre proposition est de représenter effectivement le modèle conceptuel des instances, ce qui permettra en plus, de *garder la trace de l'historique des différents schémas d'une même classe*.



(b)

class_extension		
ID	**its_class**	**tablename**
1	id_person	id_person
2	id_employee	id_employee

Primary keys	
IDClass_Extension	**IDProperty**
1	id_noSS
2	id_noSS

properties	
IDClass_extension	**IDProperty**
1	id_noSS
1	id_name
1	id_lastname
1	id_age
1	id_gender
2	id_noSS
2	id_name
2	id_lastname
2	id_age
2	Id_salary
2	id_service

FIG. 2.3 – Exemple de représentation du modèle conceptuel des instances des classes de la base de données

2.5.4 Relations entre la partie ontologie et la partie données

2.5.4.1 Position du problème

Les données à base ontologique et les ontologies étant gérées séparément, il est nécessaire d'établir deux mécanismes.

(1) Un mécanisme de *liaison bilatère entre les deux parties* doit permettre d'accéder aux données d'une partie via l'autre. L'un des principaux objectifs des BDBOs étant de représenter explicitement la sémantique des données à travers l'ontologie, ce mécanisme doit permettre à la fois d'interroger les données à partir de l'ontologie et de présenter les données en termes de l'ontologie.

(2) Un mécanisme de cohérence doit permettre d'assurer la cohérence et l'intégrité des données des instances. Des contraintes essentielles sur les données sont en effet représentées au niveau de l'ontologie. Il est donc nécessaire d'assurer le respect de ces contraintes par les données dans la base de données. Par exemple, supposons qu'au niveau de la classe *Student* de l'ontologie, nous ayons défini une contrainte spécifiant que l'*âge* des étudiants doit être compris entre *16* et *100* ans, il convient que parmi la population d'instances de la classe, il n'y ait pas d'instances violant cette contrainte. D'autres contraintes de ce type doivent également être vérifiées comme l'unicité des valeurs de propriétés, ou encore les contraintes d'intégrités référentielles.

2.5.4.2 Notre proposition

Nous proposons de représenter le mécanisme de liaison bilatère entre les ontologies et leurs instances par l'intermédiaire de deux fonctions partielles :

- **Nomination :** classe ∪ propriété → table ∪ attribut ;
- **Abstraction :** table ∪ attribut → classe ∪ propriété ;

La fonction de *Nomination* associe à tout concept de niveau ontologique les éléments (table / colonne) qui en représentent les instances. La fonction d'*Abstraction* associe à tout élément de données le concept de l'ontologie qui en définit le sens. Ces deux fonctions sont partielles car :

- certaines classes et / ou propriétés peuvent ne pas être représentées, seul un sous-ensemble des classes et des propriétés est sélectionné par le concepteur de la base de données pour constituer son modèle conceptuel ;
- certaines tables et / ou attributs, de noms prédéfinis, correspondent à des informations de type système et non à des éléments ontologiques, c'est le cas par exemple : (1) des tables qui permettent de représenter le schéma des instances des classes (cf. figure 2.3); (2) des colonnes $version_{min}$, $version_{max}$ dans les tables des instances des classes qui permettent de gérer le cycle de vie des instances de la base de données (pour plus de détails cf. Chapitre 4). La propriété $version_{min}$ indique la version de la population des instances à partir de laquelle l'instance a commencé d'exister. La propriété $version_{max}$ indique le numéro de la dernière version de la population à partir de laquelle elle a été supprimée. Lorsqu'une instance est valide dans la dernière version en cours, sa $version_{max}$ vaut NULL.

2.6 Notre proposition de modèle d'architecture de BDBO : OntoDB

Nous pouvons maintenant synthétiser l'ensemble des propositions présentées dans ce chapitre sous forme d'un modèle d'architecture pour les BDBOs, que nous avons conçu pour répondre aux besoins que nous avons identifié et que nous avions proposé dans le cadre de la thèse d'Hondjack DEHAINSALA [80]. Cette architecture répond aux objectifs O_1 - O_4 que nous nous sommes fixés.

Cette architecture vise à faire la synthèse de deux univers.

– L'univers des bases de données, où les données sont associées à des schémas de données qui permettent de traiter de façon efficace de très gros volume de données.
– L'univers des données à base ontologique, issus des concepts du Web sémantique, qui permettent de réunir, au sein d'un même système, des données, appelées individus, et des ontologies qui en donnent le sens. Dans ces systèmes, chaque individu est porteur de sa propre structure et la notion de schéma de données au sens usuel des bases de données n'est pas présent : chaque individu est décrit par des ensembles de triplets spécifiques.

L'idée centrale de notre synthèse est d'introduire dans les BDBOs le niveau schéma de données qui constitue la structure commune, à travers des hypothèses de typage fort, et, éventuellement, au prix d'un certain nombre de valeurs nulles, de l'ensemble des instances d'une classe de l'ontologie. Cette idée s'applique très naturellement au premier domaine d'application dont nous sommes portés : les catalogues de composants industriels. Après une étude des autres domaines, elle semble en fait également applicable à beaucoup d'autres domaines tels que l'annotation des documents ou les catalogues des portails du Web sémantique [3].

La figure 2.4 présente les quatre parties qui composent notre proposition d'architecture, appelée *OntoDB*, pour les bases de données à base ontologique :

1. La **partie *méta-base*.** La *méta-base*, souvent appelée *system catalog*, est une partie traditionnelle des bases de données classiques. Elle est constituée de l'ensemble des tables système. Ces tables sont celles dont le SGBD se sert pour gérer et assurer le fonctionnement de l'ensemble des données contenues dans la base de données. Dans une BDBO, toutes les tables et les attributs définis dans les trois autres parties sont documentés dans la *méta-base*.
2. La **partie *données*.** Elle représente les objets du domaine. Ceux-ci sont décrits en termes d'une classe d'appartenance et d'un ensemble de valeurs de propriétés applicables à cette classe. C'est ce que nous appelons les *données à base ontologique*. Les propriétés des classes effectivement représentées (dites *propriétés valuées*) sont celles qu'au moins une instance de la classe initialise. Le schéma de la partie données est soit défini par l'utilisateur à travers une interface graphique, soit généré automatiquement par le système lorsqu'une instance, dont l'ontologie est connue, est lue. Une autre spécificité de la partie *données* d'*OntoDB* est de permettre de gérer le cycle de vie des objets du domaine.
3. La **partie *ontologie*.** Elle contient les ontologies définissant la sémantique des différents domaines couverts par la base de données, ainsi, qu'éventuellement l'articulation, représentée par des relations subsomption, de ces ontologies locales avec des ontologies externes (par exemple normalisées). Les ontologies représentées dans cette partie peuvent être versionnées et archivées. L'accès aux données de cette partie est possible grâce à deux types d'interface de programmation : une *API à liaison tardive* et deux APIs *à liaison préalable*. Toutes ces APIs permettent l'accès aux

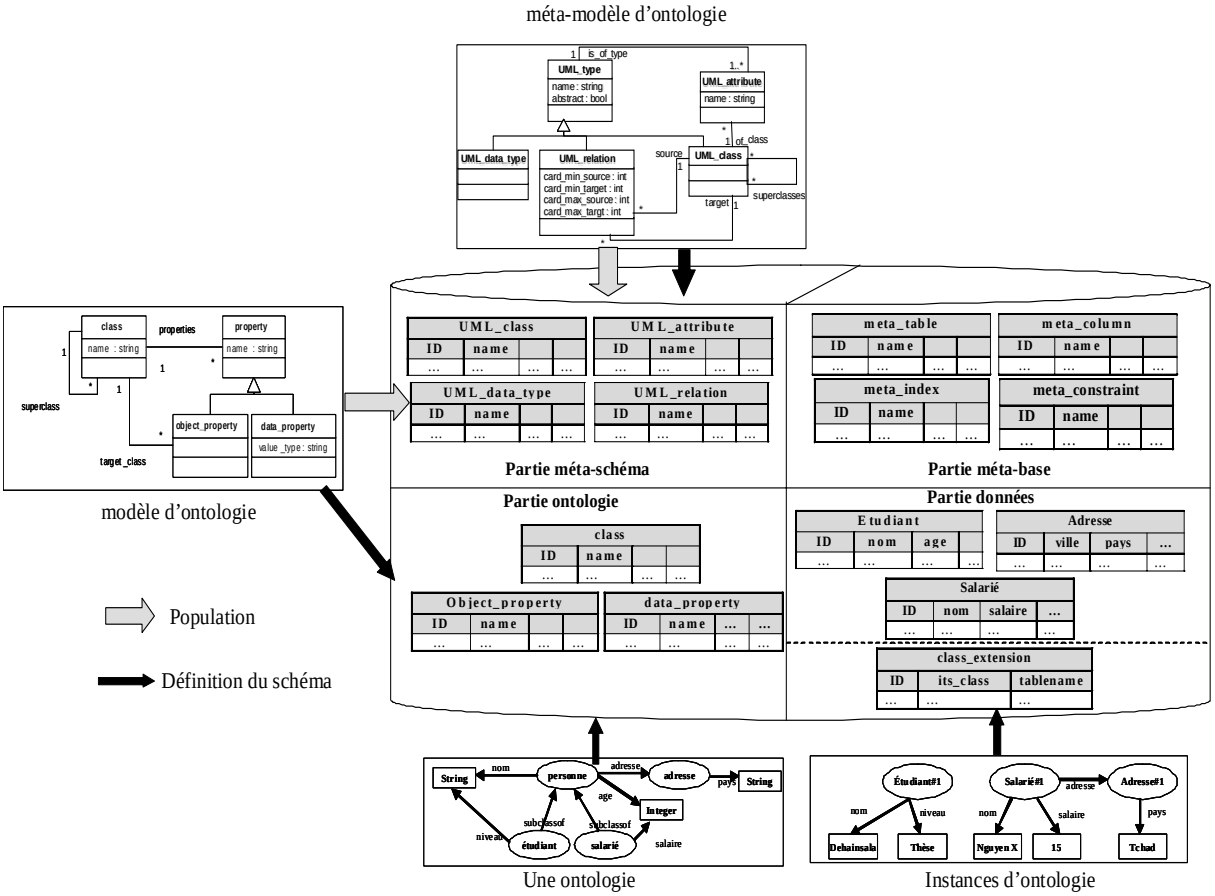


FIG. 2.4 – Architecture OntoDB

concepts des ontologies, et ce, en différentes versions.

4. La **partie *méta-schéma*.** Elle représente, au sein d'un modèle réflexif, à la fois le modèle d'ontologie utilisé et le méta-schéma lui-même. Le *méta-schéma* est pour la partie *ontologie*, ce qui est la *méta-base* pour le système. Il permet, au moyen d'une API qui exploite son contenu, de gérer de façon générique les concepts des ontologies représentés dans la partie *ontologie*. Le schéma de données de cette dernière étant susceptible d'évoluer ou d'être modifié, l'autoreprésentation du méta-modèle du modèle d'ontologie permet de rendre certains traitements génériques et/ou indépendamment du modèle d'ontologie utilisé.

2.7 Evaluation de performances

Nous avons effectué une évaluation de performance de notre architecture de base de données à base ontologique. Notre évaluation s'est principalement portée sur les choix fait pour la représentation des données à base ontologique dans la partie *données* de notre architecture. En effet, il existe dans ce domaine d'autres propositions avec lesquelles il convenait de se comparer. Peu de comparaisons ont par contre été effectuées sur les ontologies car les schémas d'implantation des ontologies dépendent étroitement du modèle d'ontologie ciblé, et, de ce point de vue là, les différents systèmes disponibles ciblent des modèles différents.

Pour notre évaluation de la partie *données*, nous avons comparé les deux approches préexistantes connues pour être les plus efficaces (*table par propriété* et *triplets*) à l'approche *table par classe* proposée dans le cadre de notre thèse. Des études antérieures ayant montré que l'approche *table universelle* n'était en effet pas compétitive.

Nos tests ont porté sur trois familles de requêtes : (1) les requêtes typées dans lesquelles l'utilisateur connaît la ou les classes dans lesquelles il effectue sa recherche, (2) les requêtes non typées dans lesquelles l'utilisateur recherche sur l'ensemble de la population à la base de données, et (3) les requêtes de modification. Ces requêtes ont été réalisées sur différentes bases de données que nous avons générées à partir d'exemples réels appartenant à notre domaine d'application cible. Pour cela nous avons développé un générateur, qui à partir d'une ontologie fournie en paramètre, génère des populations d'instances pour les différentes classes de l'ontologie.

Les tests des requêtes typées (sélection, projection, et jointure sur des classes feuilles ou non) ont montré que l'approche *table par classe (TL)* était toujours plus performante que l'approche *table par propriété (TP)* elle-même plus performante que l'approche table unique de type *triplets* (TU). Le résultat le plus marquant est que les temps d'exécution des requêtes typées portant sur l'approche *TC* sont relativement stables aussi bien lorsqu'on fait varier le nombre de propriétés impliquées dans les requêtes (propriétés projetées ou dans le prédicat de sélection) que lorsqu'on fait varier le nombre d'instances des classes. Dans l'approche *table par propriété*, par contre ces temps croissent très fortement, de sorte que l'approche *TC* supporte beaucoup mieux le passage à l'échelle que l'approche *TP*.

Les tests ont été effectués sur trois ontologies de différentes tailles :
– l'ontologie normalisée *IEC 61360-4* composée de 190 classes et 1026 propriétés. La moyenne des classes des hiérarchies est de 5,
– l'ontologie *LMPR* définie par RENAULT pour sa propre base de données de composants est consti-

tuée de 295 classes et 509 propriétés. La profondeur de la hiérarchie des classes varie de 2 à 5,
– l'ontologie *fixation* normalisée (ISO 13584-501) composée de 36 classes et de 23 propriétés et la moyenne est de 3.

Les tests sur les *requêtes non typées* ont montré un résultat plus nuancé. L'approche *table par propriété* est plus performante tant que très peu de propriétés sont concernées par les requêtes (1 à 3). Dès que le nombre de propriétés utilisées soit dans la sélection soit dans la jointure atteint un nombre de l'ordre de 4, les deux approches font jeu égal, puis l'approche *TC* devient de plus en plus préférable.

Enfin les tests des requêtes de modification ont montré que l'approche *table par classe* était toujours plus performante que l'approche *table par propriété*. Ils ont également montré que l'approche *triplet* était pratiquement inutilisable dès lors que les mises à jours intervenaient. En effet, la table unique demandait alors à être triée (clusterisée) par valeur de prédicat afin de garder des performances acceptables en recherche. Et un tel tri exige toujours des temps rédhibitoires (plus de 3 minutes pour nos plus petites bases de données).

De l'ensemble de ces tests, il résulte clairement que l'approche *table par classe* que nous avons proposée est pratiquement toujours plus efficace, voire beaucoup plus efficace pour les bases de données de taille significative, que les approches pré-existantes. Sa réelle limite réside dans les hypothèses que nous avons définies pour la mettre en œuvre, à savoir le *typage fort* des propriétés et la mono-instanciation des classes. Le typage fort des propriétés est une hypothèse également faite par beaucoup d'autres chercheurs. Elle ne nous parait guère limitative. La *mono-instanciation* par contre peut faire débat, car elle impose un style de modélisation où la *multi-instanciation* est représentée soit par l'héritage multiple, soit par la technique de l'agrégat d'instances. Quoi qu'il en soit, dans beaucoup de domaines, et en particulier dans notre domaine cible, la *multi-instanciation* ne fait pas partie des techniques de modélisation utilisées.

Pour notre évaluation de la partie *ontologie*, nous avons, au moyen d'ontologies de tailles différentes, mesuré le temps de navigation à travers l'ontologie dans l'outil PLIBEditor. Les tests nous ont révélé des temps de réponse acceptables pour des ontologies de taille usuelle. Pour les grandes ontologies, les temps de réponse étaient assez considérables. Cette médiocre performance pour les grandes ontologies est due à l'approche de traduction systématique du modèle objet en modèle logique que nous avons adoptée. En effet celle-ci engendre de nombreuses tables, causant ainsi de nombreuses jointures pour la récupération des ontologies. Pour pallier à ce défaut, une nouvelle approche basée sur la transformation de modèle a été développée dans notre laboratoire. Elle devrait être implantée dans une prochaine version d'*OntoDB*.

2.8 Conclusion

Dans la thèse d'Hondjack DEHAINSALA, nous avons présenté un nouveau modèle d'architecture pour les BDBOs appelé *OntoDB*. Le modèle d'architecture *OntoDB* se compose en quatre parties qui sont les parties *Ontologie*, *Méta-schéma*, *Données* et *Méta-base*. Les trois premières parties servent respectivement à représenter les ontologies, les modèles d'ontologies, les instances des ontologies. La dernière partie, encore appelée *system catalog*, est la partie des SGBDs qui contient la description des tables et des vues de la base de données. L'architecture *OntoDB* se distingue par rapport aux autres architectures sur les points suivants.

- **La représentation des données à base ontologique.** Les architectures existantes dans la littérature représentent toutes les instances, soit sous forme de triplets dans une table à trois colonnes (sujet, prédicat, objet), soit dans des tables binaires (id, valeur) pour chacune des propriétés des classes. Dans *OntoDB*, l'hypothèse de *typage fort* des propriétés, et de *mono-instanciation* permet d'associer à chaque classe un schéma de données constitué d'un *sous-ensemble* des propriétés applicables de la classe. Les instances des classes sont alors représentées dans *OntoDB* au sein d'une relation unique représentée dans des tables spécifiques créées pour chaque classe de la base.
  Cette approche de représentation des instances impose deux restrictions : mono instanciation et type fort des propriétés. La dernière hypothèse est très souvent utilisée dans d'autres systèmes [3]. Concernant la première hypothèse, d'une part elle est toujours effectuée dans le domaine des bases de données, et, d'autre part, elle n'interdit pas qu'un objet du monde réel soit décrit par plusieurs instances de classes. On peut en effet utiliser la technique dite de l'agrégat d'instances qui permet de représenter la même information que la technique de multi-instanciation. Cette représentation est d'ailleurs autorisée par la plupart des modèles d'ontologies, y compris le modèle PLIB qui nous intéresse particulièrement. Ces restrictions permettent par contre de gérer de façon efficace de très grands ensembles de données à base ontologie ce que ne permettait pas les méthodes antérieures [82].
- **Représentation du modèle conceptuel des données dans la base de données.** L'architecture *OntoDB* permet de représenter explicitement le modèle conceptuel des données dans la base de données. Le modèle conceptuel est un sous-ensemble cohérent des concepts des ontologies sélectionnés par le concepteur. Il définit à la fois la *structure* et par sa référence à l'ontologie, la *sémantique* des données à base ontologique.

- **Versionnement des concepts des ontologies.** L'architecture *OntoDB* permet d'archiver et de manipuler les différentes versions des concepts des ontologies. La technique utilisée pour sa mise en œuvre nous a conduit à définir la notion de version courante. Cette approche représente la dernière version disponible de chaque concept et permet de consolider toutes les relations entre les différents concepts. Les *versions historiques* des concepts sont archivées dans la base de données et leurs relations avec les autres concepts sont maintenues pour conserver toute la sémantique des différentes versions des ontologies.

- **Versionnement des instances et gestion du cycle de vie des instances.** Comme les ontologies, les instances des classes sont aussi versionnées dans la base de données lorsqu'il y a des changements. Pour ce faire, chaque instance de la base de données est associée à une liste de numéros de version de la classe à laquelle elle appartient. Son modèle conceptuel est aussi également historisé.

- **Généricité par rapport aux modèles d'ontologies.** *OntoDB* a été défini de façon à supporter toute évolution et/ou changement du modèle d'ontologie utilisé. La réalisation de cette caractéristique, nous a conduit (1) à représenter dans la base de données au sein d'un méta-modèle réflexif à la fois les modèles d'ontologies ainsi que le méta-modèle de ces modèles ontologies et (2) à proposer une interface d'accès à liaison tardive pour accéder aux concepts de ces ontologies, indépendant donc de tout modèle particulier d'ontologie.

Notons que ce modèle d'architecture est un modèle logique qui n'est pas nécessairement implanté sur un système unique.

- Partie *ontologie* et partie *données* peuvent éventuellement être représentées dans deux systèmes différents, coordonnés par une interface d'accès commune. Nous avons également réalisé un premier prototype selon cette architecture.
- Si le niveau *méta-schéma* est nécessaire en génération, ce niveau n'est réellement indispensable que lors d'un changement de modèle d'ontologie. On peut donc éventuellement effectuer les générations correspondant aux fonctions F_1, F_2, F_3 et F_4 à l'aide d'un système externe dans lequel la partie *méta-schéma* serait représentée.

On peut constater que notre architecture de BDBO a de fortes similitudes avec l'architecture metadata du MOF (Meta Object Facility) [104]. En effet, notre architecture est constituée des même quatre couches superposées. La couche modèle M_1 de l'architecture MOF correspond à notre modèle conceptuel, sous-ensemble de l'ontologie. Ce niveau contient lui-même les instances M_0. La couche méta-modèle M_2 correspond au (méta-) modèle d'ontologie, la couche méta-méta-modèle M_3 (MOF model) correspond au méta-modèle, lui-même réflexif, du langage de définition du modèle d'ontologie.

Notons enfin que notre architecture permet également (1) d'exporter, (2) d'importer des ensembles d'instances de classes d'une ontologie existante et de calculer automatiquement leur schéma de représentation.

Chapitre 3

Intégration de sources de données par articulation a priori d'ontologies

Publications : [38, 40, 39, 221, 89, 88].

Dans le chapitre précédent, nous avons proposé une démarche pour préparer des sources de données susceptibles de participer à un processus d'intégration. Cette préparation génère des sources de données à base ontologique. Chaque source contient alors à la fois sa propre ontologie et les relations sémantiques qui l'articulent a priori avec une ou plusieurs ontologies partagées. Dans ce chapitre, nous présentons nos travaux d'intégration qui rentrent dans le cadre de la thèse de doctorat de Dung Nguyen Xuan présentée au sein du laboratoire LISI, dans le cadre du projet OntoDB, co-encadrées par Pr. Guy PIERRA, soutenue en 2006. La particularité de cette approche est qu'elle laisse à chacune des sources une autonomie significative tant au niveau de sa structure qu'au niveau de son évolution, et qu'elle permet néanmoins une intégration automatique.

3.1 Architecture du système d'intégration de BDBOs

Notre architecture d'intégration a, en entrée, un ensemble de *BDBO* référençant *a priori* une ontologie partagée, et vise à produire, en sortie, un entrepôt ayant lui même la structure d'une BDBO. Cette architecture définit plusieurs opérateurs d'intégration correspondant à différents scénarii possibles qui dotent l'ensemble des BDBOs d'une structure d'algèbre. Cette architecture est illustrée dans la figure 3.1.

3.1.1 Principe d'engagement sur une ontologie de référence

Une ontologie est dite "partagée" entre plusieurs sources, lorsque les sources s'engagent sur les concepts qu'elle définit et *sur le fait d'utiliser* les définitions ontologiques de l'ontologie partagée qui ont été acceptées et éventuellement été normalisées chaque fois que cela est possible. Afin de garder son autonomie, cette source peut définir sa propre hiérarchie de classes, et, si besoin est, rajouter les propriétés qui n'existent pas dans l'ontologie partagée.

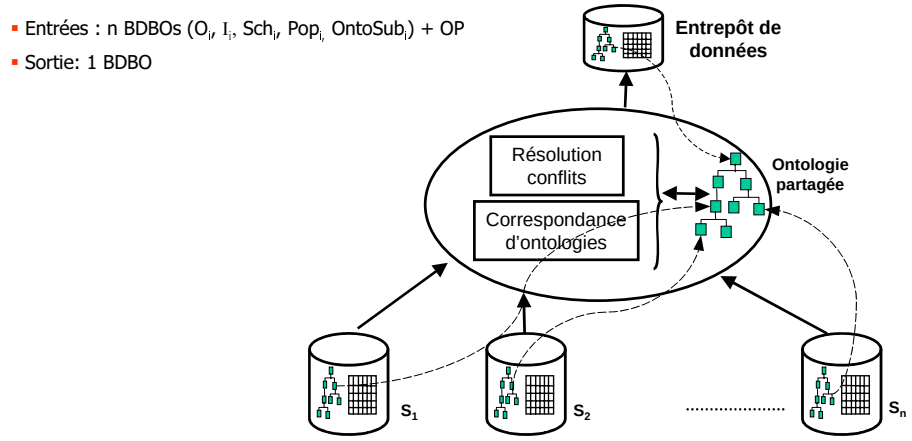


FIG. 3.1 – Système d'intégration des bases de données à base ontologique par articulation *a priori* d'ontologies: algèbre de composition.

Plus précisément, s'engager sur une ontologie partagée signifie respecter la double contrainte suivante (appelée SSCR: *Smallest Subsuming Class Reference* [39]):

- toute classe locale doit référencer, par la relation $OntoSub$, la plus petite classe subsumante existante dans la hiérarchie de référence si ce n'est pas la même que celle de sa propre super classe;
- toute propriété nécessaire à l'ontologie locale et existant dans l'ontologie de référence doit être importée à travers la relation $OntoSub$.

Si une ontologie locale O_i est articulée avec l'ontologie partagée O_p en respectant le principe $SSCR$, nous disons qu'elle "*référence autant que cela est possible*" l'ontologie O_p.

Nous avons formalisé l'articulation entre une $BDBO_i$ et l'ontologie O_p comme un triplet : $\mathcal{A}_{i,p} :< BDBO_i, O_p, OntoSub_{i,p} >$, où

- $BDBO_i :< O_i, I_i, Pop_i, Sch_i >$ pour représenter une base de données à base ontologique, dont $O_i :< C_i, P_i, Sub_i, Applic_i >$ (voir 2.1.3.2) est l'ontologie locale
- $O_p :< C_p, P_p, Sub_p, Applic_p >$ représente l'ontologie partagée portant sur le même univers du discours que $BDBO_i$,
- $OntoSub_{i,p} : C_p \rightarrow 2^{C_i}$ représente les relations de subsomption entre O_p et O_i qui associent à chaque classe c_p de C_p l'ensemble des classes $c_i \in C_i$ qui sont subsumés directement par c_p: $\forall c_p \in C_p, OntoSub_{i,p}(c_p) = \{c_i \in C_i | (c_p \text{ subsume } c_i) \wedge (\forall c'_i | c_i \in Sub_i(c'_i) \Rightarrow c'_i \notin OntoSub_{i,p}(c_p)) \wedge (\forall c'_p \in Sub_p(c_p) \Rightarrow c_i \notin OntoSub_{i,p}(c'_p))\}$.

3.1.2 Scénarii d'intégration de données

Soit $S = \{S_1, S_2, ..., S_n\}$ l'ensemble des sources de données participant au processus d'intégration. Chaque source S_i est définie comme suit: $S_i :< O_i, I_i, Sch_i, Pop_i >$. Notons que dans une BDBO, tout élément représenté dans le schéma, classe ou propriété doit appartenir à l'ontologie, de sorte que le schéma est un sous-ensemble de l'ontologie, chaque entité représentée correspondant à une classe et ses attributs correspondant aux propriétés applicables choisies. On fait abstraction ici du découpage éventuel résultant des opérations de normalisation, une vue étant, dans tous les cas, créée pour représenter la

population de chaque classe.

Pour simplifier la présentation, nous avons supposé désormais que seules les classes feuilles sont choisies comme classes de base et sont directement instanciables. Les classes non feuilles sont supposées "abstraites", c'est-à-dire que leur population est l'union des populations de leurs sous-classes.

Dans la méthode d'intégration par articulation *a priori* d'ontologie, nous avons supposé que l'ontologie partagée O_p préexiste à la définition de la source $BDBO_i$. Notons que cette hypothèse est toujours faite lorsque l'on annote des ressources avec les méta-données existantes : la base d'annotation obéit à cette hypothèse. La source $BDBO_i$ est conçue en six étapes :

1. l'administrateur choisit la hiérarchie de classes (C_i, Sub_i) de sa propre ontologie O_i.
2. Il articule cette hiérarchie de classes avec celle de l'ontologie partagée C_p en définissant les relations de subsomption $OntoSub_{i,p}$ entre C_i et C_p.
3. A travers les relations de subsomption $OntoSub_{i,p}$, le DBA importe dans $Applic_i(c_i)$ les propriétés de $Applic_p(OntoSub_{i,p}^{-1}(c_i)) \subset P_p$ qu'il souhaite utiliser dans sa propre ontologie. Ces propriétés appartiennent alors à P_i.
4. l'administrateur complète éventuellement les propriétés importées par des propriétés supplémentaires, propres à son ontologie définissant ainsi l'ontologie locale:
   $O_i :< C_i, P_i, Sub_i, Applic_i >$.
5. l'administrateur de chaque source choisit pour chaque classe feuille les propriétés qui seront valuées en définissant $Sch_i : C_i \rightarrow 2^{P_i}$, et
6. l'administrateur choisit une implémentation de chaque classe feuille (e.g., afin d'assurer la troisième forme normale), et il définit ensuite $Sch(c_i)$ comme une vue sur l'implémentation de c_i.

Dans ce cas, le schéma de chaque classe feuille est explicitement défini, et celui d'une classe non feuille est calculé comme étant l'intersection entre les propriétés applicables de c_j et l'intersection des ensembles de propriétés associées à des valeurs dans toutes les sous classes $c_{i,j}$ de c_j:

$$Sch(c_j) = Applic(c_j) \cap (\cap_i Sch(c_{i,j})) \tag{3.1}$$

Une définition alternative qui nous semble préférable peut également être utilisée pour créer le schéma d'une classe non feuille. Elle consiste à prendre l'union des propriétés existant dans au moins une sous-classe, et en complétant par des valeurs nulles :

$$Sch'(c_j) = Applic(c_j) \cap (\cup_i Sch(c_{i,j})) \tag{3.2}$$

Le choix entre ces deux représentations doit être laissé à l'utilisateur.

Plusieurs scénarii d'intégration, correspondant à différentes articulations entre les ontologies locales et l'ontologie partagée du domaine ont été décrit dans la thèse de Dung. Nous présentons trois scénarios [38, 39, 40] :

1. **FragmentOnto** : dans ce scénario, on suppose que les ontologies locales des bases de données sont directement extraites de l'ontologie partagée (chaque ontologie locale est un sous ensemble de l'ontologie partagée).

2. **ProjOnto** : dans ce scénario, chaque source définit sa propre ontologie (elle n'instancie aucune classe de l'ontologie partagée). Par contre, chaque ontologie locale référence l'ontologie partagée en respectant la condition SSCR. On souhaite intégrer les instances de chaque source qui appartiennent, par subsomption, à une classe de l'ontologie partagée, comme des instances de l'ontologie partagée;
3. **ExtendOnto** : chaque ontologie locale est définie comme dans le scénario ProjOnto, mais l'on souhaite enrichir automatiquement l'ontologie partagée. Ensuite toutes les instances de données sont intégrées, sans aucune modification, au sein du système intégré.

Par la suite, nous décrivons le contexte d'étude de chaque scénario, l'algorithme d'intégration et enfin son application au domaine des composants industriels.

3.1.3 FragmentOnto

3.1.3.1 Contexte

Ce scénario d'intégration suppose que l'ontologie partagée est suffisante pour couvrir toutes les sources locales. Une hypothèse de ce type a déjà été utilisée. Nous pouvons ainsi citer, par exemple les projets Picsel2 [185] et COIN [100]. Dans ce cas, l'autonomie des sources se limite à (1) sélectionner un sous ensemble pertinent de l'ontologie partagée (classes et propriétés) et (2) concevoir le schéma local de la base de données.

L'ontologie O_i de chaque source S_i ($1 \leq i \leq n$) étant un fragment de l'ontologie partagée O_p. Elle se définit comme le quadruplet $O_i :< C_i, P_i, Sub_i, Applic_i >$, avec :

- $C_i \subseteq C_p$;
- $P_i \subseteq P_p$;
- $\forall c \in C_i, Sub_i(c) \subseteq Sub_p(c)$;
- $\forall c \in C_i, Applic_i(c) \subseteq Applic_p(c)$.

Pour intégrer ces sources au sein d'une BDBO il suffit de trouver l'ontologie, le schéma et la population du système intégré. Le système intégré est donc défini comme $Int :< O_{Int}, Sch_{Int}, Pop_{Int} >$. Maintenant, il s'agit de calculer la structure de chaque élément de Int. Ce calcul est présenté dans la section suivante.

3.1.3.2 Algorithme

O_{Int}. Dans le cas d'une intégration par $FragmentOnto$: $\forall i : O_i \subset O_p$. Nous pouvons donc utiliser l'ontologie partagée comme l'ontologie du système intégré:

$$O_{Int} = O_p \tag{3.3}$$

Cette définition assure que O_{Int} couvre toutes les sources.
Sch_{Int}. Le schéma du système intégré est défini pour chaque classe comme suit (l'intégration par Intersection):

$$Sch_{Int}(c) = \Big(\bigcap_{i \in 1..n | Sch_i(c) \neq \phi} Sch_i(c) \Big) \tag{3.4}$$

Cette définition assure que les instances du système intégré ne seront pas complétées par des valeurs nulles. Pour chaque classe, seules les propriétés valuées dans toutes les sources de données seront préservées. Si dans certaines sources on trouve des classes vides, elles ne seront pas prises en compte pour calculer les propriétés fournies par toutes les sources.

Le schéma du système intégré peut être également défini comme suit (intégration par Union):

$$Sch'_{Int}(c) = (\bigcup_{i \in 1..n | Sch_i(c) \neq \phi} Sch_i(c)) \tag{3.5}$$

Pour le deuxième calcul du schéma intégré, les instances du système intégré seront complétées par des valeurs nulles. A la différence du cas précédent, pour chaque classe, toutes les propriétés valuées dans au moins une source seront préservées.

Pop_{Int}. La population de chaque classe du système intégré est définie comme suit:

$$Pop_{Int}(c) = \bigcup_{i} Pop_i(c) \tag{3.6}$$

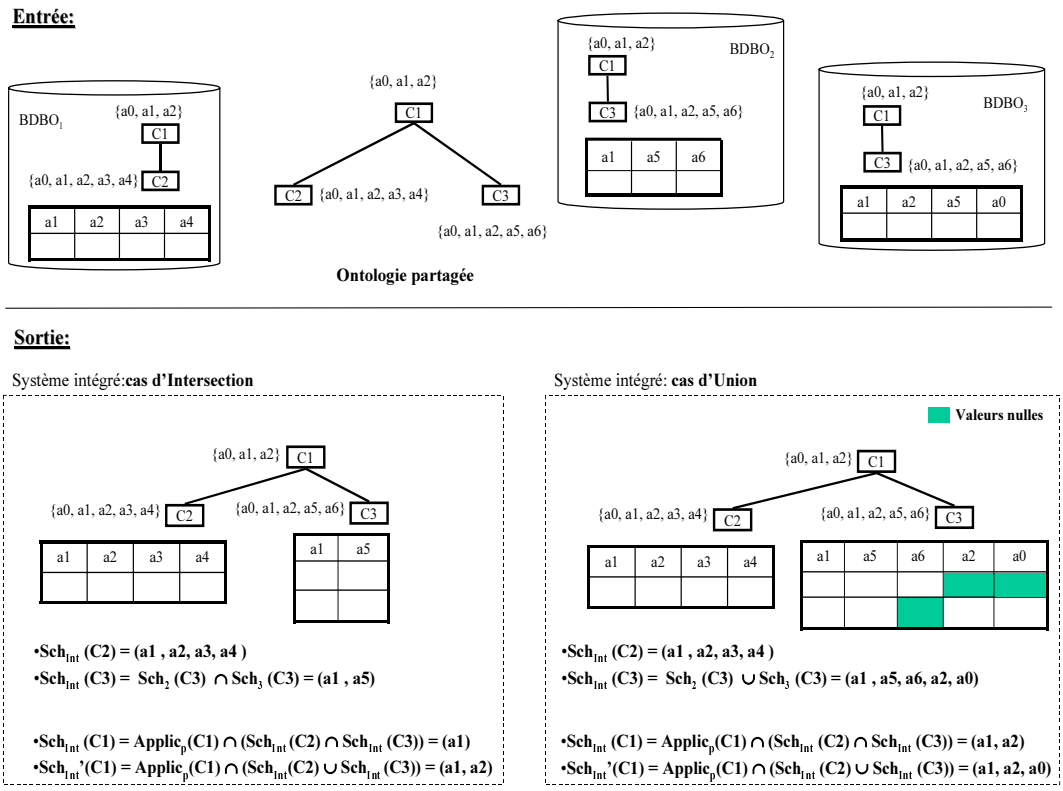


FIG. 3.2 – Exemple d'une intégration de BDBOs par *FragmentOnto*

Pour illustrer l'algorithme d'intégration par *FragmentOnto*, considérons l'exemple suivant.

Exemple. : Supposons que l'on ait trois sources de données à base ontologique référençant une ontologie partagée comme dans la figure 3.2.

La source $BDBO_1$ utilise la branche $\{C_1, C_2\}$ (un fragment de l'ontologie partagée) comme son ontologie locale. Puis la classe feuille C_2 est choisie comme la classe de base des instances de $BDBO_1$,

avec $Sch_1(C_2) = \{a_1, a_2, a_3, a_4\}$. Quant aux sources $BDBO_2$ et $BDBO_3$, la branche $\{C_1, C_2\}$ est utilisée comme leur ontologie locale. Pour la $BDBO_2$, le schéma physique des instances de données est: $Sch_2(C_3) = \{a_1, a_5, a_6\}$. Et pour la $BDBO_3$: $Sch_3(C_3) = \{a_1, a_2, a_5, a_0\}$.

Enfin, l'intégration de ces trois sources donne le résultat illustré dans la figure 3.2.

3.1.3.3 Application au domaine des composants industriels

L'opérateur d'intégration *FragmentOnto* peut être appliqué dans les systèmes d'intégration développés pour des grands groupes industriels ou des grands donneurs d'ordre (par exemple: PSA, Renault) dans le but de centraliser des catalogues de leurs différents fournisseurs. Il est facile d'imaginer le scénario suivant:

- ces groupes proposent d'abord aux différents fournisseurs leur propre ontologie,
- chaque fournisseur extrait ensuite une partie de cette ontologie selon ses besoins pour construire son catalogue.

Dans un tel environnement, l'intégration automatique de données est assurée. C'est effectivement ce scénario que le projet français PFI (dirigé par Renault) est en train de mettre en oeuvre dans le domaine des composants hors fabrication pour les industries manufacturières. Notons que cette approche fait reposer toute la difficulté sur les fournisseurs qui doivent décrire plusieurs fois leurs données, si plusieurs consortiums existent.

3.1.4 ProjOnto

3.1.4.1 Contexte

De nombreuses applications conçues autour de l'approche d'intégration *a priori* exigent plus d'autonomie. Dans le domaine du commerce électronique professionnel qui est le nôtre:

- la classification de chaque source doit pouvoir être complètement différente de celle de l'ontologie partagée, et
- certaines spécialisations de classe et certaines propriétés n'existant pas dans l'ontologie partagée doivent pouvoir être ajoutées dans les ontologies locales. Ce cas est très différent du précédent du fait que chaque source S_i a sa propre ontologie O_i et ses classes spécifiques. Néanmoins, l'ontologie O_i *référence autant que possible* l'ontologie partagée O_p à travers l'articulation $\mathcal{A}_{i,p} :< BDBO_i, O_p, OntoSub_{i,p} >$.

Dans ce scénario, les sources ont des concepts propres qui n'existent pas dans l'ontologie partagée, mais ces concepts ne sont pas supposés intéresser les utilisateurs du système intégré. Ainsi, le système intégré vise à intégrer les instances de données de chaque source comme des instances de l'ontologie partagée. Les instances de données de chaque S_i seront donc projetées sur l'ontologie partagée. Notons que, dans ce scénario, les sources ne peuplent que les classes qui lui sont propres, pas celles de l'ontologie partagé (qui ne sont peuplées indirectement que par les subsomptions $OntoSub_{i,p}$).

3.1.4.2 Algorithme

Comme dans le scénario précédent (*FragementOnto*), l'ontologie du système intégré est exactement l'ontologie partagée: $O_{Int} = O_p$:

- $C_{Int} = C_p$,
- $P_{Int} = P_p$,
- $Applic_{Int}(c) = Applic_p(c)$,
- $Sub_{Int}(c) = Sub_p(c)$.

Cette définition montre qu'aucun concept défini localement n'est intégré dans le système intégré.

Pour ce scénario, chaque instance de données d'une source sera projetée sur les propriétés applicables de sa plus petite classe subsumante dans l'ontologie partagée. Cette plus petite classe subsumante devient donc la classe de base de l'instance intégrée. Contrairement au cas précédent, la classe de base d'une instance dans le système intégré n'est pas celle d'origine de cette instance dans sa source locale.

Soit $Pop^*(c)$ la population des instances projetées sur la classe c de l'ontologie partagée, $Pop^*(c)$ est calculée par l'union des populations de toutes classes locales référençant directement c [5]. $Pop^*(c)$ est donnée par l'équation suivante :

$$Pop^*(c) = \bigcup_{i\in[1:n]} \Big(\bigcup_{c_j \in OntoSub_{i,p}(c)} Pop_i(c_j) \Big) \tag{3.7}$$

Le schéma des instances projetées sur la classe c est déterminé comme :

$$Sch^*(c) = Applic_p(c) \bigcap \Big(\bigcap_{i\in[1:n]} \Big(\bigcap_{(c_j \in OntoSub_{i,p}(c)) \wedge (Pop_i(c_j) \neq \phi)} Sch_i(c_j) \Big) \Big) \tag{3.8}$$

ou

$$Sch'^*(c) = Applic_p(c) \bigcap \Big(\bigcup_{i\in[1:n]} \Big(\bigcup_{c_j \in OntoSub_{i,p}(c)} Sch_i(c_j) \Big) \Big) \tag{3.9}$$

La population et le schéma de chaque classe feuille de l'ontologie partagée sont calculés par les équations 3.7 et 3.8, respectivement (ou 3.7 et 3.9). Autrement dit: $Pop_{Int}(c) = Pop^*(c)$ et $Sch_{Int}(c) = Sch^*(c)$.

En revanche, pour une classe non feuille, les trois équations 3.7,3.8 et 3.9 doivent être complétées de façon récursive comme suit:

$$Sch_{Int}(c) = Applic(c) \bigcap \Big(\Big(\bigcap_{(c_k \in Sub_{Int}(c)) \wedge (Pop*(c_k) \neq \phi)} Sch_{Int}(c_k) \Big) \bigcap_{Pop*(c) \neq \phi} Sch^*(c) \Big)$$

ou

$$Sch'_{Int}(c) = Applic(c) \bigcap \Big(\Big(\bigcup_{c_k \in Sub_{Int}(c)} Sch'_{Int}(c_k) \Big) \bigcup Sch^*(c) \Big)$$

et dans les deux cas:

$$Pop_{Int}(c) = \Big(\bigcup_{c_k \in Sub_{Int}(c)} Pop_{Int}(c_k) \Big) \bigcup Pop^*(c)$$

L'exemple suivant illustrera l'algorithme d'intégration par *ProjOnto*.

[5] les classes locales d'une source S_i référençant directement la classe c de l'ontologie partagée O_p sont les classes dans $OntoSub_{i,p}(c)$.

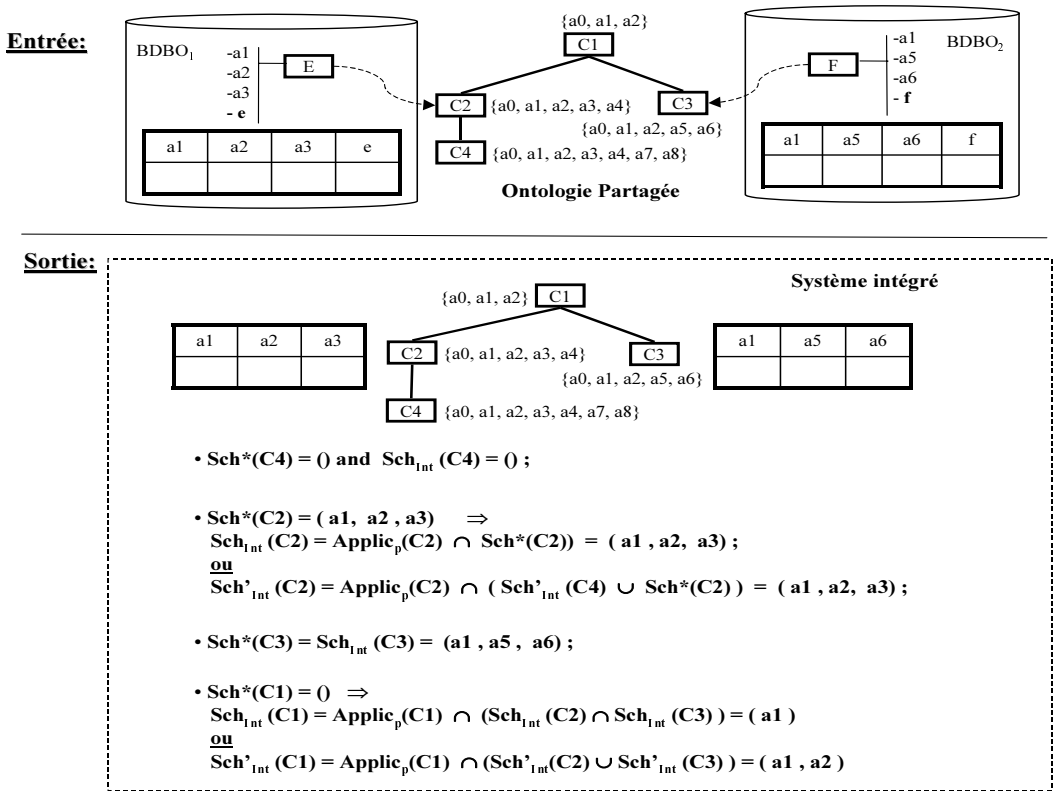


FIG. 3.3 – Exemple d'une intégration de BDBOs par ProjOnto

Exemple. : Supposons que l'on ait deux bases de données à base ontologique référençant une ontologie partagée comme dans la figure 3.3. La classe locale E dans S_1 ($BDBO_1$) référence la classe C_2 (en important les trois propriétés: a_1, a_2, a_3). Quant à la classe locale F de la source S_2 ($BDBO_2$), elle référence C_3 de l'ontologie partagée (en important: a_1, a_5 et a_6). De plus, les classes E et F ajoutent les propriétés e et f, respectivement.

Le système intégré de deux sources S_1, S_2 selon l'approche *ProjOnto* est illustré dans la figure 3.3. La classe $C2$ est une classe non feuille, mais elle devient une classe de base.

3.1.4.3 Domaine d'application: composants industriels

Le scénario d'intégration de données par *ProjOnto* est appliqué au domaine des composants industriels où :

1. il existe une ontologie normalisée du domaine (par exemple, l'ontologie normalisée *IEC 61360-4* sur le domaine des composants électroniques, ou *ISO 13399* sur le domaine des outils coupants, *ISO 13584-501* sur le domaine des matériels de mesure, etc.),
2. cette ontologie est acceptée comme une ontologie partagée par tous les participants de l'environnement "autonome" *B2B* étudié. Cet environnement *B2B* est dit *autonome*, parce que les fournisseurs gardent l'autonomie de leurs ontologies, mais,
3. chaque fournisseur décrit les classes de ses composants dans son catalogue en référençant *le plus possible* l'ontologie normalisée.

Ce scénario *ProjOnto* permet également d'intégrer automatiquement les composants issus des catalogues de différents fournisseurs comme les instances de l'ontologie normalisée.

3.1.5 ExtendOnto

3.1.5.1 Contexte

L'entrée du système intégré du scénario *ExtendOnto* est identique à celle de *ProjOnto* où chaque source a sa propre ontologie et ses classes spécifiques. En revanche, ce cas est différent du précédent du fait que le système intégré permet d'intégrer même les extensions de chaque S_i. L'ontologie partagée sera donc étendue en intégrant les ontologies locales. Pour un tel contexte, l'intégration de BDBOs consiste à intégrer d'abord les ontologies, puis les données.

Dans ce cas également, une automatisation du processus d'intégration est possible. Pour ce faire, nous devons trouver la structure finale de la *BDBO* constituant le système intégré Int : $< O_{Int}, Sch_{Int}, Pop_{Int} >$.

3.1.5.2 Algorithme

Redéfinissons d'abord la structure de l'ontologie intégrée: O_{Int} : $<C_{Int}$, P_{Int}, Sub_{Int}, $Applic_{Int}>$, où chaque élément de O_{Int} est défini comme suit:

- $C_{Int} = C_p \bigcup_{(i \mid 1 \leq i \leq n)} C_i$,
- $P_{Int} = P_p \bigcup_{(i \mid 1 \leq i \leq n)} P_i$,
- $Applic_{Int}(c) = \begin{cases} Applic_p(c), si\ c \in C_p \\ Applic_i(c), si\ c \in C_i \end{cases}$
- $Sub_{Int}(c) = \begin{cases} Sub_p(c) \cup OntoSub_i(c), si\ c \in C_p \\ Sub_i(c), si\ c \in C_i \end{cases}$

Définissons ensuite la population du système intégré. Pop_{Int} de chaque classe (c) est calculée d'une manière récursive en utilisant un parcours poste fixé de l'arbre C_{Int}.

Si la classe c appartient à une C_i et n'appartient pas à C_p, sa population est donnée par : $Pop_{Int}(c) = Pop_i(c)$. Sinon, i.e., c appartient à l'ontologie partagée, $Pop_{Int}(c)$ est définie par l'équation suivante:

$$Pop_{Int}(c) = \bigcup_{c_i \in Sub_{Int}(c)} Pop_{Int}(c_i) \tag{3.10}$$

Finalement, le schéma de chaque classe du système intégré est calculé en utilisant le même principe que la population en considérant les classes appartenant à C_i et les classes appartenant à C_p (rappelons que les classes de C_p ne sont directement peuplées par aucune source). Les schémas des classes appartenant à l'une des C_i sont explicitement définis ($Sch_{Int}(c)$=$Sch_i(c)$). Concernant les classes de C_p, le schéma de la classe c peut être calculé en appliquant la formule 3.1 (resp. 3.2) sur l'ontologie du système intégré O_{Int}:

$$Sch_{Int}(c) = Applic_{Int}(c) \bigcap (\bigcap_{(c_i \in Sub_{Int}(c)) \wedge (Pop_{Int}(c_i) \neq \phi)} Sch_{Int}(c_i)) \tag{3.11}$$

ou

$$Sch'_{Int}(c) = Applic_{Int}(c) \bigcap (\bigcup_{c_i \in Sub_{Int}(c)} Sch'_{Int}(c_i)) \tag{3.12}$$

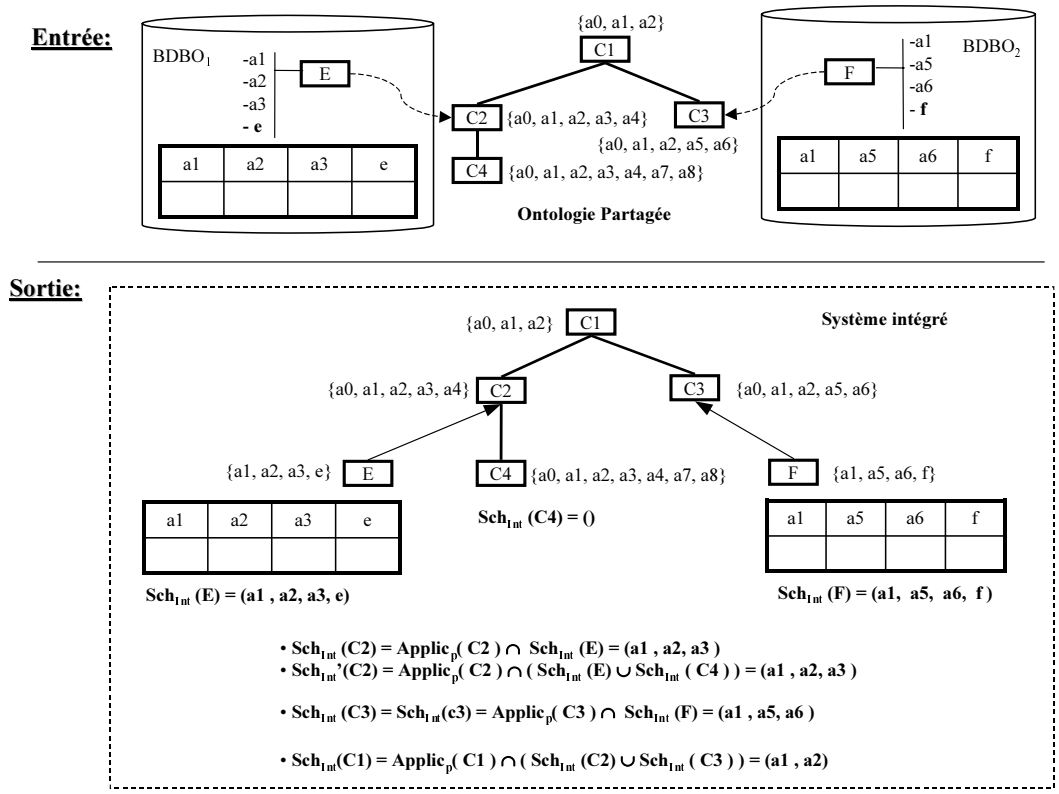


FIG. 3.4 – Exemple d'une intégration de BDBOs par *ExtendOnto*

Cela montre qu'il est possible d'offrir aux sources locales une large autonomie tout en permettant également une construction automatique du système intégré d'une manière déterministique et exacte.

Pour mieux comprendre l'algorithme d'intégration par *ExtendOnto*, nous observons l'exemple suivant.

Exemple. : nous considérons les mêmes sources et la même ontologie que dans l'exemple du scénario *ProjOnto*. La structure du système intégré est décrite par la figure 3.4. Afin d'intégrer les deux sources, l'ontologie partagée doit être étendue en intégrant deux classes locales.On note que la vue offerte à l'utilisateur à travers l'ontologie partagée est la même. Mais celui-ci peut, en plus, descendre dans le contenu initial des sources.

Il est important de noter que, lorsque toutes les sources de données utilisent une ontologie indépendante sans référencer une ontologie partagée, d'une part, l'intégration automatique peut néanmoins se produire (i.e., lecture de toutes les données dans le même entrepôt), et d'autre part, la tâche d'articulation de ces ontologies sur l'ontologie du système receveur peut être faite manuellement, par le DBA. Ensuite une nouvelle intégration peut être réalisée automatiquement comme dans le cas *ExtendOnto*. Ce scénario a également été étudié dans la thèse de Dung Nguyen XUAN [217].

3.1.5.3 Domaine d'application: composants industriels

Le scénario *ExtenOnto* est différent du scénario *ProjOnto* du fait qu'il permet de stocker au sein d'une base unique le contenu complet de chaque catalogue intégré (non seulement les concepts existants dans l'ontologie partagée, mais également les concepts propres).

Le système intégré dans ce cas supporte deux types d'accès possibles aux composants:

1. l'accès générique à travers l'ontologie normalisée, et
2. l'accès spécifique à chaque catalogue en descendant de l'hiérarchie de l'ontologie intégrée.

3.2 Développement initié par ces travaux

A notre arrivée au Laboratoire LISI en septembre 2002, nous avons intégré l'équipe ingénierie de données (IDD). Compte tenu de mes compétences en bases de données et entrepôts de données, dès mon recrutement, j'ai développé des travaux portant sur l'intégration de sources structurées dans le cadre du projet OntoDB lancé par Prof. PIERRA au début de l'année 2002. L'objectif de ce projet était de permettre la gestion, l'échange, l'intégration et l'interrogation de données structurées associées à des ontologies formelles. En collaboration avec les membres de l'équipe IDD, nous avons développé une approche ascendante d'intégration de bases de données à base ontologique dans un entrepôt de données avec pour application, les bibliothèques de composants industriels.

Les travaux de thèse de Nguyen DUNG ont permis de dégager les hypothèses pour construire une méthode d'intégration automatique. La variété des systèmes d'intégration existants nous a conduit à identifier leurs points communs et leurs divergences par le biais d'une classification basée sur trois critères orthogonaux : (1) la mise en correspondance entre les schémas locaux et le schéma global, (2) l'architecture d'un système d'intégration et (3) la nature du processus d'intégration. Le système d'intégration à base ontologique que nous avons proposé dans la thèse de Nguyen DUNG permet (1) une intégration automatique de sources de données, (2) une grande autonomie des sources et (3) une évolution asynchrone des schémas et des ontologies. Nous supposons toujours l'existence d'une ontologie partagée. Chaque source locale référence cette ontologie a priori. L'architecture que nous avons choisie pour réaliser cette intégration est de type "entrepôt", où les données des sources sont dupliquées dans le système d'intégration. Trois scénarii d'intégration (FragmentOnto, ExtendOnto et ProjOnto) ont été proposés et validés en utilisant des ontologies normalisées développées à l'initiative du laboratoire dans le domaine de l'ingénierie. Cette validation a été effectuée sur le langage EXPRESS dans l'environnement ECCO et JAVA. Elle a été mise en oeuvre pour intégrer des ensembles de catalogues de composants de données puis à gérer l'évolution de système à travers les mises à jour des catalogues tant au niveau des définitions contenues (ontologies) qu'au niveau des composants décrits (données).

La thèse de DUNG a été fortement appuyée par d'autres travaux de DEA et de master. Elle est la base de notre reflexion sur les bases de données à base ontologique du fait que l'approche d'intégration que nous avons proposée suppose une préparation des sources participant au processus d'intégration.

Une première ouverture de cette thèse est le travail que nous avons mené dans le cadre de la thèse (en cours) de Chimène FANKAM sur la proposition d'une nouvelle démarche ontologique de conception de bases de données. Vu les similarités entre les modèles conceptuels et les ontologies de domaine,

nous avons proposé une approche de conception de bases de données à partir d'une ontologie de domaine. Cette approche, appelée SISRO pour *Spécialisation Importation Spécifique et Représentation des Ontologies*, utilise une ontologie locale pour assurer une meilleure autonomie de la base de données par rapport à l'ontologie de domaine. Elle représente l'ontologie locale et les ontologie(s) partagée(s) soit par un mécanisme de vues, soit par une représentation explicite des ontologies au sein de la base de données. Les classes de l'ontologie locale sont différentes des classes des ontologies partagées avec lesquelles elles sont articulées par des relations de subsomption. Elles importent à travers cette relation celles des propriétés applicables à leur classe subsumante qui s'avèrent pertinentes pour le cahier des charges à satisfaire. Notre méthodologie comporte quatre étapes principales. Dans l'étape 1, une ontologie locale est construite par identification des concepts et propriétés des ontologies partagées pertinents par rapport au cahier des charges. Le concepteur dispose d'une grande liberté dans la structuration des classes (qui représentent les concepts retenus) et des propriétés. Cette ontologie locale peut alors être spécialisée par les concepts nécessaires qui ne figuraient pas dans les ontologies partagées, puis étendue par ajout de nouvelles propriétés. Dans la seconde étape, le modèle conceptuel de l'application est défini à partir de l'ontologie locale. Le concepteur peut ne choisir qu'un sous ensemble de l'ontologie locale pour prévoir par exemple les extensions futures. Cette différence entre ontologie locale et modèle conceptuel de données est justifiée par le caractère descriptif des ontologies et la nature prescriptive des modèles conceptuels. Dans la troisième étape, le modèle conceptuel est traduit en modèle logique, puis en modèle physique à partir d'un ensemble de règles de transformation suivant le formalisme utilisé. Enfin, la quatrième étape fournit un accès aux données au niveau ontologique afin de permettre l'intégration automatique. Dans cette étape, le concepteur a le choix entre deux possibilités : (1) utiliser des vues pour représenter les classes des ontologies concernées ; (2) représenter dans la même base de données à la fois les données, les concepts ontologiques qui en définissent le sens (ontologie locale, ontologie partagée), mais aussi le lien entre ontologie et données au travers des identifiants de concepts.

Une deuxième direction inspirée des travaux de de thèse de Dung Nuyen XUAN concerne une proposition d'enrichissement de l'architecture ANSI/SPARC par une couche ontologique (figure 3.5). Cette proposition a été faite dans le cadre de la thèse de Stéphane JEAN [125], effectuée au laboratoire LISI.

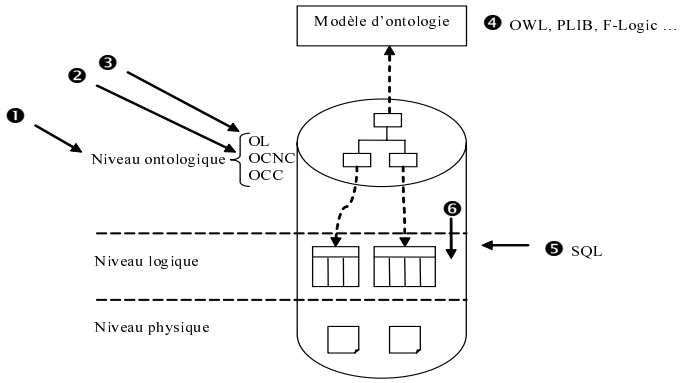


FIG. 3.5 – Proposition d'un enrichissement de l'architecture ANSI/SPARC

3.3 Conclusion

Dans ce chapitre, nous avons proposé une méthode complètement automatique d'intégration de sources de données structurées hétérogènes et autonomes. Cette approche, appelée intégration par articulation *a priori* d'ontologies, suppose l'existence d'une (ou plusieurs) ontologie(s) de domaine, mais elle laisse chaque source autonome quant à la structure de sa propre ontologie.

Au lieu de réaliser l'intégration des ontologies *a posteriori*, comme c'est le cas dans toutes les approches classiques, notre approche exige de l'administrateur de chaque source à intégrer que:

1. sa base de données contienne une ontologie, et
2. qu'il *s'engage sur l'ontologie de domaine*, c'est-à-dire qu'il ajoute *a priori* à cette ontologie les relations (articulations de subsomption) existant entre celle-ci et l'ontologie de domaine.

Cette hypothèse est réaliste dans tous les secteurs où des ontologies de domaines existent ou apparaissent, et où chaque administrateur qui publie sa base de données souhaite à la fois lui conserver sa structure propre, et la rendre accessible à des usagers de façon homogène à travers une ontologie de domaine. C'est en particulier le cas dans le cadre du commerce électronique professionnel. Elle exige également que chaque source publie non seulement ses données, mais également son ontologie, ce qui correspond à une généralisation de l'approche de type meta-données utilisée pour les sources semi-structurées.

Nous avons également défini plusieurs opérateurs d'intégration correspondant à différents scénarii possibles, à savoir $FragmentOnto$, $ProjOnto$ et $ExtendOnto$, qui dotent l'ensemble les BDBOs d'une structure d'algèbre. Ces différents opérateurs correspondent à différents scénarii de coopération entre les sources de données.

L'ensemble de ces propositions a été validé sur des exemples issus du commerce électronique professionnel (B2B) et de l'intégration des catalogues de composants industriels (projet PLIB). Nous avons présenté ici la mise en oeuvre de cette approche dans une perspective d'intégration physique des données au sein d'un entrepôt. Nous utilisons des ontologies PLIB qui sont bien adaptés pour représenter les entités d'un domaine fortement structuré et les propriétés intrinsèques qui les caractérisent. Ce type d'ontologie est bien adapté au domaine du commerce électronique professionnel et se généralise dans ce domaine. De plus, ce modèle d'ontologie, formellement défini dans le langage EXPRESS, est associé à une structure d'échange permettant de représenter de façon neutre tant des ontologies que des instances référençant ces ontologies. Les algorithmes que nous avons présentés permettent alors, à travers un tel échange, l'intégration automatique du contenu de toute nouvelle source autonome au sein d'un entrepôt déjà constitué à partir de l'ontologie de domaine. Cette intégration peut même étendre automatiquement, si on le souhaite, l'ontologie de domaine en respectant la compatibilité ascendante. Nous avons utilisé dans cette implémentation des ontologies PLIB qui permettent de typer finement les valeurs des propriétés des composants (unités explicites, représentation des dépendances entre propriétés) et de référencer une classe d'une autre ontologie sans importer entièrement cette dernière.

Chapitre 4

Gestion de l'évolution asynchrone d'un système d'intégration à base ontologique

Publications : [221, 219, 220].

Le système d'intégration à base ontologique doit répondre aux évolutions des sources du fait que les fournisseurs des sources de données sont différents, chaque source doit pouvoir, en plus, se comporter indépendamment des autres (tout particulièrement dans des environnements dynamiques comme le WWW [116]). La relation entre le système intégré et ses sources est faiblement couplée. Dans un tel contexte, il est difficile de synchroniser l'évolution de l'ontologie partagée et les évolutions des sources. Cela est dû aux facteurs suivants : (i) l'indépendance des sources et (ii) le temps nécessaire pour diffuser les évolutions des ontologies partagées dans une communauté. Les sources à intégrer peuvent donc ne pas référencer à une même version de l'ontologie partagée. La figure 4.1 montre un exemple de l'intégration de BDBOs dans un environnement asynchrone. Dans ce contexte, l'automatisation du processus d'intégration n'est pas assurée.

L'objet du chapitre est de présenter une approche permettant d'offrir des solutions à la deuxième étape de cycle de vie d'un système d'intégration qui est l'évolution asynchrone de l'ensemble de sources, tout en maintenant la possibilité d'intégration automatique de ces dernières au sein d'un entrepôt. *Deux problèmes* doivent être résolus:

1. la gestion de cycle de vie des instances, c'est-à-dire la représentation de la connaissance des instances qui existaient dans l'entrepôt à un instant donné, et
2. la gestion des évolutions des ontologies afin de maintenir la cohérence des relations entre ontologies et entre ontologies et données.

Avant de décrire nos solutions sur l'évolution, nous présentons un état de l'art sur les travaux antérieurs.

4.1 Travaux antérieurs sur les évolutions de données

L'évolution dans notre contexte doit concerner deux axes : (1) l'évolution de données et de schéma et (2) l'évolution d'ontologies.

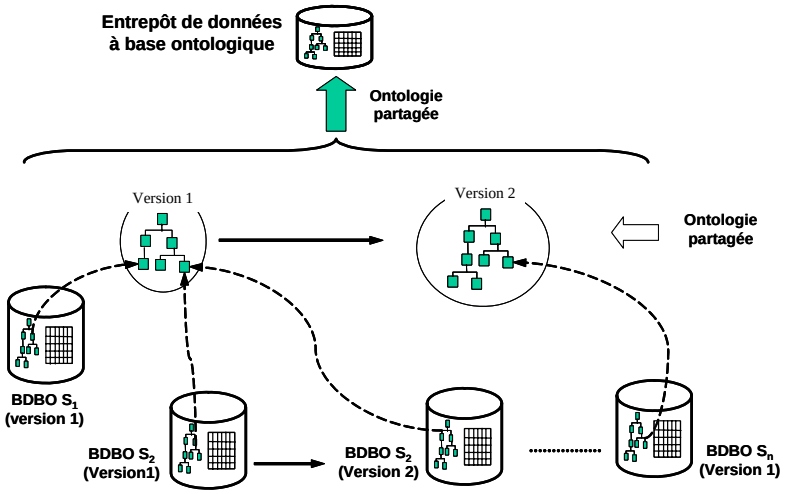


FIG. 4.1 – Exemple de l'intégration de bases de données à base ontologique dans un environnement asynchrone.

4.1.1 Évolution de données

Les changements que subissent au cours du temps les schémas de bases de données représentent un problème classique. Celui-ci a été abordé dans le contexte des bases de données relationnelles et des bases orientées objet [186, 187, 194, 151, 213]. On peut distinguer deux approches de ces problèmes, à savoir, l'approche par *schéma versionné* et l'approche par *schéma évolutif*.

- Dans l'approche par *schéma versionné* [151, 213], toutes les versions de chaque table sont explicitement stockées. Cette solution a deux avantages principaux :(i) elle est facile à implémenter et permet une automatisation du processus de mise à jour, et (ii) elle offre un traitement des requêtes rapide dans le cas où l'on précise la ou les versions de recherche. Par contre, le coût peut devenir important si la requête nécessite un parcours de toutes les versions de données disponibles dans la base. Un autre inconvénient est le coût de stockage à cause de la duplication des données.
- Dans l'approche par *schéma évolutif* [186, 213], un seul schéma est représenté pour chaque table. Ce schéma est obtenu en faisant l'*union* de toutes les propriétés figurant dans les différentes versions. On y ajoute, à chaque rafraîchissement, toutes les instances existant dans la nouvelle version de la table même si elles existaient déjà dans la version précédente. Les instances sont complétées par des valeurs nulles. Cette solution évite la représentation de plusieurs versions de chaque table. Les inconvénients majeurs de cette solution sont : (i) le problème de duplication est toujours présent, (ii) l'implémentation est un peu plus difficile que l'approche précédente en ce qui concerne le calcul automatique du schéma des tables stockées; (iii) le tracé du cycle de vie de données est difficile à mettre en oeuvre (*"valid time"* [213]) et (iv) l'ambiguïté sémantique des valeurs nulles: figuraient-elles dans la table insérée, ou résultent-elles de l'intégration ?.

Pour les entrepôts de données (data warehouse), la problématique d'évolution de données a également été abordée. Les travaux dans ce domaine visent à adapter les résultats obtenus précédemment aux caractéristiques particulières des entrepôts telles que la nature de données intégrées et leur caractère multidimensionnel. On peut identifier deux types de travaux portant respectivement sur l'évolution du schéma d'entrepôt [7, 53, 136], et sur la maintenance des vues d'entrepôt [190].

Les travaux sur les entrepôts WHIPS relèvent de la première problématique. Dans un entrepôt WHIPS [136], par exemple, les évolutions de données sont faites en ligne. En parallèle, l'entrepôt de données peut recevoir des changements issues des différentes sources. Si ces changements sont appliqués dans un ordre inadapté, cela peut produire des inconsistances de données dans l'entrepôt. WHIPS a proposé un algorithme de synchronisation permettant de calculer un ordre pertinent pour les changements [224].

Le système d'intégration EVE [190] relève du dernière type de travaux. Il vise à automatiser la maintenance des vues. Ce système propose une nouvelle extension de SQL (E-SQL) qui permet de définir *a priori* des évolutions acceptables pour une vue matérialisée. E-SQL ajoute des paramètres spécifiques à chaque vue. Ces paramètres déterminent *a priori* quelles informations (attributs/relations/conditions) sont indispensables, quelles informations sont remplaçables par des informations similaires provenant des sources, et quelles conditions doivent être respectées tout au long de l'évolution d'une vue. En se basant sur E-SQL, EVE permet de re-définir ses vues d'une façon automatique.

4.1.2 Évolution d'ontologies

Comme tout composant informatique, une ontologie évolue et nécessite donc une gestion de ses différentes versions. L'évolution d'une ontologie est normalement beaucoup plus problématique que les évolutions d'une base de données. Ceci est dû au caractère partageable de l'ontologie. Pour illustrer ce problème, citons Rogozan [188] qui s'interroge : *"comment préserver l'accès et l'interprétation des objets au moyen de leur référencement à une ontologie évolutive"* ?

En effet, le changement d'une ontologie utilisée comme une interprétation sémantique peut produire des pertes des coordonnées sémantiques, c'est-à-dire des références d'objets établies par rapport à un certain référentiel sémantique. En conséquence, les objets utilisant ces références ne sont plus accessibles au moyen de l'ontologie modifiée [115, 162]. Pour cette raison, Noy et al. [162] considèrent que toutes les versions d'une même ontologie doivent être conservées. Ils ont donc définit l'évolution de l'ontologie comme: "*la capacité de gérer les changements de l'ontologie et leurs effets sur les instances, en créant et en maintenant différentes versions d'une ontologie. Cette capacité consiste à différencier et à identifier les versions, à spécifier les relations qui explicitent les changements effectués entre versions et à utiliser des mécanismes d'accès pour les artefacts dépendants*".

Ainsi, la plupart des approches s'intéressent plus aux conséquences sur les individus de l'ontologie (que nous appellons *instances à base ontologique*) que sur des évolutions de l'ontologie. Deux approches peuvent, néanmoins, être distinguées dans la littérature:

- La première approche [204, 205, 203] vise à supporter les processus d'évolution des ontologies. Elle étudie les différents opérateurs de modification d'ontologies (adjonction, suppression, modification de domaines) et les conséquences qui en résultent pour la cohérence des instances à base ontologique. Lorsque les modifications sont incompatibles avec l'état courant des instances, cette approche étudie comment modifier les instances pour les rendre cohérentes avec la nouvelle version de l'ontologie.
- La deuxième approche [132, 162, 163] ne vise pas à gérer le processus d'évolution des ontologies et les conséquences sur les instances existantes, mais plutôt à permettre la représentation et la comparaison des différentes versions d'une ontologie afin de faire migrer les instances à base ontologique de l'une vers l'autre. Cette approche analyse et représente la compatibilité entre les

versions de l'ontologie.

Par exemple, le travail de Stojanovic [203] relevant de la première approche, propose un processus d'évolution d'une ontologie composée de six étapes principales:

1. *CLa capture du changement*. Le processus d'évolution de l'ontologie commence à capturer les changements à partir des exigences explicites ou du résultat des méthodes de découverte des changements qui induisent des changements à partir des données existantes.
2. *La représentation du changement*. Cette étape vise à représenter les changements dans un format approprié. Les changements peuvent être représentés à deux niveaux: (i) les changements élémentaires (l'adjonction, la suppression, la modification des entités ontologiques) et (ii) les changements complexes (par exemple, la fusion ou la séparation des entités ontologiques).
3. *La sémantique du changement*. L'ontologie doit évoluer d'un état consistant vers un autre état consistant [162]. Cette phase permet de résoudre des changements induits d'une manière systématique en assurant la consistance de toute l'ontologie. Afin de résoudre les inconsistances introduites par les changements, d'autres changements additionnels sont nécessaires.
4. *La propagation du changement*. Le but de cette étape est de modifier automatiquement les instances et les ontologies dépendantes (qui utilisent une partie de l'ontologie évolutive dans sa structure ontologique [188]) afin de préserver leur consistance avec l'ontologie évoluée.
5. *L'implémentation du changement*. Cette phase consiste à (i) informer l'administrateur de toutes les conséquences d'un changement, (ii) à exécuter le changement, une fois approuvé par l'administrateur.
6. *La validation du changement*. Cette phase permet la justification des changements exécutés et de les annuler à la demande de l'utilisateur.

Par ailleurs, dans OWL [146], les ontologies, les classes, et les propriétés peuvent être annotées pour tracer l'existence d'évolutions ontologiques. Elles peuvent toutes être associées à un numéro de version ($versionInfo$). De plus, trois étiquettes peuvent être utilisées pour représenter la compatibilité entre les différentes versions d'une ontologie: *priorVersion*, *backwardCompatibleWith* et *incompatibleWith* [114, 116]. Ceci permet, dans le cas où la compatibilité est déclarée (*backwardCompatibleWith*), de savoir qu'il n'y a pas lieu de modifier les instances.

4.1.3 Synthèse d'intégration et évolution

Cet état de l'art montre que la prise en compte de l'évolution des ontologies dans les systèmes d'intégration à base ontologique est un problème crucial et que personne, tout au moins à notre connaissance avant le commencement de la thèse de Dung NGUYEN XUAN [217], n'a encore proposé de méthode permettant simultanément l'intégration automatique de sources de données hétérogènes et la prise en compte de l'évolution asynchrone tant des ontologies que des données. Les conditions de faisabilité d'une telle approche, et la description de sa mise en oeuvre au sein d'un système d'intégration de type entrepôt constituent précisément l'objectif de la suite du chapitre.

Dans les travaux de thèse de Dung NGUYEN XUAN [217], nous avons proposé une approche permettant à l'ensemble de sources de données d'évoluer de façon asynchrone tout en maintenant la possibilité d'intégration automatique.

4.2 Gestion des évolutions des ontologies

Permettre à des ontologies que on souhaite intégrer d'évoluer au cours du temps nécessite sans aucun doute qu'une certaine cohérence soit maintenue tout au long de ces évolutions pour assurer une notion de "compatibilité" entre ontologies [116]. Nous définissons d'abord dans cette section les contraintes que nous proposons d'introduire pour limiter les évolutions autorisées et qui correspondent à l'étiquette *backwardCompatibleWith* de OWL. Nous présentons ensuite le modèle de gestion que nous proposons pour supporter de telles évolutions même lorsqu'elles sont réalisées de façon asynchrone.

4.2.1 Principe de continuité ontologique

Les contraintes que l'on peut définir pour régler l'évolution des entrepôts de données à base ontologique résultent des différences fondamentales existantes, du point de vue évolution, entre les modèles conceptuels et les ontologies. Un modèle conceptuel est un modèle, c'est-à-dire, selon Minsky [152], un objet qui permet de répondre à certaines questions que l'on se pose sur un autre objet, à savoir définir les informations représentées dans une base de données. Lorsque les questions changent [6], son modèle conceptuel est modifié en conséquence, et ceci, sans que cela signifie le moins du monde que le domaine modélisé a été modifié. Au contraire, une ontologie est une conceptualisation visant à représenter l'essence des entités d'un domaine donné sous forme consensuelle pour une communauté. C'est une théorie logique d'une partie du monde, partagée par toute une communauté, et qui permet aux membres de celle-ci de se comprendre. Ce peut être, par exemple, la théorie des ensembles (pour les mathématiciens), la mécanique rationnelle (pour les mécaniciens) ou la comptabilité analytique (pour les comptables). Pour de telles ontologies, deux types de changements doivent être distingués : (1) *l'évolution normale* d'une théorie est son approfondissement. Des vérités nouvelles, plus détaillées s'ajoutent aux vérités anciennes. Ce qui était vrai hier reste vrai aujourd'hui. (2) Il peut également arriver que des axiomes de la théorie aient à être remis en cause. Dans ce cas, il ne s'agit plus d'une évolution mais d'une *révolution*, où deux systèmes logiques différents vont coexister ou s'opposer.

Les ontologies que nous visons correspondent à cette conception. Il s'agit d'ontologies soit normalisées, par exemple au niveau international (ISO 13584-511), soit définies par des consortiums importants, et qui formalisent de façon stable les connaissances d'un domaine technique. Les changements auxquels nous nous intéressons dans notre approche ne sont donc pas des révolutions, qui correspondent à un changement d'ontologie, mais les évolutions d'ontologie.

Nous avons imposé donc aux ontologies manipulées, qu'elles soient globales ou locales de respecter la contrainte suivante dont nous détaillons les **conséquences dans la section 4.2.**

Principe de continuité ontologique: *si l'on considère chaque ontologie intervenant dans le système d'intégration à base ontologique comme un ensemble d'axiomes, tout axiome vrai pour une certaine version de l'ontologie restera vrai pour toutes les versions ultérieures*.

[6] les objectifs organisationnels auxquels répond un système d'information sont modifiés

4.2.2 Contraintes sur les évolutions des ontologies

Nous détaillons ici les conséquences du principe de continuité ontologique.

4.2.2.1 Identification des classes et propriétés

Gérer l'évolution suppose de pouvoir désigner, et donc identifier, tous les éléments faisant l'objet d'évolution.

Nous avons déjà précisé que toute source, toute classe et toute propriété étaient associées à des identifiants universels (ID: universal identifier). En fait, dans le contexte multi-versions, qui est le notre, ces identifiants contiennent deux parties: un code (unique) identifiant chaque concept et une version, entière, identifiant les versions d'un concept: $ID ::= code\ version$. Le code du même concept reste invariant. Chaque changement dans sa définition change sa version. Nous noterons $code(c)$ la fonction qui, à chaque concept de l'ontologie (classe ou propriété) fait correspondre son code et $version(c)$ celle qui lui fait correspondre sa version. Les deux fonctions sont étendues aux ensembles. Ainsi, si $C = \{c_i, i = 1...n\}$, $code(C) = \{code(c_i)|c_i \in C\}$.

Toute référence entre éléments utilisant cet ID, la référence est elle-même versionnée par les versions de ses extrémités. Enfin, toute définition de classe ou de propriété contient, en particulier, la date à partir de laquelle cette version est valide.

Notons que cette caractéristique est systématiquement présentée dans les ontologies PLIB que nous manipulons [178]. Elle est également possible en OWL [146] à travers l'étiquette $versionInfo$.

Notons maintenant par un indice supérieur la version des différentes composantes d'une ontologie : $O^k =< C^k, P^k, Sub^k, Applic^k >$.

4.2.2.2 Permanence des propriétés

De même :$\forall k, code(P^k) \subset code(P^{k+1})$.

Une propriété pourra, de même, devenir obsolète sans que la valeur existante d'une propriété pour une instance existante puisse être remise en cause. Une propriété pourra évoluer dans sa définition ou dans son domaine de valeurs, par contre le principe de continuité ontologique implique que les *domaines de valeurs* ne puissent être que *croissants*, certaines valeurs étant, éventuellement, marquées comme obsolètes.

4.2.2.3 Permanence des classes

L'existence d'une classe ne pourra être infirmée à une étape ultérieure : $\forall k, code(C^k) \subset code(C^{k+1})$.

En effet, si un individu est une vis, il restera éternellement (sauf révolution) une vis. Par contre, il pourra être défini plus précisément comme une "vis hexagonale avec embase". Pour tenir compte de la réalité, il pourra apparaître pertinent de considérer comme obsolète telle ou telle classe. Elle sera alors marquée en tant que telle ("deprecated"), mais elle continuera à faire partie des versions ultérieures de l'ontologie. Par ailleurs la définition d'une classe pourra être affinée sans que l'appartenance à cette classe

d'une instance antérieure ne puisse être remise en cause. Cela signifie que : (i) la définition des classes pourra elle-même évoluer, (ii) chaque définition d'une classe sera associée à un numéro de version, et (iii) la définition (intensionnelle) de chaque classe englobera les définitions (intentionnelles) de ses versions antérieures.

4.2.2.4 Permanence de la subsomption

La subsomption est également un concept ontologique qui ne pourra être infirmé. Notons $Sub^* : C \rightarrow 2^C$ la fermeture transitive de la relation de subsomption directe Sub. On a alors : $\forall c \in C^k, code(Sub^{*k}(c)) \subset code(Sub^{*k+1}(c))$.

Cette contrainte permet évidemment un enrichissement de la hiérarchie de subsomption des classes, par exemple en intercalant des classes intermédiaires entre deux classes liées par une relation de subsomption.

4.2.2.5 Description des instances

Le fait qu'une propriété $p \in Applic(c)$ signifie que la propriété est rigide [107] pour toute instance de c. Il s'agit encore d'un axiome qui ne pourra être infirmé :

$\forall c \in C^k, code(Applic^k(c)) \subset code(Applic^{k+1}(c))$

Soulignons que ceci ne suppose pas que les mêmes propriétés soient toujours utilisées pour décrire les instances d'une même classe. Il ne s'agit plus là en effet d'une caractéristique de nature ontologique, mais seulement de nature schématique.

4.2.2.6 Cycle de vie des instances

Dans une ontologie PLIB, l'extension d'une classe (c'est-à-dire l'ensemble d'instances qui lui appartient à un moment donné) est également associée à un autre numéro de version, c'est la version de l'extension.

Nous supposons que le cycle de vie des instances est définie par ce graphe de la figure 4.2.

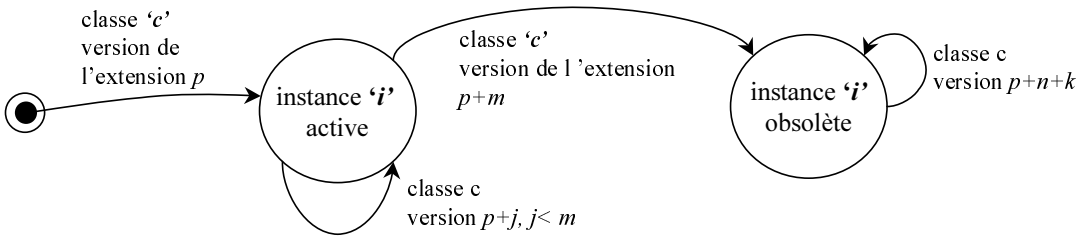


FIG. 4.2 – Cycle de vie d'une instance

Le cycle de vie d'une instance (apparition et obsolescence) peut donc être défini par les versions de l'extension de la classe de base auxquelles elle appartient.

4.2.2.7 Bilan

Les deux tables ci-dessous résument les contraintes d'évolutions ontologiques que nous proposons pour assurer le principe de continuité ontologique :

- **V** signifie que l'opération sur l'attribut affecte (augmente) la *Version* du concept,
- **X** signifie que l'opération n'est pas autorisée sur l'attribut du concept, et
- **-** signifie que l'opération sur cet attribut n'affecte pas la version.

Attribut	Ajout	Modification	Suppression
ID(Code)	X	X	X
Classe auquelle elle est définie	V (nouvelle superclasse) / X (autre classe)	X	X
Type de données	X	V(étendu)/X(restreint)	X
Description (Name, définition, ...)	-	-	X

TAB. 4.1 – Évolution de propriété en PLIB

Attribut	Ajout	Modification	Suppression
ID(Code)	X	X	X
Classe subsumante	X	V (nouvelle classe, sous classe de la subsumante précédente) / X (autre classe)	X
Description (Name, Définition)	-	-	X
Propriétés applicables (*Applic*)	V	V(ajout)/X(suppression)	X

TAB. 4.2 – Évolution de classe en PLIB

4.2.3 Modèle de gestion des évolutions

Les objectifs de ce modèle de gestion sont:

1. de permettre d'intégrer automatiquement dans l'entrepôt des sources qui sont elles-mêmes dans des versions différentes et qui référencent des *versions différentes de l'ontologie partagée*,
2. de permettre *l'accès uniforme à l'ensemble des instances* existant à un instant donné dans l'entrepôt via l'ontologie de l'entrepôt,
3. de connaître *l'historique des instances*, et
4. éventuellement, de savoir, pour chaque instance, *à quelle version de l'ontologie elle correspond*.

Nous décrivons ci-dessous, d'abord l'hypothèse portant sur la méthode de rafraîchissement de l'entrepôt, ensuite comment les deux premiers objectifs peuvent être atteints par le mécanisme des versions flottantes.

4.2.3.1 Réalisation des mises à jour

Nous supposons que notre entrepôt de données est rafraîchi de la façon Suivante : à des moments donnés, choisis par l'administrateur de l'entrepôt, la version courante d'une source S_i est intégrée dans l'entrepôt. Cette version courante de S_i comporte son ontologie, ses références à l'ontologie partagée, et son extension. Notons que, dans cette extension, certaines instances pouvaient déjà exister a posteriori

dans l'entrepôt, d'autres peuvent être nouvelles, d'autres enfin peuvent avoir été supprimées (i.e., être devenues obsolètes).

Ce scénario correspond, par exemple, dans le domaine de l'ingénierie, à un entrepôt qui consolide les descriptions de composants d'un ensemble de fournisseurs. Un rafraîchissement est effectué chaque fois qu'une nouvelle version d'un catalogue électronique d'un fournisseur est reçue.

4.2.3.2 Accès uniforme aux instances courantes: Modèle des versions flottantes

On appelle *instances courantes* de l'entrepôt les instances résultant du plus récent rafraîchissement de chacune des sources. La principale difficulté qui résulte de l'autonomie de chaque source est que, lors de deux rafraîchissements simultanés par deux sources différentes, la même classe de l'ontologie partagée c peut être référencée par une articulation de subsomption dans des versions différentes. Par exemple les versions c^k et c^{k+j} peuvent être référencées, au même moment, par deux classes c_i^n et c_j^p.

Il convient de noter que, compte tenu du principe de continuité ontologique :

1. toutes les propriétés applicables à c^k sont également applicables à c^{k+j}, et
2. toutes les classes subsumées par c^k sont également subsumées par c^{k+j}.

Donc la relation de subsomption entre c^k et c_i^n entraîne l'existence d'une relation de subsomption entre c^{k+j} et c_i^n . La classe c^k n'est donc pas nécessaire pour accéder aux instances de c_i^n.

Cette remarque nous amène à proposer un modèle appelé, *modèle des versions flottantes,* qui nous permet d'accéder à toutes les instances courantes de l'entrepôt via une seule version de l'ontologie de l'entrepôt. Cette version, appelée "*version courante*" de l'ontologie de l'entrepôt, est telle que la version courante de chacune de ses classes c^f est supérieure ou égale à la plus grande version de cette même classe qui a été référencée par une articulation de subsomption lors d'un quelconque rafraîchissement.

Pratiquement, cette condition est assurée de la façon suivante :
- si une articulation $\mathcal{A}$ définit une relation subsomption dans une classe c_p^f dans une version inférieure à f, alors la relation de subsomption $OntoSub_{i,p}^{-1}$, est modifiée pour référencer c_p^f,
- si une articulation $\mathcal{A}$ définit une relation subsomption avec une classe c^f avec une version supérieure à f, alors l'entrepôt télécharge la dernière version de l'ontologie partagée et fait migrer toutes les références $\mathcal{A}_{i,n}$ (i = 1..nombre des sources) vers les nouvelles versions courantes de l'ontologie partagée.

Exemple. : (cet exemple est illustré dans la Figure 4.3). Lors d'un rafraîchissement, une classe C_1^2 est déclarée subsumée par une classe partagée c^2 par l'articulation $\mathcal{A}_{1,p}$ (a). Mais la version courante de c dans l'entrepôt est 1 (b). Alors l'entrepôt télécharge la version courante de l'ontologie partagée (c). Celle-ci étant 3, l'articulation $\mathcal{A}_{1,p}$ est modifiée pour définir la classe c_1^2 comme subsumée par c^3 (d).

Le modèle des versions flottantes permet donc bien d'accéder à toutes les instances de l'entrepôt quelles que soient les sources dont elles proviennent à l'aide de la seule version courante de l'ontologie de l'entrepôt. La section suivante présente le modèle de gestion du cycle de vie des instances.

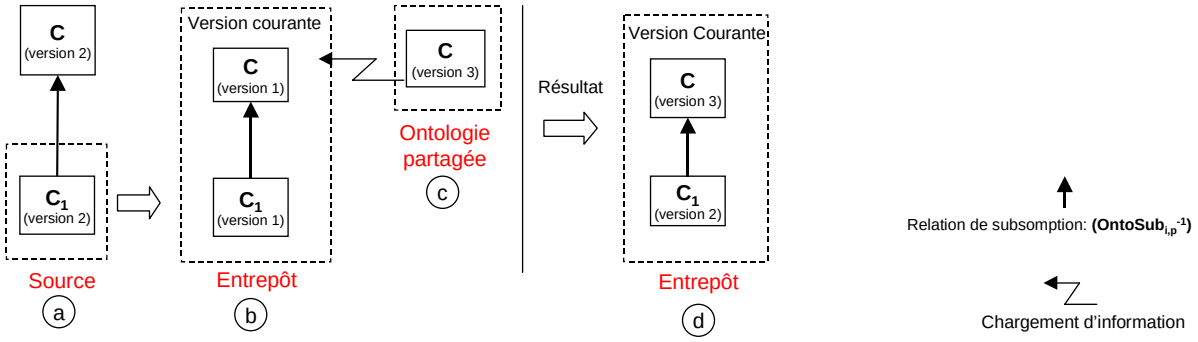


FIG. 4.3 – Exemple de comportement du modèle des versions flottantes

4.3 Gestion de l'évolution des instances

Nous présentons dans cette section nos propositions pour gérer l'évolution des instances dans l'entrepôt. Nous présentons d'abord un mécanisme, la *clé sémantique*, qui permet de reconnaître une instance même si sa représentation change. Nous proposons ensuite une approche pour tracer le cycle de vie des instances.

4.3.1 Identification des instances

En règle générale, la durée de vie d'une instance peut être largement supérieure au cycle de mise à jour des sources de données. C'est en particulier le cas pour les instances des catalogues de composants industriels [38], dont certaines durent de nombreuses années alors que les catalogues sont mis à jour sur une base annuelle.

Afin de pouvoir identifier d'une façon non ambiguë une instance, toute source doit définir pour la population de chacune de ses classes de base une *clé sémantique*. Cette clé consiste à une ou plusieurs propriétés applicables de la classe dont les valeurs fournies pour les instances obéissent à une *contrainte d'unicité*. Un cas courant dans les catalogues de composants industriels est la "part number" qui identifie de façon permanente un objet dans sa classe. Ces valeurs devront toujours exister et être non "*null*" pour chaque instance, et elles ne devront jamais être modifiées d'une version à l'autre pour une même instance. Les clés sémantiques permettent donc non seulement de distinguer les instances d'une même version de l'extension d'une classe, mais également d'identifier la même instance dans différentes versions de cette extension.

4.3.2 Gestion du cycle de vie des instances

Nous souhaitons représenter, dans l'entrepôt, l'historique des instances intégrées.

Dans les bases de données usuelles, ce problème a été étudié sous le nom de *version de schéma*. Dans ce contexte, deux solutions ont été proposées: (1)l'approche par "*schéma versionné*" et (2)l'approche par "*schéma évolutif*". Chaque approche possède des avantages et des inconvénients. Un des inconvénients commun aux deux approches est le fait de dupliquer une instance si elle apparaît successivement dans

plusieurs versions du même schéma. Ceci provient du fait que, en l'absence d'ontologie et de clé sémantique, il n'est pas possible de reconnaître une instance si son schéma de représentation a été modifié.

Notre approche va tirer profit de l'existence de ces deux éléments pour mettre en oeuvre de façon originale l'approche *schéma évolutif*. Dans l'approche que nous proposons, toutes les instances des différentes versions d'une même classe de base seront représentées dans une unique table, mais de plus:

- Deux colonnes, appelées *version_min* et *version_max*, permettent de définir entre quelles versions de l'extension de sa classe, une instance a été active. Ces colonnes nous permettent de savoir:
  1. la première version de l'extension de sa classe (*version_min*), pour laquelle une instance est apparue, et
  2. la première version de l'extension de sa classe (*version_max*), pour laquelle une instance a disparu (Null signifie que l'instance est toujours active).
- Chaque instance n'est représentée qu'une seule fois en étant reconnue à chaque version par sa clé sémantique et par l'identification de sa classe de base.
  Si cette instance est définie par certaines propriétés différentes dans les différentes versions, toutes ses valeurs de propriétés sont accumulées dans sa description. Notons qu'il ne peut y avoir de conflit de valeurs pour une même propriété d'une même instance car nous ne nous intéressons dans le cadre de nos applications qu'aux propriétés rigides [107], c'est à dire celles dont la valeur ne peut changer sans changer l'instance. Pour les propriétés dont la valeur dépend également du contexte, cette dépendance est représentée au niveau de l'ontologie ce qui évite toute ambiguïté [178].
- Le schéma (Sch) de chacune des versions d'extension est conservé dans l'entrepôt de façon à savoir, pour chaque version de l'extension, quelles propriétés étaient fournies. Ainsi, la sémantique de la valeur *null* est explicitée.

Cette approche élimine tous les inconvénients identifiés pour l'approche *schéma évolutif*, à savoir (1) la duplication de données, (2) l'absence de la représentation du cycle de vie des instances et (3) l'ambiguïté de la sémantique de la valeur *"null"*. Une instance implémentée selon la solution ci-dessus est dite *instance multi-versionnée*.

Dans la section qui suit, nous présentons l'implémentation validant nos propositions.

4.4 Mise en oeuvre de notre modèle

Nous présentons ci-dessous deux implémentations réalisées dans notre modèle de gestion d'évolution, à savoir, (1) *l'entrepôt avec versionnement des instances mais sans historisation ontologique* et (2) *l'entrepôt avec versionnement et historisation ontologique*. Ces deux implémentations sont en fait disponibles au choix de l'utilisateur, au sein d'un même entrepôt de données, nommé *OntoDaWa*.

4.4.1 Entrepôt avec versionnement des instances mais sans historisation ontologique

Il peut être souvent suffisant de pouvoir gérer les évolutions asynchrones et de savoir quelles instances existaient dans l'entrepôt à tout instant passé sans qu'il apparaisse nécessaire d'archiver toutes les

versions successives d'ontologies. En effet, la version courante de l'ontologie est compatible avec toutes les instances passées, elle suffit pour interpréter le contenu.

Un tel entrepôt comporte deux parties :

1. la partie d'*Ontologie*, contient la version courante de l'ontologie intégrée.
2. la partie de *Contenu*, est composée des *tables multi-versionnées* dont chaque instance est une *instance multi-versionnée*.

Soulignons que, seules sont représentées les versions courantes des diverses classes de l'ontologie, un tel entrepôt peut néanmoins être rafraîchi de façon asynchrone par diverses sources qui référencent des versions différentes de l'ontologie partagée.

4.4.1.1 Gestion de la version courante de l'ontologie

Nous résumons ci-dessus le processus pour créer et mettre à jour la version courante d'un concept (c_i) de l'entrepôt. Nous nous placons dans le cas du scénario ExtendOnto, où les nouveaux concepts venant des sources sont intégrés dans l'entrepôt. Il consiste à intégrer le concept c_i en respectant le modèle des versions flottantes. Avant de décrire ce processus, notons que:

– $c_i^{current}$ représente la version courante du concept c_i dans l'entrepôt, si elle existe.
– c_i^{new} représente la nouvelle version à intégrer du concept c_i.

L'intégration du concept c_i dans l'entrepôt se compose de trois étapes :

1. ***vérifier l'existence*** du concept c_i dans l'entrepôt. Il s'agit de trouver un concept dans l'entrepôt qui lui équivaut. Grâce à la séparation code/version dans l'identifiant, nous ne comparons que $code(c_i^{new})$ avec $code(c_i^{current})$ $(1 \leq i \leq i)$ sans tenir compte de leurs versions.
2. ***créer*** soit le concept $c_i^{current}$ dans l'entrepôt. Il s'agit d'ajouter le concept c_i, s'il n'existe pas dans l'entrepôt. Cette étape consiste à:
   (a) ajouter le concept c_i^{new} dans l'entrepôt, et
   (b) mettre à jour c_i^{new} pour que toutes les références qu'il avait dans la source, soient mises à jour pour qu'il référence les autres concepts de la version courante (notons que ceci peut entrainer de façon récursive l'introduction de tout un ensemble de concepts nouveaux).
3. ***mettre à jour*** le concept $c_i^{current}$. Il s'agit de mettre à jour la version courante du concept c_i, s'il existe déjà dans l'entrepôt :
   (a) **si** $(version(c_i^{current}) > version(c_i^{new}$ ou $(version(c_i^{current}) = version(c_i^{new})$ **alors**: on ne fait rien.
   (b) **si** $version(c_i^{current}) < version(c_i^{new})$ **alors** mettre à jour $c_i^{current}$ par c_i^{new} :
      i. supprimer $c_i^{current}$ de l'entrepôt,
      ii. si c_i appartient à l'ontologie partagée :
         A. charger la dernière version de c_i (voir Figure 4.3)
         B. mettre à jour récursivement ses références.
      iii. si c_i n'appartient qu'à l'ontologie d'une sources :
         A. remplacer $c_i^{current}$ par c_i^{new},
         B. mettre à jour récursivement ses références.

4.4.1.2 Gestion des versions de la partie contenu

Concernant l'implémentation des tables *multi-versionnées*, nous associons chaque table à un ensemble de propriétés qui permet de représenter explicitement à la fois l'origine mais également le cycle de vie des instances dans cette table. Ainsi, une classe racine de toute classe ontologique de l'entrepôt a été implémentée. La classe racine est caractérisée par les propriétés suivantes:

1. "*SupplierCode*" permettant de savoir le code du fournisseur de chaque instance intégrée *i*.
2. "*ClassCode*" permettant de savoir le code de la classe de base (d'origine) de l'instance *i*.
3. "*InstanceCode*" permettant de savoir la clé sémantique de l'instance *i*. La valeur de la propriété *InstanceCode* est constituée, de la représentation sous la forme d'une chaîne de caractères, la clé sémantique d'origine de l'instance *i*.

   Remarquons que dans un entrepôt, le schéma intégré des instances n'est pas leur schéma d'origine. Il est nécessaire donc de choisir, pour chaque schéma intégré, une clé sémantique pouvant identifier d'une façon globale les instances de ce schéma intégré. Dans notre implémentation, ces trois premières propriétés permettent non seulement de tracer l'origine des instances intégrées, mais également de formuler la clé sémantique de toute classe de base de l'entrepôt.
4. "*VersionMin*" qui indique la première version de la population d'origine, pour laquelle *i* est valide.
5. "*VersionMax*" représentant la valeur de la première version de la population d'origine, pour laquelle *i* n'est plus valide.
6. "*SupplierName*" qui représente le nom du fournisseur de *i*.
7. "*ClassName*" qui représente le nom de la classe de base d'origine de *i*.

Notons que les deux dernières propriétés offrent simplement la possibilité d'accéder rapidement le nom de la source et le nom de la classe de base d'origine des instances intégrées.

4.4.2 Entrepôt avec versionnement des instances et historisation ontologique

Il est clair que le mécanisme des versions flottantes fait que l'articulation stockée dans la version courante de l'ontologie d'entrepôt entre une ontologie locale et l'ontologie partagée peut ne pas être sa définition originale.

Dans le cas où il apparaît nécessaire de pouvoir accéder à une instance à travers les définitions ontologiques qui existaient lorsque cette instance était elle-même active, il est nécessaire d'archiver également toutes les versions successives de l'ontologie de l'entrepôt. Cela est utile, pour connaître ce qu'était, à l'époque de l'instance, le domaine précis d'une de ses propriétés énumérées (dont le domaine peut, ensuite, avoir été étendu).

Nous avons également implémenté cette possibilité en offrant d'archiver, dans l'entrepôt OntoDaWa, les définitions de toutes les versions de classes ayant existées dans la vie de l'entrepôt, ainsi que toutes les relations sous leur forme originale.

Cette structure est constituée de quatre parties :

1. L'ontologie courante : elle contient la version flottante de l'ontologie de l'entrepôt. Elle constitue également l'interface générique d'accès aux données.

2. Les tables multiversionnées contiennent toutes les instances intégrées ainsi que les caractéristiques de leurs cycles de vie.

   Les deux premières parties sont les deux parties que nous avons présenté dans l'architecture précédente. Lorsqu'on veut historiser complètement les schémas et les ontologies intégrées, deux nouvelles parties sont ajoutées.

3. L'archivage des ontologies : qui contient toutes les descriptions ontologiques des différentes versions de chaque classe et de chaque propriété de l'ontologie de l'entrepôt. Cette partie fournit aux utilisateurs les vraies définitions des versions de chaque concept si cela leur est nécessaire.

4. L'archivage des schémas : les versions du schéma de chaque ensemble des instances I_i sont également historisées en archivant la fonction $Sch^k(c_i)$ de chaque version k de c_i où c_i est la classe de base des I_i. Dans notre implémentation, cette historisation est faite en conservant une instance unique dont toutes les valeurs seront à "*nulle*" (Instance dite "formelle").

Notons que le principe de la continuité ontologique semble rendre rarement nécessaire ces deux derniers archivages qui sont à la fois complexes et coûteux.

4.5 Conclusion

Dans ce chapitre, nous avons présenté le problème de l'évolution asynchrone des données et des ontologies dans un système d'intégration à base ontologique de type entrepôt de données. Les sources sont autonomes; elles évoluent de façon asynchrone et peuvent étendre ou/et spécialiser, en cas de besoin, l'ontologie de domaine. Notre processus d'intégration intègre d'abord les ontologies puis les données. En l'absence d'évolution, la présence de ces ontologies permet une automatisation complète du processus d'intégration et la résolution des conflits usuels dans l'intégration de bases de données hétérogènes. Lorsque les ontologies évoluent de façon incohérente entre les différentes sources, l'intégration automatique devient impossible. Pour résoudre ce problème, nous avons proposé d'encadrer les évolutions d'ontologies autorisées par le principe de continuité ontologique. Ce principe stipule qu'une ontologie ne peut infirmer un axiome qui se trouvait vérifié dans une version antérieure de l'ontologie. Ce principe nous a alors permis de proposer un mécanisme, dit *de version flottante*, qui permet de ne conserver dans l'entrepôt qu'une seule version de chaque classe et de chaque propriété, appelée version courante (en fait, la plus grande version connue). Ceci permet de consolider entre les versions courantes toutes les relations ayant existé entre les différentes versions des différents concepts. L'ensemble des concepts en version courante constitue l'ontologie courante, et cette ontologie permet d'interpréter toutes les instances existant dans le système d'intégration, quelles que soient les versions de classe pour lesquelles elles avaient été définies.

Nous avons ensuite proposé une possibilité d'historisation des instances figurant dans les différentes sources et réintroduites lors de chaque rafraîchissement de l'entrepôt. Afin d'identifier les instances ontologiques, bien qu'elles puissent, au cours du temps, être décrites par des ensembles différents de propriétés, nous avons proposé la notion de *clé sémantique*. Il s'agit d'un sous-ensemble de propriétés applicables d'une classe qui sont suffisantes pour en identifier les instances tout au long de l'évolution. Cette notion nous a alors permis de proposer une méthode d'historisation des instances qui évite toute duplication: chaque instance n'est représentée qu'une seule fois, elle consolide toutes les propriétés qui

ont existées au cours du temps et deux attributs identifient les versions de classe pour lesquelles l'instance est ou a été valide (instance *multi-versionnée*).

La structure de notre système d'intégration, appelé OntoDaWa, est celle d'un entrepôt à base ontologique, qui référence également l'ontologie partagée et dont les instances sont multi-versionnées. Elle est constituée de quatre parties, à savoir, (1) l'ontologie courante qui contient la version flottante de chacun des concepts de l'ontologie de l'entrepôt, (2) l'archivage des ontologies qui contient, si besoin, toutes les versions des définitions ontologiques de chaque classe et propriété, (3) l'historique des schémas de représentation des instances des différentes classes, et (4) les tables multi-versionnées contenant toutes les instances ainsi que leur première et dernière versions d'activité. Cette structure permet à la fois d'historiser les ontologies, les schémas, et les instances. L'historisation des ontologies peut être évitée dans la plupart des cas, l'ontologie courante suffisant à la fois pour intégrer de façon automatique les sources, et pour accéder à l'ensemble des instances.

Notre travail sur l'évolution a été le premier, à notre connaissance, qui permet à la fois d'intégrer de façon complètement automatique des sources de données disposant d'une grande autonomie de représentation, et de supporter l'évolution asynchrone de ces sources. La plupart des propositions sur les systèmes d'intégration s'intéressent principalement à la construction du schéma intégré et peu à sa évolution. Nous avons proposé un règlement pour encadrer l'évolution des ontologies qui est basé sur le principe de continuité ontologique. Il semble être rigide pour certaines ontologies de domaines, mais pour les ontologies pratiquement utilisées, qui sont celles que nous manipulons, ce principe apparaît tout à fait raisonnable.

Lorsque nous ne sommes pas dans ce cas, il n'existe plus aucune manière d'intégrer automatiquement des ontologies évolutives, les seules approches proposées nécessitent des interventions manuelles.

Deuxième partie

Phase en aval de conception

Chapitre 5

Sélection isolée de structures d'optimisation

Decision-support systems demand speedy access to data, no matter how complex the query–or the data : Charles Bontempo and George Zagelow - IBM [54]

Dans la première partie de ce manuscrit, nous avons présenté une démarche ontologique de construction d'un système d'intégration dans une approche matérialisée. Dans cette partie, nous présentons le deuxième volet de notre travail qui consiste à proposer des structures d'optimisation pour faciliter l'exploitation d'un système d'intégration. Les travaux présentés sur ce volet représentent une activité de recherche débutée en 1996 (avant notre thèse). Les travaux se sont poursuivis pendant ma thèse et ont évolué notamment avec la thèse de Kamel BOUKHALFA, préparée au sein du laboratoire LISI dans le cadre d'une bourse franco-algérienne, co-encadrée par Prof. Guy PIERRA et moi même et le magister de Soumia Benkrid à l'Ecole Nationale d'Informatique d'Alger - Algérie. Ils sont le fruit de collaborations nationales avec Michel et Ana SIMONET du laboratoire TIMC-IMAG (1994-1996) et Michel Schneider du laboratoire LIMOS de Clermont Ferrand, et internationales avec Prof. Karlapalem KAMALAKAR en poste alors à Hong Kong University of Science and Technology, Dr. Mukesh MOHANIA d'IBM d'Inde et Prof. Rokia MISSAOUI, de l'Université du Québec en Outaouais, Canada.

Nos travaux sur la fragmentation horizontale ont été largement étudiés dans le contexte des bases de données traditionnelles (relationnelles et orientées objet) et les entrepôts de données. Un entrepôt de données relationnel peut être vu comme un système d'intégration, où les données des sources sont dupliquées dans une base de données unique. Il est habituellement modélisé par un *schéma en étoile* (ou un *schéma en flocon de neige*) qui est caractérisé par une ou plusieurs tables des faits de taille volumineuse et un certain nombre de tables de dimension plus petites. Sur ce schéma, un ensemble de requêtes décisionnelles de type OLAP est exécuté. La principale caractéristique de ces requêtes est qu'elles imposent des restrictions sur les valeurs des tuples des tables de dimension utilisées pour sélectionner des faits spécifiques; ces faits seront groupés et agrégés selon les demandes des décideurs. Le goulot d'étranglement principal en évaluant de telles requêtes est de joindre une table des faits volumineuse avec les tables de dimension. En plus de la volumétrie et de la complexité des requêtes, les décideurs qui exigent un temps de réponse de requêtes *raisonnable*. Ces caractéristiques ont largement contribué à faire évoluer la conception physique traditionnelle dédiée aux applications de type OLTP. Pour offrir une meilleure utilisation de l'entrepôt de données, la *conception physique* devient un enjeu important [69].

Nous avons vu dans le chapitre 1 qu'il existe une large panoplie de structures d'optimisation qui peuvent être sélectionnées durant la conception physique d'un entrepôt de données. Afin de mieux comprendre cette sélection, nous présentons dans la section suivante une formalisation du problème de la conception physique.

5.1 Problème de la conception physique

Étant donnés :

- Une charge de requêtes $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_m\}$, où chaque requête Q_j possède une fréquence d'accès f_j;
- Un ensemble de structures d'optimisation $\mathcal{SO} = \{SO_1, SO_2, \cdots, SO_l\}$ supportées par le SGBD hôte utilisé pour répondre aux requêtes ;
- Un ensemble de contraintes liées à SO : $Cont = \{Cont_1, Cont_2, \cdots, Cont_l\}$, où chaque contrainte $Cont_i$ $(1 \leq i \leq l)$ est associée à une structure d'optimisation SO_i[7].

le problème de la conception physique ($\mathbb{PCP}$) consiste à sélectionner des schémas de structures d'optimisation afin de réduire le coût d'exécution de la charge de requêtes $\mathcal{Q}$ en présence de ces derniers et satisfaire les contraintes définies.

La résolution de ce problème nécessite une exploitation de l'espace de recherche englobant tous les sous espaces correspondants aux structures d'optimisation. Soit Ins_i le nombre d'instances de la structure d'optimisation SO_i. L'espace de recherche global est $2^{\sum_{i=1}^{l} Ins_i}$, du fait que les différentes instances peuvent interagir [225].

Dans la pratique, pour résoudre ce problème complexe, l'administrateur doit effectuer les étapes suivantes: *(a)* choisir une ou plusieurs structures d'optimisation parmi $\mathcal{SO}$ supposé(es) pertinente(s) pour optimiser cette charge de requêtes $\mathcal{Q}$ (ce choix peut se faire manuellement en se basant sur son expérience et les caractéristiques des requêtes), *(b)* identifier le mode de sélection des schémas d'optimisation de ces dernières dans le cas où il décide d'utiliser plusieurs structures et *(c)* choisir les algorithmes pour les sélectionner. Dans les sections suivantes, nous détaillons les deuxième et troisième étapes.

5.1.1 Modes de sélection des schémas de structures d'optimisation

En fonction du cardinal de l'ensemble de structures d'optimisation SO, deux modes de sélection peuvent être envisagés :

1. $||SO|| = 1$: cela signifie que l'administrateur (le concepteur) souhaite sélectionner une seule structure d'optimisation pour satisfaire sa charge de requêtes. Cette sélection est alors appelée *sélection isolée* [56].
2. $||SO|| > 1$: cela signifie que l'administrateur opte pour plusieurs structures d'optimisation. Cette sélection est appelée *la sélection multiple* [56].

La sélection isolée a connu beaucoup d'intérêt dans le contexte de bases de données traditionnelles. Mais, elle a montré des limites dans le contexte des bases de données volumineuses avec des requêtes

[7] Pour plus de simplicité, nous supposons que chaque structure a une seule contrainte.

complexes (appelées *méga requêtes* [196]). Pour offrir aux concepteurs une meilleure optimisation des requêtes, certains travaux se sont intéressés à la sélection multiple. Nous avons distingué trois implémentations de cette sélection (voir chapitre 1) : (i) une implémentation indépendante, (ii) une implémentation conjointe et (iii) une implémentation séquentielle. Cette dernière semble la plus pertinente dans le cadre des entrepôts de données, car la majorité des structures d'optimisation présentent de fortes dépendances non mutuelles.

Dans nos travaux sur l'optimisation physique, nous nous sommes intéressés à la fois aux sélections isolée et multiple. La sélection isolée a concerné trois structures d'optimisation redondantes : la *fragmentation horizontale primaire*, la *fragmentation dérivée* et *l'allocation sans réplication* et les deux structures non redondantes : les *index* et les *vues matérialisées*. Tandis que la sélection multiple a concernée la fragmentation dérivée et les index de jointure binaires, la distribution de l'espace de stockage entre les vues matérialisées et les index et la fragmentation et l'allocation dans les entrepôts de données parallèles.

Certains de nos travaux ont été étudiés conjointement dans les contextes des bases de données traditionnelles et les entrepôts de données. On peut ainsi citer, la fragmentation horizontale et l'allocation.

Dans ce chapitre, nous présentons deux exemples de la sélection isolée: la fragmentation horizontale et les index multi table. Pour chaque exemple, nous présentons une démarche de son utilisation dans les entrepôts de données relationnels ainsi que les algorithmes de leurs sélections.

5.1.2 Fragmentation horizontale dans les bases et entrepôts de données

Publications [42, 33, 34, 25, 26, 35].

La fragmentation horizontale est une structure d'optimisation qui a été étudiée massivement dans les différentes générations de bases de données : les *bases de données relationnelles*, les *bases de données orientées objet* et les *entrepôts de données*. Initialement proposée comme une technique de conception logique des bases de données réparties dans les années 80 [64, 65, 168], la fragmentation horizontale consiste à partitionner une table en fonction de ses n-uplets de façon à réduire le nombre des accès non nécessaires pour le traitement de certaines requêtes. Deux types de fragmentation horizontale existent : *primaire* et *dérivée* [64, 169]. La fragmentation horizontale primaire d'une table se base sur des attributs définis sur cette table. La fragmentation horizontale dérivée consiste à propager la fragmentation d'une table sur une autre table. Cette propagation n'est possible que si un lien *père-fils* existe entre deux tables. Par conséquent, la fragmentation horizontale dérivée d'une table se base sur les attributs définis sur une ou plusieurs autres tables [26, 64].

Contrairement aux autres structures d'optimisation (comme les index et les vues matérialisées), où la sélection des schémas d'optimisation se fait généralement lorsque la base de données est opérationnelle (ou créée), la sélection d'un schéma de fragmentation d'une base de données (ou un entrepôt de données) doit se décider avant sa création [43]. Cette situation rend sa sélection plus sensible que les autres structures. De plus, elle peut également être combinée avec d'autres structures d'optimisation comme les index, les vues matérialisées [191] et le traitement parallèle [200]. Plusieurs travaux se sont intéressés à la fragmentation horizontale. Ils proviennent de deux milieux différents : académique et industrielle.

1. *Milieu académique* : La fragmentation horizontale a connu beaucoup d'attention de la part des chercheurs durant les différentes générations de bases de données : les bases de donnée relation-

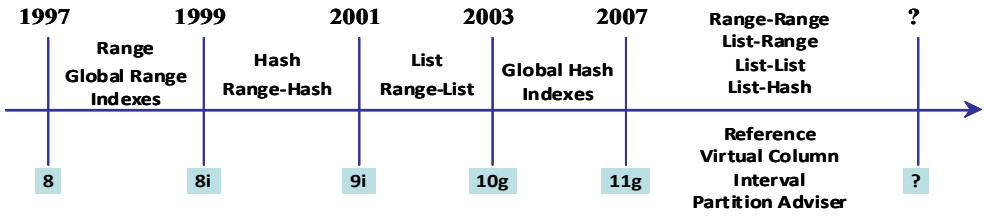


FIG. 5.1 – Évolution de la fragmentation horizontale sous Oracle

nelles [64, 169, 83, 183], les bases de données orientées objet [129, 130, 33, 26, 25, 35, 86, 184] et les entrepôts de données [10, 41, 161]. L'idée sous-jacente à la fragmentation horizontale est de pouvoir constituer des ensembles de données partielles dont les n-uplets (ou instances d'objet) ont des propriétés géographiques communes. Le mot *géographique* peut être perçu à une échelle différente selon le contexte d'utilisation: *centralisé*, *répartie*, et *parallèle*. Dans le contexte centralisé, la fragmentation horizontale permet de réduire le nombre de n-uplets accédés inutilement. Dans le contexte réparti, le principe reste le même, seule, l'interprétation de la notion de propriétés géographiques change. Elle consiste à découper une relation par rapport à un ensemble de sites. Dans un tel contexte, deux phases se distinguent lorsqu'on parle de la fragmentation : la phase de partitionnement en elle-même et la phase *d'allocation des fragments sur les sites*. Dans le contexte parallèle, la fragmentation consiste à partitionner les données d'une table afin de pouvoir les traiter en parallèle et ainsi diminuer le temps de réponse des requêtes. Il n'est donc plus question de rassembler les données utilisées simultanément, mais au contraire de les disséminer pour obtenir un taux important de parallélisme pour l'exécution d'une même opération. La fragmentation horizontale permet l'augmentation du parallélisme inter-requêtes et intra-requêtes [55]. La fragmentation horizontale peut engendrer le problème de *migration d'instances* si des attributs instables comme *Age* sont utilisés pour fragmenter les tables [169].

Dans les entrepôts de données, la fragmentation horizontale a été introduite comme une alternative importante de conception physique [191]. Elle permet de partitionner les tables de l'entrepôt (tables de dimension et/ou table des faits) [10, 41], les vues matérialisées et les index [191] en un ensemble de fragments de plus petite taille. Elle a été utilisée essentiellement pour améliorer la performance des requêtes et la gestion de l'entrepôt de données.

2. *Milieu industriel* : Depuis qu'elle est considérée importante dans la conception physique, les éditeurs de SGBD commerciaux (Oracle, DB2, SQL Server, Sybase, etc.) ou non commerciaux (PostgreSQL, MySql, etc.) s'intéressent de plus en plus à la fragmentation en proposant des commandes au niveau du langage de définition de données (DDL) pour la supporter. Plusieurs modes de fragmentation ont été proposés. Actuellement, la plupart des SGBD supporte la fragmentation horizontale et offre des commandes pour fragmenter les objets de la base de données et manipuler les partitions obtenues. La figure 5.1 montre l'évolution de la fragmentation horizontale dans le SGBD Oracle qui offre le plus de modes de fragmentation. Cette évolution a été motivée par l'apparition des entrepôts de données (où les données sont historisées) et les applications décisionnelles exigent un temps de réponse raisonnable pour traiter les requêtes.

   Le premier mode de partitionnement supporté par Oracle a été le partitionnement par intervalle (*Range*) dans Oracle8i. Oracle9 et 9i ont ajouté d'autres modes de fragmentation qui sont: le par-

titionnement par hachage (*Hash*) le partitionnement par liste (*List*) et le partitionnement composé (*Range-Hash* et *Range-List*)[8]. Les partitionnements *Range, List* et *Hash* sont des modes de base supportés par la plupart des SGBDs commerciaux.

Récemment, Oracle 11g a fait évoluer la fragmentation horizontale en proposant plusieurs modes : (1) le partitionnement par une colonne virtuelle (*virtual column partitioning*) dans lequel, une table est fragmentée en utilisant un attribut virtuel, qui est défini par une expression utilisant un ou plusieurs attributs. Cette colonne est stockée seulement dans les méta-données. (2) Le partitionnement par référence (*referential partitioning*) : permet de fragmenter une table en utilisant une autre table (à condition qu'il y ait une relation de type père-fils entre les deux tables [85]). Ce partitionnement est similaire à la *fragmentation horizontale dérivée*, mais son inconvénient majeur est qu'une table est fragmentée en fonction d'une seule autre table. Dans les entrepôts de données, une table des faits doit être fragmentée en utilisant les schémas de fragmentation de plusieurs tables de dimension pour garantir une optimisation des requêtes complexes. (3) Toutes les combinaisons de modes de base c'est-à-dire, Range, List et Hash sont possibles: *Range-Range, List-List, Hash-Hash, List-Hash*, etc.

Étant donné que la fragmentation horizontale préserve le schéma logique des tables et des vues, elle implique que toutes les opérations effectuées sur les objets d'origine sont aussi applicables sur leurs partitions. Plusieurs opérations ont été définies pour manipuler des partitions. Nous pouvons ainsi citer l'opération de fusion de deux partitions (*MERGE PARTITION*), l'opération d'éclatement d'une partition en deux partitions (SPLIT PARTITION), l'opération de conversion d'une partition en une table (*EXCHANGE PARTITION*), etc.

Cette évolution rapide de la fragmentation horizontale nous a poussée à l'étudier en détail. En explorant les principaux travaux académiques et industriels sur la manière de sélectionner des schémas de fragmentation horizontale, nous constatons que cette sélection suppose une décomposition du domaine des valeurs d'attributs participant au processus de fragmentation[9]. Cette décomposition peut être réalisée de deux manières: (1) une décomposition manuelle et (2) une décomposition automatique.

1. Dans la première catégorie, l'administrateur décompose manuellement le domaine de valeurs de chaque attribut de fragmentation en se basant sur ses connaissances des applications et *impose a priori* le nombre de fragments horizontaux générés. Les principaux inconvénients de ce mode de partitionnement sont : (i) l'absence d'une métrique garantissant l'efficacité du schéma de fragmentation obtenu et (ii) la manière de décomposer chaque domaine en plusieurs sous domaines. L'exemple ci-dessous montre un schéma de fragmentation de la table *Temps* en quatre fragments en utilisant l'attribut *Saison*.

   Exemple.

```
CREATE TABLE TEMPS
(TID NUMBER, Saison VARCHAR2(10), Année Number)
PARTITION BY LIST(Saison)
(PARTITION Time_Summer VALUES('Été'),
PARTITION Time_Spring VALUES ('Printemps'),
PARTITION Time_Autumn_Winter VALUES('Automne', 'Hiver'));
```

[8]Ces modes de partitionnement sont aussi supportés par les autres SGBDs commerciaux comme SQL Server, Sybase, DB2, etc.

[9]La décomposition des domaines de valeurs des attributs d'une table implique une fragmentation horizontale de cette table.

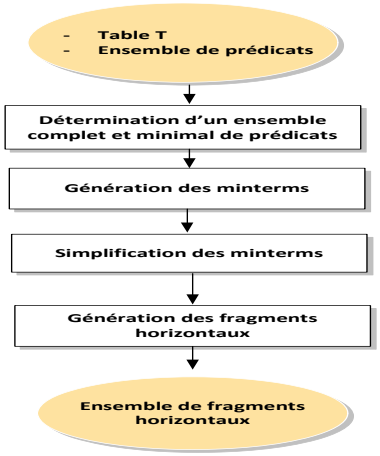


FIG. 5.2 – Approche basée sur les prédicats

2. Dans le partitionnement automatique (pour le mode Range et List), les domaines des valeurs des attributs de fragmentation sont décomposés en se basant sur les requêtes définies sur le schéma de la base de données. Le principal inconvénient de cette catégorie, est l'absence de contrôle du concepteur sur le nombre de fragments générés.

Dans la section suivante, nous étudions la fragmentation horizontale dans les bases de données traditionnelles et les entrepôts de données relationnels.

5.1.2.1 Algorithmes de fragmentation horizontale dans les bases de données traditionnelles

Dans le cadre des bases de données traditionnelles (relationnelles et orientées objet) les travaux sur la fragmentation horizontale se sont largement concentrés sur le développement des algorithmes de partitionnement primaire. Nous avons assisté à la proposition de plusieurs algorithmes de fragmentation horizontale, que nous classons en trois principales catégories [22]: les *algorithmes dirigés par les prédicats*, les *algorithmes dirigés par les affinités* et les *algorithmes dirigés par un modèle de coûts*. Tout algorithme de fragmentation a comme entrées : une table T à fragmenter et un ensemble de prédicats ou de requêtes les plus fréquentes $\mathbb{Q}$ ainsi que leurs fréquences d'accès. En sortie, un schéma de fragmentation horizontale de la table[10] est obtenu.

5.1.2.1.1 Algorithmes dirigés sur les prédicats Ce sont les premiers algorithmes de la fragmentation horizontale proposés dans le cadre des bases de données réparties [64, 169]. Ils sont basés sur la génération des minterms à partir des prédicats de sélection utilisés par les requêtes. Un minterm est la conjonction de prédicats de sélection et leurs négations. Quatre étapes principales les caractérisent : (1) la génération d'un ensemble *complet* et *minimal* de prédicats, (2) la génération de l'ensemble des minterms, (3) la simplification des minterms et (4) la génération des fragments.

L'approche basée sur les prédicats est caractérisée par une grande complexité par rapport au nombre de prédicats. Pour n prédicats simples, cette approche génère 2^n minterms. Son utilisation est donc limitée

[10]Un schéma de fragmentation est le résultat du processus de fragmentation

à un nombre de prédicats raisonnable.

5.1.2.1.2 Algorithmes basés sur les affinités entre prédicats Cette approche a été proposée dans le souci de réduire la complexité de l'approche basée sur les prédicats [223, 129]. Nous avons proposé un algorithme adaptant les travaux de Navathe [157] sur la fragmentation verticale [33, 35]. Il est composé de cinq étapes : (1) l'énumération des prédicats simples, (2) la construction de la matrice d'usage des prédicats, (3) la génération de la matrice d'affinité des prédicats, (4) le regroupement des prédicats et (5) la génération des fragments horizontaux.

La complexité de cet algorithme est : $O(m \times n + n^2)$ [10, 33, 35] (où m et n représentent respectivement le nombre de requêtes et le nombre de prédicats simples contenus dans ces requêtes). Cette approche est donc moins complexe que celle basée sur les prédicats dont la complexité est $O(2^n)$. Mais, elle ne prend en considération que la fréquence d'accès notamment critère de regroupement. Or, pour fragmenter une base de données, d'autres paramètres doivent être pris en compte, comme les facteurs de sélectivité des prédicats, la taille de la table, la taille de page disque, le nombre de pages occupées par cette table, etc.

Les approches basées sur les prédicats et sur les affinités ne fournissent aucune métrique permettant d'évaluer la qualité du schéma de fragmentation obtenu. Pour pallier ce problème, nous avons proposé dans le cadre de nos travaux de thèse une nouvelle approche de fragmentation basée sur un modèle de coût [25, 35, 10]. Grâce à ce dernier, il est possible d'évaluer la qualité de la solution fournie. Notons qu'un modèle de coût est la composante principale d'un optimiseur physique d'un base de données [97].

5.1.2.1.3 Algorithmes basés sur un modèle de coût Afin d'offrir aux concepteurs la possibilité de quantifier la qualité de leurs schémas de fragmentation, nous avons proposé un algorithme de fragmentation horizontale utilisant un modèle de coût dans le contexte des bases de données orientées objet [10, 25, 35]. Les composantes principales de cet algorithme sont: (i) *un générateur de schémas de fragmentation*, (ii) un *modèle de coût* et (iii) une *sélection du schéma optimal*.

1. *Générateur de schémas de fragmentation :* Cette étape consiste à générer tous les schémas de fragmentation possibles, d'une classe d'un schéma d'une base de données orientée objet. Cette génération est guidée par l'ensemble de prédicats de sélection défini dans la charge de requêtes.
2. *Modèle de coût:* Il constitue le noyau de ce type d'algorithmes. Il peut être vu comme une fonction d'une seule variable représentant un schéma de fragmentation S_i et attribuant le coût d'exécution $C(S_i)$ d'un ensemble de requêtes. Il prend en considération la taille des tables, le nombre de pages, les facteurs de sélectivité des prédicats, fan-out, etc. L'optimalité et la qualité des solutions obtenues par ces algorithmes sont liées à ce modèle de coût.
3. *Sélection du schéma de fragmentation optimal:* détermine le schéma de fragmentation S_{min} garantissant la meilleure performance parmi tous les schémas possibles.

Pour mettre en oeuvre cet algorithme, nous avons développé un modèle de coût estimant le nombre d'entrées-sorties nécessaires pour exécuter un ensemble de requêtes dans le contexte des bases de données orientées objet [10]. Cet algorithme peut être coûteux, car le nombre de schémas de fragmentation générés peut être très grand [10]. Pour réduire cette complexité, nous avons proposé un algorithme approximatif basé sur la technique de l'escalade (hill-climbing dans l'appellation anglo-saxonne) et un modèle de coût [10]. Pour amorcer ce type d'algorithme, une solution initiale est nécessaire (qui peut

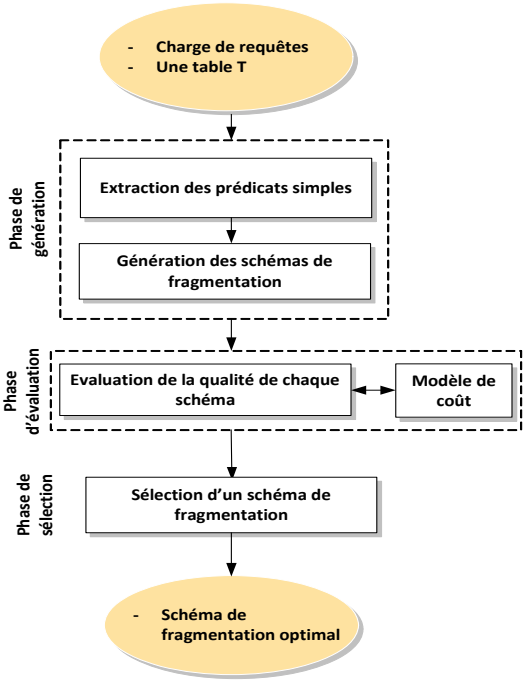


FIG. 5.3 – Approche basée sur un modèle de coût

être obtenue en appliquant l'algorithme dirigé par les affinités). La solution initiale doit avoir une complexité raisonnable. Ensuite il effectue des changements graduels sur cette solution pour se rapprocher d'un *but*. Le but est déterminé par une fonction objectif que nous désirons maximiser ou minimiser.

En se basant sur cette analyse de complexité, nous avons proposé des recommandations aux utilisateurs :

si le nombre de prédicats est faible, l'algorithme exhaustif est utilisé, sinon, l'algorithme approximatif est utilisé.

5.1.2.1.4 Le modèle de coût pour évaluer la qualité des schémas de fragmentation Notre modèle de coût peut avoir une autre utilisation. IL peut être utilisé comme un moyen de quantification de la qualité de tout schéma de fragmentation obtenu par n'importe quel algorithme de fragmentation. Pour mesurer cette qualité, nous définissons un *facteur de qualité* FQ comme suit :

$$FQ = \frac{CF}{CNF} \tag{5.1}$$

où CN et CNF représentent respectivement le coût d'exécution d'une charge de requêtes sur un schéma de base de données fragmenté et le coût d'exécution de la même charge sur un schéma non fragmenté. Si le concepteur partitionne sa base de données manuellement ou en utilisant n'importe quel algorithme appartenant aux deux premières catégories (dirigés par les prédicats ou dirigés par les affinités), il peut à l'aide d'un modèle de coût quantifier sa réduction en utilisant le facteur de quantification (FQ). Si ce facteur est inférieur à un certain seuil fixé, il peut opter pour le schéma généré.

5.1.2.2 Bilan et discussion

Le tableau 5.1 résume les principaux travaux effectués sur la fragmentation horizontale. Il montre pour chaque travail le type de fragmentation effectué (horizontale primaire, horizontale dérivée), l'approche de fragmentation suivie (basée prédicats, affinité ou modèle de coût), l'algorithme utilisé, le modèle de coût utilisé et le contexte dans lequel les travaux ont été proposés : bases de données relationnelles (BDR), bases de données orientées objet (BDOO).

Travaux	*Type de fragmentation*	*Approche*	*Algorithme*	*Modèle de coût*	*Contexte*
Ceri et al. [64]	Primaire	Basée sur les prédicats		Non	BDR
Zhang et al. [223]	Primaire	Basée sur les affinités	Bond Energy Algorithm + Groupement graphique	Non	BDR
Ozsu et al. [168]	Primaire	Basée sur les prédicats		Non	BDR
Bellatreche et al. [10]	Primaire et dérivée	Basée sur un modèle de coût	Hill Climbing & Exhaustif	Mathématique	BDOO

TAB. 5.1 – Comparaison des travaux effectués sur la fragmentation horizontale

En se basant sur ces travaux, nous formalisons le problème de sélection de schéma de fragmentation horizontale dans le contexte des bases de données traditionnelles :

Étant donnés :
- Une table T (ou une classe C) d'un schéma d'une base de données
- Une charge de requêtes $\mathbb{Q} = \{Q_1, Q_2, \cdots, Q_m\}$, où chaque requête Q_j possède une fréquence d'accès f_j. Ses requêtes peuvent être exécutées sur des sites.

Le problème de la sélection de schéma de fragmentation horizontale consiste à fragmenter la table T en fragments optimisant le traitement des requêtes. Cette formalisation ne *contrôle pas le nombre de fragments générés*. En d'autres termes, quand le concepteur exécute son algorithme de fragmentation favori quelle que soit sa catégorie, il ne découvre, qu'à la fin, le nombre de fragments généré, bien que ce processus de fragmentation impose au concepteur le nombre de fragments qu'il souhaite avoir. Mais aucune métrique n'est utilisée pour quantifier l'intérêt de cette fragmentation. Ce constat nous a poussé à proposer des algorithmes de fragmentation dans le contexte des entrepôts de données contraints par le nombre de fragments générés. Ces travaux rentrent dans le cadre de la thèse de Kamel BOUKHALFA [56].

5.1.3 Sélection de schéma de fragmentation horizontale dans les entrepôts de données

Publications [9, 8, 31, 30, 10, 15, 16, 57, 17, 21, 43, 11, 22, 12, 13].

Nos travaux sur la fragmentation horizontale dans les entrepôts de données ont débuté au début des années 2000, par la proposition d'une démarche de fragmentation horizontale d'un entrepôt de données. Ensuite, nous nous sommes intéressés à la proposition des algorithmes efficaces et contrôlant le nombre de fragments générés.

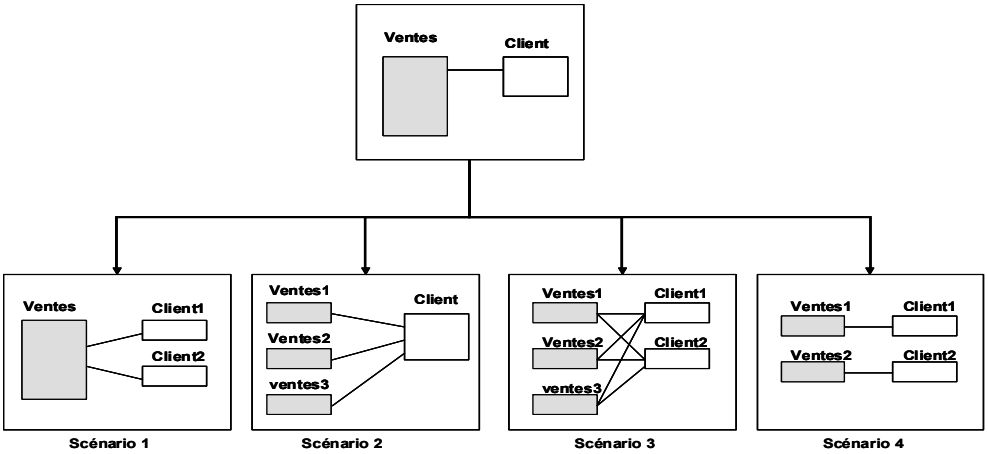


FIG. 5.4 – Scenarii de fragmentation

5.1.3.1 Scénarii de fragmentation horizontale pour les entrepôts de données relationnels

Un entrepôt de données est souvent modélisé par un schéma en étoile ou une de ses variantes. Il est souvent un concept centré-requête, par opposition au schéma centré-mise à jour employé par les applications de type OLTP. Les requêtes typiques de ce schéma sont appelées les *requêtes de jointure en étoile* (star-join queries) qui ont les caractéristiques suivantes:

1. Il y a des jointures multiples entre la table des faits et les tables de dimension.
2. Il n'y a pas de jointure entre les tables de dimensions.
3. Chaque table de dimension impliquée dans une opération de jointure a plusieurs prédicats de sélection sur ses attributs descriptifs.

La présence des prédicats de sélection dans les requêtes rend l'utilisation de la fragmentation horizontale intéressante. Notons au passage que le banc d'essai *Transaction Processing Concil (TPC-H) www.tpc.org)*, dans ses règles d'implémentation, recommande l'utilisation de la fragmentation horizontale.

Rappelons que dans les bases de données traditionnelles, les algorithmes de fragmentation horizontale concernent une table uniquement. Dans le contexte des entrepôts de données, deux types de tables existent : des *tables de dimension* de taille raisonnable et *une table des faits* volumineuse. Vue cette situation, le concepteur ou l'administrateur est donc amené à faire deux choix pour partitionner un entrepôt de données [10]: (i) le choix des tables à partitionner (dimension ou faits) et (ii) le type de partitionnement (primaire ou dérivée). Ces différents choix nous amène à étudier quatre principaux scénarii [10, 30]: (1) fragmenter seulement certaines ou toutes les tables de dimension par la fragmentation primaire, (2) fragmenter seulement la table de faits par la fragmentation primaire, (3) fragmenter certaines ou toutes les tables de dimension et la table des faits par la fragmentation primaire et (4) fragmenter certaines ou toutes les tables de dimension par la fragmentation primaire et fragmenter la table des faits par la fragmentation dérivée en utilisant les schémas de fragmentation des tables de dimension. Figure 5.4 présente les quatre scénarii de la fragmentation.

1. Fragmenter seulement les tables de dimensions en utilisant la fragmentation horizontale primaire.

Ce choix n'est pas souhaitable pour les requêtes décisionnelles pour deux raisons: (i) la taille des tables de dimensions est généralement inférieure à celle de la table des faits et (ii) les requêtes décisionnelles accèdent à la table des faits par des opérations de jointures coûteuses [138]. Par conséquent, toute fragmentation ne prenant pas en considération la table des faits est à exclure.

2. Partitionner seulement la table des faits en utilisant la fragmentation primaire. Notons que la table des faits est composée de clés étrangères des tables de dimension et des mesures numériques, comme le montant des ventes, le nombre d'articles soldés, ..., etc.. Généralement, dans une requête décisionnelle, nous trouvons rarement des prédicats de sélection définis sur des attributs d'une table des faits [161]. Par conséquent, ce scénario est écarté pour notre processus de fragmentation.

3. Fragmenter les tables de dimension et la table des faits indépendamment. Ce scénario constitue une combinaison des deux précédents scenarii. Comme le premier scénario, les opérations de sélection peuvent être optimisées, par contre, la fragmentation primaire de la table des faits ne prend pas en considération ses liens avec les tables de dimensions. Par conséquent, les opérations de jointure ne sont pas toujours optimisées par ce scénario.

4. Fragmenter totalement ou partiellement les tables de dimensions en utilisant la fragmentation primaire et propager leurs schémas de fragmentation sur la table des faits[11]. Ce choix est bien adapté aux entrepôts, car il prend en considération les exigences des requêtes décisionnelles, ainsi que les relations entre les tables de dimension et la table des faits. Ce scénario a été adopté pour notre étude.

Exemple. Pour illustrer le scénario adopté pour notre démarche de fragmentation horizontale, nous considérons un schéma en étoile avec trois tables de dimension : *Client*, *Temps* et *Produit* et une table de faits *Ventes*. Supposons que la table de dimension *Client* est fragmentée en deux fragments *Client1* et *Client2* définis par les clauses suivantes : $Client_1 = \sigma_{Sexe='M'}(Client)$ et $Client_2 = \sigma_{Sexe='F'}(Client)$. La table de faits Ventes peut être fragmentée en deux fragments Ventes1 et Ventes2 définis par : $Ventes_1 = Ventes \ltimes Client_1$ et $Ventes_2 = Ventes \ltimes Client_2$, où $\ltimes$ représente l'opération de semi-jointure. Ce processus de fragmentation génère deux sous schémas en étoile S_1 et S_2 tels que :
S_1 : $(Ventes_1, Client_1, Produit, Temps)$ (activités de ventes réalisées par les clients masculins seulement) et S_2 : $(Ventes_2, Client_2, Produit, Temps)$ (activités de ventes réalisées par les clientes seulement).

5.1.3.2 Méthodologie de fragmentation horizontale d'un entrepôt de données

Pour partitionner un entrepôt de données relationnel en utilisant le dernier scénario, nous pouvons naïvement utiliser un des algorithmes définis dans le contexte traditionnel, où chaque table de dimension candidate à la fragmentation est décomposée *indépendamment* des autres et leurs schémas sont propagés sur la table des faits. Cette solution a cependant deux inconvénients principaux : (i) la non exploitation des caractéristiques des requêtes de jointure en étoile qui sont souvent définies sur toutes les tables de dimension et la table des faits et (ii) le nombre de fragments de la table des faits peut être important. Soit

[11] La table des faits est alors fragmentée en utilisant la fragmentation dérivée

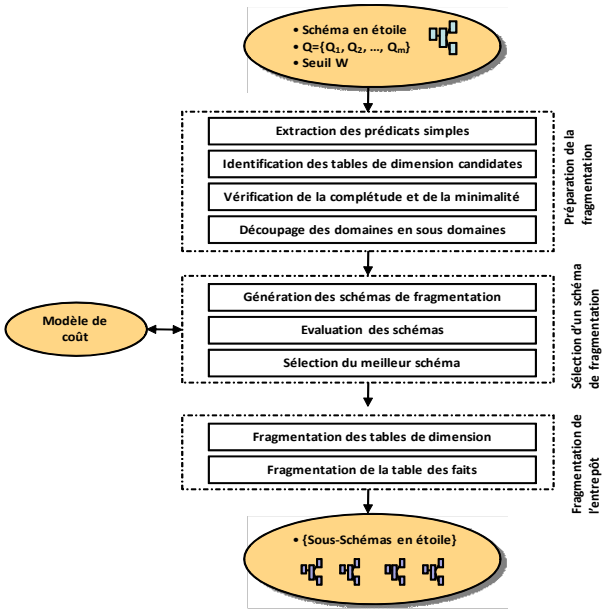


FIG. 5.5 – Démarche de fragmentation

un entrepôt de données composé d'une table de faits F et d tables de dimension $\{D_1, D_2, \cdots, D_d\}$. Supposons que g tables de dimension sont fragmentées ($g \leq d$) et leurs schémas ont utilisés pour partitionner la table des faits. Le nombre de sous-schémas en étoile généré est calculé par l'équation 5.2.

$$N = \prod_{i=1}^{g} m_i \tag{5.2}$$

où m_i représente le nombre de fragments de la table de dimension D_i.

Exemple. Supposons que les trois tables de dimension *Client*, *Produit* et *Temps* sont fragmentées comme suit : *Client* est fragmentée en 100 fragments en utilisant l'attribut *Ville*; *Produit* en 80 fragments en utilisant l'attribut *type de produit* et *Temps* en 48 fragments en utilisant l'attribut *mois*[12].

Par conséquent, la table des faits est fragmentée en $100 \times 80 \times 48 = 384\,000$ fragments de faits (et en 384 000 sous-schémas en étoile). Gérer et maintenir ce nombre important de sous-schémas devient une tâche très difficile pour l'administrateur.

En se basant sur cette analyse, nous réclamons que tout algorithme de fragmentation dans les entrepôts de données doit prendre en considération les tables de dimension candidates au partitionnement tout en offrant aux concepteurs la possibilité de contrôler le nombre de fragments générés tout en optimisant l'ensemble de requêtes.

Pour réaliser ces objectifs, nous avons proposé une approche de fragmentation offrant aux concepteurs la possibilité de choisir le nombre de sous-schémas qu'il souhaite avoir en imposant un seuil W [15, 16, 57]. Nous avons constaté que le contrôle du nombre de fragments passe obligatoirement par le contrôle soit du nombre de tables de dimension fragmentées (g), soit du nombre de fragments (m_i) de chaque table de dimension D_i, soit les deux paramètres. Dans la thèse de Kamel BOUKHALKA,

[12] Pour analyser les ventes effectuées durant 4 ans

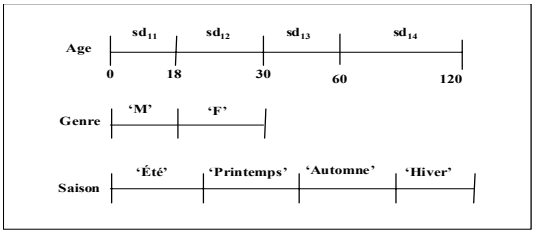


FIG. 5.6 – Découpage des domaines en sous-domaines

nous avons agi sur les deux paramètres afin de donner la même chance aux tables de dimension d'être fragmentées [56].

5.1.3.2.1 La fragmentation d'une table de dimension implique le partitionnement des domaines de valeurs de ses attributs Soient D_i et AF_{D_i} une table de dimension et ses attributs de fragmentation. L'une des conséquences du processus de fragmentation de la table D_i (quand les modes de partitionnement sont *Range* et *List*) est le découpage de domaine de valeurs de chacun de ses attributs en partitions.

Exemple. Pour illustrer le découpage, supposons que les tables de dimension *Client* et *Temps* sont fragmentées en utilisant *Age* et *Saison*. Le domaines de valeurs de ces derniers sont:
$Dom(Age) = [0, 120]$ et $Dom(Saison) =< Été\ Automne, Hiver, Printemps >$.
Supposons que le concepteur a opté pour la fragmentation suivante:
$Client_1 : \sigma_{(Age \leq 18)}(Client)$; $Client_2 : \sigma_{(Age>18) \wedge (Age \leq 30)}(Client)$; $Client_3 : \sigma_{(Age>30 \wedge Age \leq 60)}(Client)$; $Client_1 : \sigma_{(Age>60)}(Client)$;
$Temps_1 : \sigma_{Saison=''Automne''}$, $Temps_2 : \sigma_{Saison=''Hiver''}$, $Temps_3 : \sigma_{Saison=''Printemps''}$ et $Temps_4 : \sigma_{Saison=''Eté''}$.
Par conséquent, le domaine de l'attribut *Age* est décomposé en quatre sous-domaines définis comme suit $Dom(\text{Age}) = d_{11} \cup d_{12} \cup d_{13} \cup d_{14}$, avec $d_{11} =]0, 18]$, $d_{12} =]18, 30]$, $d_{13} = [30, 60]$ et $d_{14} =]60, 120]$. De la même manière le domaine *Saison* est découpé en sous-domaines comme suit : $Dom(\text{Genre}) = d_{21} \cup d_{22}$, avec $d_{21} = \{'M'\}$, $d_{22} = \{'F'\}$ et $Dom(\text{Saison}) = d_{31} \cup d_{32} \cup d_{33} \cup d_{34}$, avec $d_{31} = \{'Printemps'\}$ et $d_{32} = \{'Eté'\}$, $d_{33} = \{'Automne'\}$ et $d_{34} = \{'Hiver'\}$. Les différents sous-domaines des trois attributs de fragmentation sont représentés sur la figure 5.6.

Le découpage des domaines de valeurs des attributs de fragmentation nous donne le nombre de fragments m_i d'une table de dimension fragmentée D_i ($1 \leq i \leq g$) comme le montre l'équation suivante :

$$m_i = \prod_{j=1}^{r_i} n_{ij} \tag{5.3}$$

où r_i et n_{ij} représentent respectivement le nombre d'attributs de fragmentation de la table D_i et le nombre de sous-domaines de l'attribut A_j de la table D_i.

D'après l'équation 5.3, pour contrôler le nombre de fragments de chaque table de dimension D_i, il faut agir soit sur le nombre d'attributs de fragmentation (r_i) soit sur le nombre de sous-domaines de chacun de ces attributs(n_{ij}) soit sur les deux. Dans nos travaux, nous avons agi sur les deux paramètres en même temps. Pour réduire le nombre de fragments, le concepteur peut fusionner deux ou plusieurs

sous domaines d'un attribut de fragmentation. Pour augmenter le nombre de fragments, il peut éclater une partition en plusieurs sous partitions. Les opérations d'éclatement et fusion peuvent être contrôlées par un *modèle de coût estimant le nombre d'entrées sorties nécessaires pour exécuter un ensemble de requêtes*.

Afin de supporter les opérations de fusion et d'éclatement, nous avons proposé un codage représentant tout schéma de fragmentation. Il peut être vu comme un tableau multidimensionnel (que nous appelons *CODE*). Chaque ligne de ce tableau représente le découpage de domaine de valeurs de chaque attribut. Son remplissage se fait de la manière suivante :
$CODE[i][j] = k$ si le sous-domaine sd_j de l'attribut A_i appartient à la partition P_k de cet attribut, avec $1 \leq i \leq n, 1 \leq j \leq n_i, 1 \leq k \leq n_i$

Deux sous-domaines sd_j et sd_l appartiennent à la même partition si et seulement si $CODE[i][j] = CODE[i][l]$. Si toutes les cellules d'une ligne contiennent la même valeur, cela signifie que le domaine de l'attribut correspondant n'est pas partitionné et par conséquent cet attribut ne participe pas dans le processus de fragmentation.

Nous proposons dans la section suivante, une formalisation de ce problème, nous présentons sa complexité et des algorithmes que nous avons développés.

5.1.3.3 Problème de sélection d'un schéma de fragmentation horizontale

Nous formalisons le problème de sélection d'un schéma de fragmentation horizontale ($\mathbb{PSSFH}$) comme suit:
Étant donné :

1. un entrepôt de données composé d'un ensemble de tables de dimension $\mathbb{D} = \{D_1, D_2..., D_d\}$ et une table de faits F,
2. un ensemble de requêtes OLAP les plus fréquentes $\mathbb{Q} = \{Q_1, Q_2..., Q_m\}$, où chaque requête $Q_i (1 \leq i \leq m)$ possède une fréquence d'accès $freq_i$ et
3. un seuil (W) fixé par l'administrateur représentant le nombre maximum de fragments qu'il peut maintenir.

$\mathbb{PSSFH}$ consiste à fragmenter des tables de dimension $D' \subseteq D$ et utiliser leurs schémas de fragmentation pour partitionner la table de faits F en un ensemble de N fragments horizontaux $\{F_1, F_2, \cdots, F_N\}$ tels que :

1. le coût total d'exécution des requêtes sur le schéma en étoile fragmenté soit réduit,
2. la contrainte de maintenance soit satisfaite ($N \leq W$)

5.1.3.4 Etude de Complexité

Nous avons montré que ce problème est NP-complet au sens fort. Nous l'avons réduit comme un problème de trois partitions connu comme un problème NP-complet [21, 22]. Pour plus de détails, voir [56].

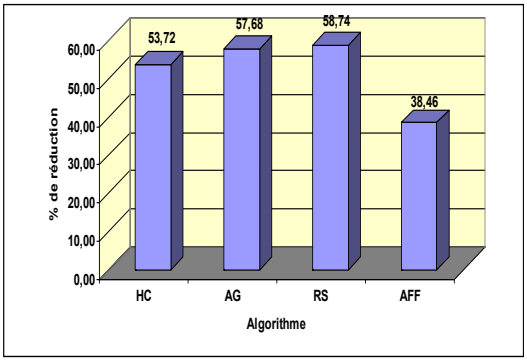


FIG. 5.7 – Taux de réduction par algorithme

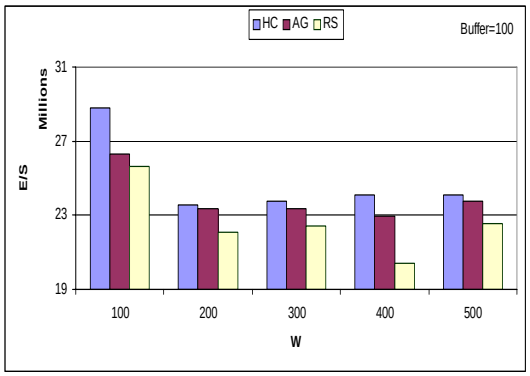


FIG. 5.8 – Effet de W

5.1.3.5 Algorithmes de sélection de schéma de fragmentation

Vue la complexité de notre problème de fragmentation, nous avons proposé trois algorithmes : un algorithme de hill climbing (HC), un recuit simulé (RS) et un algorithme génétique (AG) [56, 17, 21]. Les trois utilisent le codage que nous avons proposé. Ils sont contrôlés par un modèle de coût.

5.1.3.6 Expérimentations

Afin d'évaluer les algorithmes proposés, nous avons mené deux types d'expérimentation: (1) une en utilisant un modèle de coût théorique estimant le nombre de pages nécessaires pour exécuter une charge de requêtes [56] et (2) une autre en utilisant le SGBD Oracle10G pour valider les résultats obtenus. Nous avons utilisé l'entrepôt de données issu du benchmark APB1 [75]. Le schéma en étoile de ce benchmark est constitué d'une table de faits Actvars et de quatre tables de dimension, *Prodlevel, Custlevel, Timelevel* et *Chanlevel*. Sur cet entrepôt, nous avons considéré 60 requêtes de recherche. Chaque requête a une fréquence d'accès et un ensemble de prédicats de sélection. L'ensemble des requêtes utilise 45 prédicats de sélection définis sur 12 attributs de sélection. Le facteur de sélectivité de chaque prédicat de sélection est calculé directement sur l'entrepôt réel avec des requêtes SQL.

5.1.3.6.1 Evaluation Théorique Dans la première expérimentation, nous avons exécuté les quatre algorithmes : Affinité (AFF), AG, HC et RS pour un seuil 100. La figure 5.7 montre le taux de réduction du coût d'exécution des requêtes apporté par le schéma de fragmentation sélectionné par chaque algorithme par rapport au schéma non fragmenté. L'algorithme d'affinité que nous avons proposé dans le contexte de bases de données relationnelles a sélectionné un schéma de fragmentation générant 16 fragments. Ce schéma réduit le coût de 38% par rapport au cas non fragmenté. Les deux algorithmes HC et RS ont permis d'améliorer cette solution pour atteindre 53% et 58% respectivement. L'AG a sélectionné quant à lui, un schéma de fragmentation réduisant le coût de 57%. Il est clair que les trois algorithmes RS, HC et AG qui sont basés sur un modèle de coût et guidés par le nombre de fragments donnent de meilleurs résultats que celui de l'algorithme d'affinité qui n'est basé que sur la fréquence d'accès comme critère de fragmentation. *Ce résultat confirme nos critiques aux algorithmes dirigés par affinités.*

La figure 5.8 montre les performances de chaque algorithme par rapport au seuil *W*. Nous avons

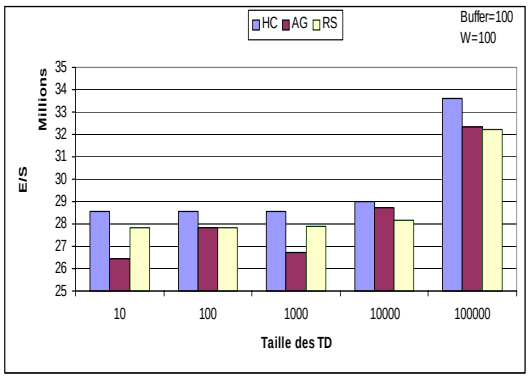


FIG. 5.9 – Effet de la taille des TD

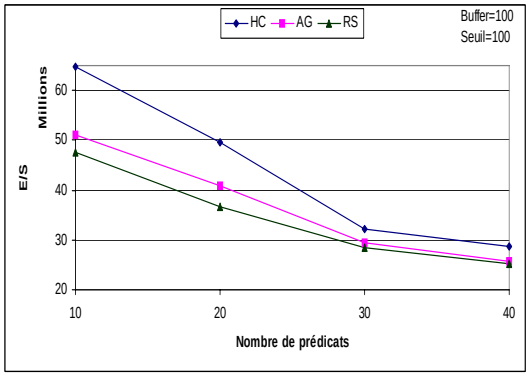


FIG. 5.10 – Effet du nombre de prédicats

fait varier W de 100 à 500 en utilisant 45 prédicats, et pour chaque valeur de W nous exécutons chaque algorithme. Le recuit simulé et le génétique donnent de meilleurs résultats. L'algorithme HC est moins performant du fait qu'il est confronté au problème de blocage dans des optimums locaux. L'augmentation du seuil améliore généralement les performances des requêtes car en relâchant W, plus d'attributs sont utilisés pour fragmenter l'entrepôt. Lorsque W est grand, les partitions sont plus fines, donc moins volumineuses. Cela implique moins de données non pertinentes chargées pour exécuter les requêtes utilisant les attributs de fragmentation. Lorsque W est petit, les partitions générées sont volumineuses et chaque requête charge plus de données non pertinentes contenues dans ces partitions. Ainsi, lorsque W est petit peu d'attributs sont utilisés pour fragmenter l'entrepôt, ce qui implique que les requêtes utilisant les autres attributs (non utilisés pour fragmenter l'entrepôt) accèdent à toutes les partitions, ce qui explique la diminution de la performance.

La figure 5.9 montre l'effet de la taille des tables de dimension (en termes de nombre de tuples) sur la qualité des schémas de fragmentation des algorithmes. Nous avons considéré que toutes les tables de dimension ont le même nombre de tuples et nous avons fait varier ce nombre entre 10 et 100 000. Les résultats montrent que pour les petites tailles des tables (de 10 à 1000), les résultats sont presque similaires, du fait que pour ces cas les fragments des tables de dimension sont suffisamment petits pour tenir dans un nombre de pages similaires. Lorsque la taille des tables de dimension devient importante, le temps d'exécution de requêtes augmente, car les fragments de ces dernières occupent plus d'espace et provoquent plus de données enregistrées sur le disque lors de leur jointure avec les fragments de faits.

Pour voir l'effet du nombre de prédicats utilisés dans les 60 requêtes sur la performance globale, nous faisons varier ce nombre entre 10 et 40. Nous avons créé quatre instances de requêtes utilisant chacune un nombre de prédicats différents (10, 20, 30 et 40 prédicats). Pour chaque instance, nous avons exécuté nos algorithmes pour W=100. Les résultats obtenus montrent que le nombre de prédicats utilisés par les requêtes a un effet considérable sur la performance des requêtes. Lorsque ce nombre est petit, la plupart des requêtes ne bénéficient pas de la fragmentation. Ceci s'explique par le fait qu'elles accèdent à un nombre important de sous-schémas, voire la totalité des sous-schémas si elles ne possèdent pas des prédicats définis sur des attributs de fragmentation. Par conséquent, plusieurs opérations d'union sont nécessaires pour avoir le résultat final. Par contre, si le nombre de prédicats est important, le nombre de sous-schémas valides pour chaque requête est réduit (surtout pour celles n'utilisant que des attributs de fragmentation), ce qui implique le chargement de moins de données (les sous-schémas valides seulement).

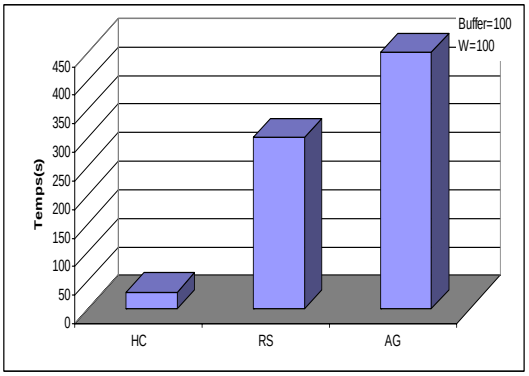


FIG. 5.11 – Temps d'exécution de chaque algorithme

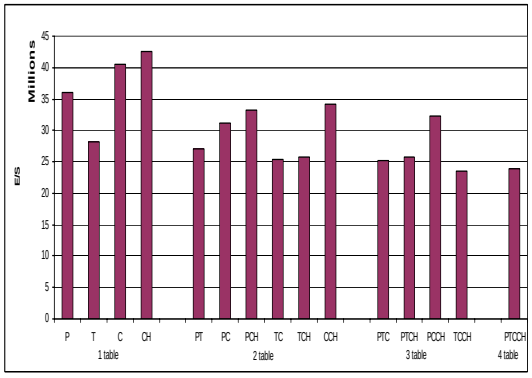


FIG. 5.12 – Choix et nombre de tables de dimension

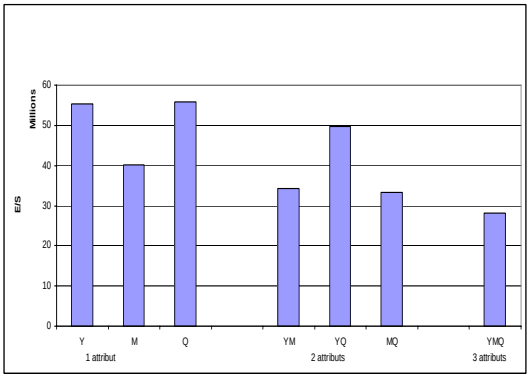


FIG. 5.13 – Choix des attributs de la table *Timelevel*

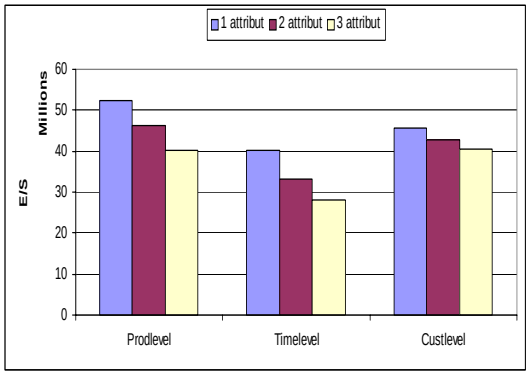


FIG. 5.14 – Effet du nombre d'attributs de fragmentation de chaque table

La figure 5.11 donne le temps moyen d'exécution de chaque algorithme pour $W = 100$. Le génétique et le recuit simulé nécessitent plus de temps d'exécution vu qu'ils utilisent plusieurs opérations. L'algorithme génétique prend plus de temps car il manipule plusieurs solutions en même temps. Le hill climbing est l'algorithme le plus rapide puisqu'il est basé sur deux opérations simples, à savoir *Merge* et *Split*.

Pour montrer l'effet du choix des tables de dimension à fragmenter, nous avons mené des expérimentations en considérant les cas suivants : *Cas 1:* une table parmi les quatre tables de dimension est fragmentée, *Cas 2:* deux parmi quatre, *Cas 3:* trois parmi quatre et *Cas 4:* quatre parmi quatre. Pour chaque cas, toutes les combinaisons de tables sont considérées. Ces expérimentations ont été effectuées en utilisant le RS, puisqu'il donne les meilleurs résultats. Les tables utilisées sont : *Prodlevel(P)*, *Timelevel(T)*, *Custlevel(C)* et *Chanlevel(CH)*. La figure 5.12 montre les résultats obtenus. Ces derniers montrent que le choix des tables de dimension à fragmenter est important. Par exemple, la table *Timelevel* est la plus adaptée pour être partitionnée, cela est justifié par le fait qu'elle est celle ayant le plus grand nombre de prédicats dans la charge de requêtes. Elle est aussi la table la plus utilisée par la charge de requêtes et par conséquent ces requêtes accéderont à moins de sous-schémas en étoile, donc moins de données chargées. Les résultats montrent aussi que choisir plus de tables de dimension pour la fragmentation donne

plus de performance, à condition de choisir les bonnes tables.

Pour montrer l'effet du choix des attributs, nous avons pris l'exemple de la table *Timelevel*. Nous avons considéré toutes les combinaisons possibles pour le choix des attributs (voir figure 5.13). Ces attributs sont (*Monthlevel(M)*, *Yearlevel(Y)* et *Quarterlevel(Q)*. L'attribut *Monthlevel* donne les meilleurs résultats. Ceci s'explique par le fait que *Monthlevel* est l'attribut ayant le plus grand nombre de prédicats dans la charge de requêtes et est l'attribut le plus utilisé. Choisir cet attribut comme un attribut de fragmentation permet aux requêtes ayant des prédicats sur cet attribut d'accéder à un nombre réduit de sous-schémas, donc moins de données sont chargées.

La même expérience a été menée pour les deux autres tables de dimension. Nous avons pris pour chaque table le meilleur temps d'exécution des requêtes pour le cas de 1, 2 ou 3 attributs choisis pour fragmenter l'entrepôt (voir figure 5.14). Les résultats montrent que lorsqu'on choisit plus d'attributs pour fragmenter l'entrepôt nous obtenons plus de performance, car plus de requêtes sont satisfaites (celles utilisant ces attributs) en accédant à moins de sous-schémas en étoile à condition de bien choisir ces attributs.

5.1.3.7 Bilan sur les résultats théoriques

Les résultats montrent que le choix de l'*algorithme de sélection*, des *tables de dimension* ainsi que les *attributs utilisés* pour fragmenter l'entrepôt est très important pour garantir une bonne performance des requêtes. Ils montrent l'efficacité (en termes de performance) du recuit simulé. Le hill climbing représente l'algorithme le plus rapide en temps d'exécution. Ils montrent également que lorsque plus de tables de dimension et d'attributs sont utilisés dans le processus de fragmentation, la performance est meilleure. Cela est dû à la nature des requêtes décisionnelles qui sont utilisées sur la totalité de l'entrepôt (la majorité des dimensions et des attributs).

Dans la section suivante, nous effectuons une réelle validation en utilisant les données issues de banc d'essai APB1 sous Oracle10G.

5.1.3.7.1 Validation sous Oracle 10g Afin de mesurer la qualité des solutions proposées par nos algorithmes, nous avons effectué une validation réelle sur le SGBD Oracle 10G. Les schémas de fragmentation obtenus par chaque algorithme sont implémentés sous Oracle10G et l'ensemble de requêtes est exécuté sur chaque entrepôt de données partitionné (*un entrepôt par algorithme*). Plusieurs difficultés sont apparues pour implémenter la fragmentation primaire. Elles sont liées au fait qu'Oracle10g ne supporte pas une fragmentation primaire sur trois niveaux ou plus en utilisant des combinaisons des modes de base (seule une fragmentation à un ou deux niveaux est supportée). Pour remédier à ces difficultés, nous avons développé une technique permettant d'implémenter la fragmentation sur plusieurs attributs [56]. La fragmentation dérivée est réalisée avec le mode de partitionnement par référence. Or, ce mode permet de fragmenter la table des faits en utilisant le schéma d'une seule table de dimension. Nous avons donc rencontré un autre problème pour implémenter la fragmentation référentielle pour plus de deux tables de dimension. Une solution a été proposée à ce problème [56]. Une architecture d'implémentation pour cette validation a été proposée. Elle est décrite dans la figure 5.15. Elle est composée de trois modules principaux: (1) le *module de sélection d'un schéma de fragmentation*, (2) le *module de fragmentation de l'entrepôt* et (3) le *module de réécriture des requêtes*.

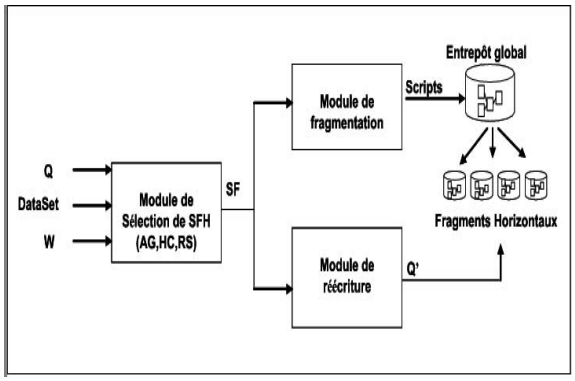


FIG. 5.15 – Architecture de notre implémentation

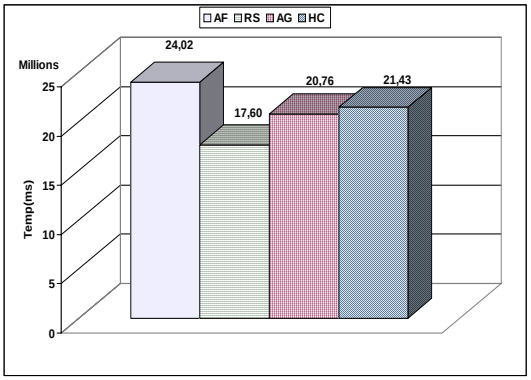


FIG. 5.16 – Résultats sous Oracle

Nous avons implémenté les schémas de fragmentation obtenus à partir de chaque algorithme sous Oracle 10g. Nous avons choisi le scénario suivant: le seuil *W* est fixé à 100. Pour ce dernier l'AG et le RS ont généré un schéma de fragmentation composé de 80 sous-schémas en étoile. Le schéma de fragmentation obtenu par l'AG utilise 5 attributs parmi 12 et 3 tables de dimension (*Timelevel, Custlevel* et *Chanlevel*). Le RS utilise 5 attributs mais toutes les tables de dimension ont été utilisées pour fragmenter l'entrepôt. Le HC a généré 96 sous-schémas en étoile en utilisant 4 attributs et 3 tables de dimension (*Prodlevel*, *Timelevel* et *Chanlevel*).

Nous avons implémenté chaque schéma en utilisant notre module de fragmentation. Les requêtes d'origine ont été réécrites par notre module de réécriture sur les schémas de fragmentation obtenus à partir de chaque algorithme. Nous avons exécuté les requêtes d'origine sur l'entrepôt non fragmenté (AF) et les requêtes réécrites sur l'entrepôt fragmenté correspondant. Le tampon est vidé après l'exécution de chaque requête pour éviter l'influence de l'exécution d'une requête sur l'autre. La figure 5.16 montre les résultats obtenus. Elle illustre deux points importants: (i) la fragmentation horizontale dans les entrepôts de données est cruciale pour la performance de requêtes et (ii) le choix de l'algorithme de fragmentation a un impact considérable sur cette performance. L'algorithme de recuit simulé donne de meilleurs résultats (comme dans l'évaluation théorique) que l'algorithme génétique ou le hill climbing, ce qui *confirme les résultats théoriques et la qualité de notre modèle de coût mathématique*.

5.1.3.8 Conclusion

La fragmentation horizontale a connu une évolution importante ces dernières années. Elle est supportée par la plupart des SGBDs commerciaux et non commerciaux. Pour effectuer une bonne fragmentation, l'administrateur doit effectuer plusieurs choix : les tables à fragmenter, les attributs à utiliser pour fragmenter ces tables ainsi que le partitionnement des domaines des attributs utilisés. Il doit aussi maîtriser le nombre de fragments générés après le processus de fragmentation.

Dans cette section, nous avons établi un état de l'art sur les principaux travaux de fragmentation horizontale proposés dans le cadre des bases de données traditionnelles. Cet état de l'art nous a permis d'identifier les limites des algorithmes proposés et proposer une démarche de fragmentation pour les entrepôts de données relationnels, tout en prenant en compte les caractéristiques de requêtes décisionnelles en étoile. Notre démarche consiste à décomposer d'abord les tables de dimension ensuite à propager

leurs schémas de fragmentation pour partitionner la table des faits. Nous avons formalisé le problème de sélection d'un schéma de fragmentation horizontale et nous avons prouvé qu'il est NP-Complet. Pour répondre à ce problème, nous avons proposé trois algorithmes : un algorithme génétique, un recuit simulé et un hill climbing. Les trois algorithmes sont guidés par un modèle de coût. Nous avons effectué une évaluation de performances des algorithmes proposés et une validation sous Oracle. Les résultats obtenus montrent l'intérêt de la fragmentation horizontale dans la conception physique des entrepôts de données.

5.2 Un autre exemple de la sélection isolée: les index multi tables

Publications [42, 33, 34, 25, 26, 35].

Dans la section précédente, nous avons étudié la fragmentation horizontale primaire et dérivée qui sont deux structures d'optimisation non redondantes réduisant le coût d'exécution des opérations de sélections et de jointures, respectivement. Dans cette partie, nous présentons une structure d'optimisation redondante optimisant également les jointures et/ou les sélections : les index multi table. Deux types d'index multi table sont étudiés : *l'index de graphe de jointure* et *l'index de jointure binaire*. Avant d'introduire ces derniers, quelques définitions s'imposent.

Définition 3

Un graphe d'une base de données est un graphe connecté dont les noeuds représentent les tables. Une arête entre deux noeuds T_i et T_j représente une possibilité de jointure entre eux.

Définition 4

Un graphe de jointure est un graphe d'une base de données tel qu'une seule condition de jointure existe entre deux différentes tables. Le nombre de noeuds d'un graphe de jointure est appelé la cardinalité de ce graphe.

Dans les entrepôts de données modélisés par un schéma en étoile, le graphe de jointure et le graphe de l'entrepôt de données sont équivalents (un graphe de jointure est un graphe d'entrepôt de données et vice versa).

Définition 5

Soit un graphe de jointure de cardinalité m ($m \geq 2$). Un index de graphe de jointure (IGJ) est un sous ensemble de h ($h \leq m$) noeuds de graphe de jointure.

Nous avons proposé les IGJs pour optimiser les opérations de jointure définies sur des vues matérialisées [10]. Une fois les vues matérialisées sélectionnées, toutes les requêtes décisionnelles définies sur l'entrepôt doivent être réécrites en fonction de ces vues et des tables de base. Cette réécriture permet de réduire le coût de requêtes [198, 44].

Plus formellement, soit Q une requête définie sur un schéma d'un entrepôt. Supposons que, sur ce dernier, un ensemble de vues matérialisées $V = \{V_1, V_2, ..., V_k\}$ ait été sélectionné. L'optimiseur doit prendre en compte ces vues matérialisées pour trouver le meilleur plan d'exécution de la requête Q. Trois scénarios se présentent à l'optimiseur pour réaliser sa tâche:

- *Scénario 1:* Q est exécutée en utilisant seulement des vues matérialisées.
- *Scénario 2:* Q est exécutée en utilisant des vues matérialisées et des tables de base,
- *Scénario 3:* Q est exécutée en utilisant des tables de base seulement.

Certes, la présence des vues matérialisées a un effet positif sur la réduction du coût d'exécution des requêtes, mais sans technique d'indexation, ce coût reste toujours important [32, 135, 192]. Etant donné que les vues peuvent être stockées sous forme de tables, les techniques d'indexation des tables peuvent être utilisées sur vues. La définition de ces index dépend des précédents scénarii.

- Les index dans le premier scénario sont définis seulement sur les vues. Deux types d'index sont possibles : (1) *les index mono vue* et (2) *les index de multi vues*. Un index mono vue peut être défini sur une ou plusieurs colonnes d'une vue. Cette indexation peut être réalisée en utilisant les

mêmes techniques que celles dédiées aux tables (e.g., arbre B^+, index de projection, index binaire, etc). Un index multi-vue peut être défini sur deux ou plusieurs vues.

– Des index simples peuvent être définis sur les vues ou les tables de base dans le deuxième scénario. Les index de jointure peuvent être définis entre : (1) des vues seules, pour accélérer par exemple des jointures de plusieurs vues, (2) des tables seules, ou (3) entre des vues et des tables. Les observations sur les index de jointure décrites dans le paragraphe précédent sont valides également dans ce scénario.
– Les index dans le troisième scénario ressemblent au premier, mais au lieu d'indexer les vues, on indexe les tables. Des index simples et de jointure binaire peuvent être utilisés si l'entrepôt est modélisé par un schéma en étoile.

Notons que les index de jointure binaires sont proposés pour satisfaire les requêtes de jointure en étoile [123][13]. A partir de ce constat et pour optimiser les requêtes des scénarii 1 et 2 qui sont définies sur des vues matérialisées, nous avons proposé dans le cadre de nos travaux de thèse les index de graphe de jointure avec des algorithmes pour les sélectionner [29, 10].

Rappelons qu'un index de jointure en étoile est un cas particulier d'index de graphe de jointure. Il pré-calcule la jointure entre la table des faits et une ou plusieurs tables de dimension. Cette jointure est matérialisée à travers un ensemble de vecteurs de bits construits sur la table des faits en se basant sur un ou plusieurs attributs de dimension de faible cardinalité. Un vecteur de bits représentant les n-uplets de la table de faits est créé pour chaque valeur distincte de l'attribut de la table de dimension sur lequel l'index est construit. Le $i^{ème}$ bit du bitmap est égal à 1, si le n-uplet correspondant à la valeur de l'attribut indexé peut être joint avec le n-uplet de rang i de la table de faits. Dans le cas contraire, le $i^{ème}$ bit est à zéro. Les IJB sont efficaces pour les requêtes de type COUNT, AND, OR, NOT, d'où leur implémentation dans les SGBD commerciaux. Les IJB définis sur des attributs de *faible cardinalité* sont souvent bénéfiques par rapport aux index B-arbre pour deux considérations : la taille réduite et la possibilité de combinaison. La taille des index binaires est proportionnelle à la cardinalité des attributs indexés. Ils ne sont pas recommandés pour les attributs de forte cardinalité. Pour réduire la taille de ces index, plusieurs travaux ont été proposés. Ils s'orientent vers deux directions : l'encodage [216] et la compression [128]. Grâce à la compression, les index de jointure binaire sont les moins gourmands en stockage des structures redondantes.

Un IJB peut être défini sur un ou plusieurs attributs de dimension utilisés dans des prédicats de sélection des requêtes de jointure en étoile. Par conséquent, plusieurs index sont candidats. Choisir un ensemble d'IJBs optimisant les requêtes est une *tâche difficile* pour un administrateur. Peu de travaux se sont intéressés au problème de sélection des IJB [4, 36, 37], contrairement à la sélection des index mono-table[14].

Nous formalisons d'abord le problème de sélection des IJB, ensuite nous proposons des algorithmes de sélection.

[13] Une requête de jointure en étoile est une requête définie sur un schéma en étoile
[14] Un index mono-table est un index défini sur une seule table

5.2.1 Formalisation

Le problème de sélection des IJBs sous la contrainte de l'espace de stockage peut être formalisé comme suit:
Étant donné:

- un entrepôt de données modélisé par un schéma en étoile ayant un ensemble de tables de dimension $D = \{D_1, D_2, ..., D_d\}$ et une table des faits F,
- une charge de requêtes $Q = \{Q_1, Q_2, ..., Q_m\}$, où chaque requête Q_j a une fréquence d'accès f_j, et
- une capacité de stockage S;

le problème de sélection des index de jointure consiste à trouver une configuration d'index CI minimisant le coût d'exécution de Q en respectant la contrainte de stockage ($Taille(CI) \leq S$).

5.2.1.1 Etude de complexité du problème de sélection des IJBs

La sélection d'une configuration d'IJB est généralement une tâche difficile comparée à d'autres types d'index. Cela est dû à plusieurs considérations :

- Un IJB peut être défini sur tout attribut non clé de *faible cardinalité* des tables de dimension. Un attribut est dit indexable s'il est utilisé par un prédicat de sélection. Dans le contexte d'entrepôts de données, le nombre d'attributs indexables peut être important, vu le nombre de tables de dimension et le nombre d'attributs non clés par table de dimension [196].
- Un IJB peut être défini sur seul un attribut (dans ce cas il est appelé un IJB *mono-attribut*) ou sur plusieurs attributs issus d'une seule table ou plusieurs tables de dimension (il est appelé un IJB *multi-attributs*), ce qui augmente le nombre d'IJB possibles.
- Un attribut indexable peut être présent dans plusieurs IJB.
- La taille d'un IJB est proportionnelle à sa cardinalité. Un attribut de forte cardinalité rend l'index volumineux, donc difficile à stocker et à maintenir.

Soit $A = \{A_1, A_2, ..., A_K\}$ un ensemble d'attributs indexables candidats pour la sélection d'une configuration d'IJB. Chaque IJB dans cette configuration est constitué d'un sous-ensemble d'attributs de A, par conséquent cette configuration constitue une partition de A en un ensemble de groupes. Chaque groupe d'attributs représente un IJB potentiel. Nous pouvons considérer deux scenarii :

1. *Sélection d'un seul index de jointure* : si on souhaite sélectionner un seul IJB, le nombre de possibilités est donné par:
$$N = \binom{K}{1} + \binom{K}{2} + ... + \binom{K}{K} = 2^K - 1 \tag{5.4}$$
2. *Sélection de plus d'un IJB* : Le nombre de possibilités est donné par:
$$N = \binom{2^K-1}{1} + \binom{2^K-1}{2} + ... + \binom{2^K-1}{2^K-1} = 2^{2^K-1}$$

5.2.1.2 Résolution du problème de sélection d'index

Pour résoudre le problème de sélection des index de jointure binaire, nous avons d'abord effectué un état des lieux sur la sélection traditionnelle des index mono table[15] afin de s'en inspirer. Cette sélection est également difficile [76, 67].

[15] Un index mono table est un index défini sur une seule table.

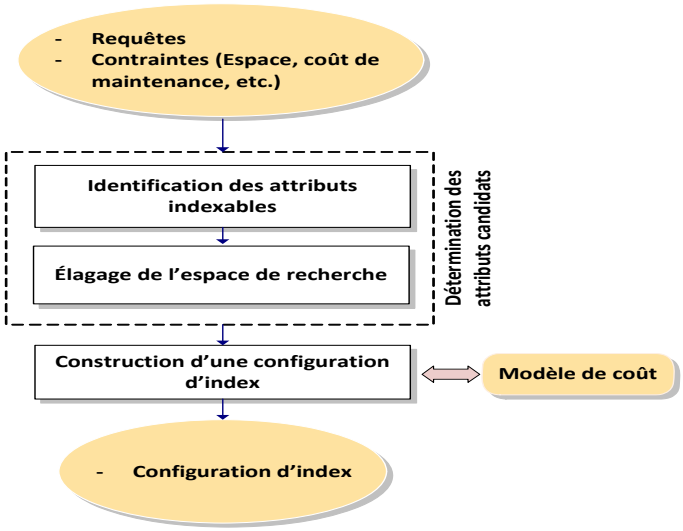


FIG. 5.17 – Approche générique de sélection d'index

5.2.1.2.1 Sélection des index mono table Un nombre important d'algorithmes a été proposé pour sélectionner des index mono attributs. Ces derniers possèdent généralement deux étapes principales (voir figure 5.17): (1) *l'élagage des attributs non pertinents* et (2) *la construction de la configuration finale d'index.*

1. La première étape consiste à réduire l'espace de recherche associé aux index. Plusieurs techniques d'élagage ont été proposées que nous avons classé en deux catégories: (1) *manuelle* et (2) *automatique*. Dans la première catégorie, l'administrateur élague manuellement des attributs qu'il considère non nécessaires pour le processus d'indexation. Ce élagage est basé généralement sur son expertise [91]. L'inconvénient majeur de cette catégorie est l'absence d'une métrique quantifiant la qualité des index sélectionnés. La deuxième catégorie, quant à elle, consiste d'abord à énumérer les attributs candidats à l'indexation, élague ensuite d'une manière automatique des attributs non pertinents pour l'indexation. Un *modèle de coût* est souvent utilisé pour arbitrer cet élagage [68, 208, 67, 135]. Le travail du groupe de Microsoft est un exemple de cette sélection, où le modèle de coût de l'optimiseur du SQL Server est utilisé [68]. Ce modèle de coût quantifie l'intérêt de chaque attribut pour le processus d'indexation, en conséquence, leur élagage est facile. Ce travail impose un nombre d'index candidats générés. DB2 Advisor est un autre exemple de cette catégorie [208], où l'analyseur de requêtes est utilisé pour récupérer les attributs de sélection utilisés dans la charge de requêtes. Les index candidats sont alors générés en utilisant une simple combinaison des attributs de sélection. Ces deux élagages sont propriétaires des SGBD commerciaux.
2. Une fois l'élagage établi, l'étape de construction d'index vise à construire un ensemble d'index à partir des attributs indexables candidats et l'ensemble des requêtes définies sur la base de données. Deux catégories de construction de configurations finales existent: les *approches ascendantes* et les *approches descendantes*. Les approches ascendantes commencent par une configuration vide et l'enrichissent jusqu'au but atteint [91, 73, 68]. Les approches descendantes commencent par une configuration complète contenant tous les index candidats. Dans chaque itération, un ou plusieurs

index sont éliminés afin de réduire les coûts de stockage et d'exécution. Le processus s'arrête lorsqu'aucune amélioration du coût n'est possible par n'importe quelle élimination d'index.

Dans la section suivante, nous présentons nos travaux sur le processus de sélection des index de jointure binaires qui concerne à la fois l'élagage et la sélection de la configuration finale.

5.2.2 Notre approche d'élagage pour la sélection des index de jointure binaire

Peu de travaux d'élagage de l'espace de recherche du problème de sélection des index multi table existe. Le premier travail basé sur les techniques de fouille de données a été proposé par Aouiche et al. [4]. Les auteurs ont proposé une approche basée sur une recherche des motifs fréquents [173], où l'algorithme *Close* basée sur le *treillis de Galois* est utilisé. L'inconvénient principal de cette approche est l'utilisation seule de la fréquence d'accès pour élaguer les attributs indexables. Ce choix peut être discutable, car nous avons fait le lien entre ce problème et le problème de sélection d'un schéma de fragmentation verticale [158]. Les travaux sur la fragmentation verticale ont montré la limite des algorithmes de fragmentation basés sur la fréquence d'accès [93]. En effet, les IJB sont créés pour optimiser des jointures entre la table des faits et les tables de dimension. La jointure est coûteuse lorsqu'elle concerne des tables volumineuses.

Compte tenu que les techniques d'élagage automatiques existantes reposent sur un modèle de coût [68], nous avons proposé une approche d'élagage hybride combinant les techniques de fouille de données et un modèle de coût. Avant de la présenter, nous définisons quelques termes.

Définition 6

Un motif (itemset) fréquent : soient $I = \{i, ..., m\}$ un ensemble de m items et $B = \{t_1, ..., t_n\}$ une base de données (contexte) de n transactions. Chaque transaction est composée d'un sous ensemble d'items. Un sous ensemble I' de taille k est appelé un k-itemset.

Définition 7

Le support d'un motif I', noté $sup(I')$, est la proportion de transactions de B qui contiennent I'.

Définition 8

Motif fréquent. Soit un seuil $minsup \in [0, 1]$, appelé le support minimum. Un motif est dit fréquent si : $sup(P) \geq minsup$.

Définition 9

Motif fréquent fermé. Un motif fermé est un ensemble maximal d'items communs à un ensemble d'objets. Un motif $i \subseteq I$ tel que le $support(i) \geq minsup$ est appelé motif fréquent fermé.

Dans le contexte de sélection des index de jointure binaires, les transactions sont des requêtes fréquentes et les items sont les attributs utilisés dans les requêtes.

Pour montrer les limites de l'approche existante [4], considérons l'exemple suivant. Soit un ensemble de cinq requêtes définies sur un schéma en étoile composé de deux tables de dimension *Channels* et *Customers* et une table des faits *Sales*. La taille des tables (en termes d'instances) est:

$||Sales|| = 46521, ||channels|| = 5$ and $||customers|| = 30000$. Sur ce schéma, cinq requêtes sont définies (Table 5.2).

(1) select S.channel_id, sum(S.quantity_sold) from S, C where S.channel_id=C.channel_id and *C.channel_desc='Internet'* group by S.channel_id
(2) select S.channel_id, sum(S.quantity_sold), sum(S.amount_sold) from S, C where S.channel_id=C.channel_id and *C.channel_desc ='Catalog'* group by S.channel_id
(3) select S.channel_id, sum(S.quantity_sold),sum(S.amount_sold) from S, C where S.channel_id=C.channel_id and *C.channel_desc ='Partners'* group by S.channel_id
(4) select S.cust_id, avg(quantity_sold) from S, C where S.cust_id=C.cust_id and C.cust_gender='M' group by S.cust_id
(5) select S.cust_id, avg(quantity_sold) from S, C where S.cust_id=C.cust_id and *C.cust_gender='F'* group by S.cust_id

TAB. 5.2 – Description de requêtes

Supposons que $minsup = 3$ (en valeur absolue). Un algorithme de sélection des index candidats en utilisant une approche basée sur l'extraction des itemsets fermés fréquents, comme *Close* utilisée dans [4], retourne un index de jointure binaire construit sur la table *Channels* et *Sales* sur l'attribut *Channel_desc* (figurant 3 fois dans les prédicats de sélection des requêtes). L'index de jointure entre la table de dimension *Customers* et la table des faits *Sales* (sur l'attribut *Gender*) est élagué car il n'est pas fréquent (il ne figure que deux fois). Mais ce dernier pourrait réduire le coût d'exécution de requête car il est défini sur une table de dimension plus importante que la table *Channels* (30 000 vs. 5). Cet exemple montre l'insuffisance des travaux utilisant les approches d'extraction des itemsets fréquents ayant comme métrique d'élagage la fréquence d'apparition des attributs dans les requêtes. Nous affirmons que pour une bonne sélection d'index de jointure, d'autres paramètres comme la taille des tables doivent être pris en considération.

Cet exemple a démontré les insuffisances des approches d'extraction des itemsets fréquents dans le contexte de sélection des index de jointure binaires. En conséquence, nous les avons adapté afin de prendre en compte les paramètres comme la taille des tables [37, 36].

Nous avons comparé notre approche d'élagage avec celle proposée dans [4]. Les résultats ont permis de tirer des résultats préliminaires assez encourageants

Dans la section suivante, nous présentons quelques algorithmes pour la sélection finale d'une configuration d'index.

5.2.3 Nos algorithmes de construction de configuration

Une fois le processus d'élagage établi, nous passons à la construction de configuration d'IJBs. Nous proposons deux types d'algorithmes de sélection, le premier sélectionne une configuration d'index mono-attributs et le deuxième un ensemble d'index multi-attributs.

5.2.3.1 Sélection d'une configuration d'index mono-attributs

Dans la sélection mono-attributs, les IJB sont définis entre la table des faits et une seule table de dimension. Dans [20], nous avons proposé un algorithme de sélection qui se déroule en trois étapes :(1)

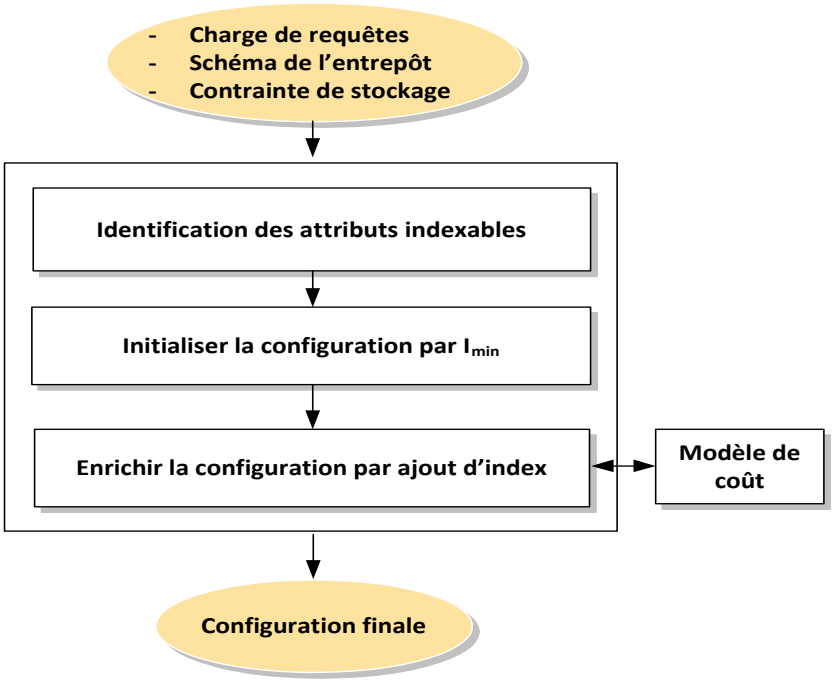


FIG. 5.18 – Architecture de notre approche de sélection d'IJB mono-attribut

l'identification des attributs indexables, (2) l'initialisation de la configuration et (3) l'enrichissement de la configuration initiale. Dans la première étape, l'ensemble des requêtes est analysé afin d'extraire les attributs indexables. Ces attributs sont les attributs sur lesquels un prédicat de sélection est défini dans la charge de requêtes. Les attributs indexables candidats sont choisis parmi les attributs indexables de faible et de moyenne cardinalité. Les étapes 2 et 3 sont réalisées au sein d'un algorithme glouton basé sur le modèle de coût. Il commence par une configuration composée d'un index mono-attribut défini sur l'attribut ayant une cardinalité minimum noté BJI_{min}. La configuration initiale est améliorée itérativement par l'ajout d'index définis sur d'autres attributs non encore indexés. L'algorithme s'arrête lorsque deux conditions sont satisfaites : aucune amélioration n'est possible et l'espace de stockage est consommé.

5.2.4 Sélection d'une configuration d'index multi-attributs

Un IJB multi-attributs peut être défini sur plusieurs attributs $\{A_1, A_2, \cdots, A_K\}$ où chaque attribut A_j peut appartenir à n'importe quelle table de dimension. L'algorithme descendant que nous avons proposé commence par une configuration initiale dans laquelle un IJB est défini sur chaque requête. Cette configuration ne prend pas en considération le coût de stockage, en conséquence, son espace de stockage peut dépasser la contrainte initiale. Pour réduire cette complexité, nous proposons l'algorithme en effectuant des améliorations itératives sur la configuration initiale afin de réduire le coût d'exécution des requêtes et de respecter la contrainte de stockage.

Notons que si la taille de la configuration initiale est inférieure à la capacité de stockage, alors elle est automatiquement choisie comme configuration finale. Dans le cas contraire, des opérations de réduction

de la taille de la charge sont effectuées jusqu'à ce que la contrainte de stockage soit satisfaite. La réduction de la taille d'un index peut être faite en éliminant certains attributs participant à sa définition. Supposons que l attributs sont utilisés pour définir les index de la configuration initiale. L'élimination d'attributs de cette configuration pose le problème du choix d'attributs à éliminer. Plusieurs stratégies peuvent être suivies pour effectuer ce choix. Nous avons considéré les quatre stratégies suivantes [56]:

1. *Elimination des attributs de forte cardinalité (AFC):* La principale cause de l'explosion de la taille des IJB est la cardinalité des attributs indexés. Dans cette stratégie, l'attribut de forte cardinalité est éliminé de tous les index constituant la configuration initiale. Pour cela, les attributs sont triés par ordre décroissant sur leur cardinalité, ils seront éliminés dans l'ordre jusqu'à ce que la taille de la configuration devienne inférieure à la capacité de stockage.
2. *Attributs appartenant aux tables moins volumineuses (TMV):* le principe de cette stratégie est de donner une priorité aux index pré-calculant des jointures entre la table des faits et les tables de dimension volumineuses. Notons que le coût d'une jointure est proportionnel aux tailles des tables jointes. Si plusieurs attributs d'une même table sont candidats pour la suppression, alors ceux ayant une forte cardinalité sont éliminés les premiers.
3. *Attributs les moins utilisés (AMU):* cette stratégie suppose que les attributs les plus utilisés doivent être indexés pour satisfaire plus de requêtes. Par conséquent, les attributs les moins utilisés sont éliminés en premier.
4. *Attributs apportant moins de réduction de coût (MC) :* l'inconvénient des stratégies que nous venons de présenter est qu'elles éliminent des attributs sans avoir une évaluation des configurations intermédiaires issues de chaque élimination. Cela nécessite de définir une stratégie utilisant un modèle de coût pour choisir les meilleures configurations. Pour avoir ces configurations, les éliminations d'attributs qui engendrent des configurations de mauvaise qualité seront retenues. Pour k attributs candidats, k éliminations sont effectuées et celles engendrant un coût maximum seront retenues. Pour ce faire, nous avons développé un modèle de coût estimant le coût d'une requête en présence d'un IJB [56].

5.2.4.1 Expérimentations

Nous avons effectué une évaluation de performance de notre approche d'indexation sur l'entrepôt de données issu du benchmark APB-1 [75]. Nous avons considéré 12 attributs candidats à l'indexation (*ClassLevel, GroupLevel, FamilyLevel, LineLevel, DivisionLevel, YearLevel, MonthLevel, QuarterLevel, RetailerLevel, CityLevel,GenderLevel* et *ALLLevel*) dont les cardinalités sont respectivement : 605, 300, 75, 15, 4, 2, 12, 4, 99, 4, 2, 5.

Nous avons également considéré une charge de 60 requêtes de jointure en étoile. L'identificateur de ligne (*RowId*) est codé sur 10 octets et la taille du buffer est 100 Mo. Nous présentons ci-dessous deux types d'expérimentations, le premier est effectué en utilisant le modèle de coût théorique tandis que le second utilise un entrepôt réel sous Oracle 10g.

5.2.4.1.1 Evaluation Théorique Nous avons effectué plusieurs expérimentations en utilisant le modèle de coût théorique. Ces expérimentations visent à comparer les performances de chaque stratégie de sélection. Nous avons implémenté trois algorithmes de sélection :

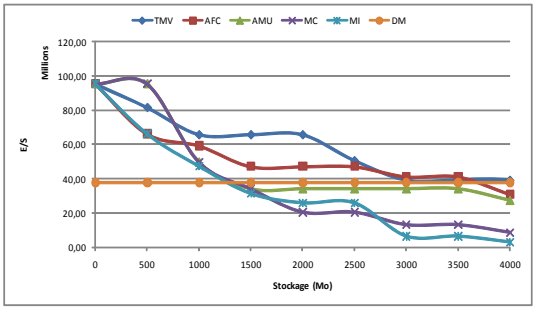


FIG. 5.19 – Ccomparaison de performance en fonction de l'espace stockage

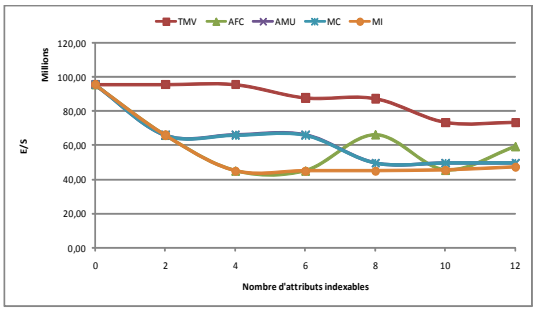


FIG. 5.20 – Nombre d'attributs indexables vs performance

1. un algorithme de sélection d'une configuration mono-attributs (MI);
2. un algorithme basé sur une technique de datamining (DM), à savoir la recherche de motifs fréquents fermés ;
3. un algorithme glouton de sélection d'une configuration d'index multi-attributs. Pour réduire la taille de la configuration initiale, nous avons implémenté quatre stratégies de réduction : TMV, AFC, AMU et MC.

L'application de notre approche de sélection d'IJB multi-attributs génère une solution initiale occupant *26,4 Go*.

Dans la première expérimentation, nous avons considéré plusieurs valeurs de l'espace de stockage nécessaire pour les index sélectionnés (de 0 à 4 Go). La figure 5.19 montre les résultats obtenus. L'algorithme DM ne prend pas en considération la contrainte d'espace dans la stratégie de sélection. Les résultats contenus dans cette figure ont été obtenus avec *minsup*=0.25. Les résultats montrent que les meilleures performances ont été obtenues par l'algorithme MI et MC. Cela s'explique par le fait que ces deux algorithmes sont basés sur un modèle de coût. Ce modèle prend en considération plusieurs paramètres comme les facteurs de sélectivité, les cardinalités, les tailles des tables de dimension, etc. Les autres algorithmes se basent chacun sur un seul paramètre. Par exemple, l'algorithme AMU est moins bon que les deux derniers algorithmes, car il ne prend en considération que la fréquence d'utilisation comme seul paramètre de sélection. Les algorithmes TMV et AFC donnent les plus mauvais résultats du fait qu'ils reposent respectivement sur la taille des tables de dimension et la cardinalité des attributs comme paramètres de sélection. Nous pouvons conclure que la sélection d'une configuration d'index doit considérer plusieurs paramètres en même temps pour garantir une meilleure performance.

L'algorithme *DM* donne des résultats qui ne prennent pas en considération l'espace de stockage, ce qui explique la ligne horizontale sur le graphe. Lorsque l'espace de stockage est réduit, la configuration générée par *DM* est meilleure, car les autres algorithmes sélectionnent peu d'index. Lorsque le quota d'espace augmente, ces algorithmes permettent de sélectionner une configuration meilleure. Pour permettre à DM de sélectionner une meilleure configuration, il faut configurer la valeur de *minsup* qui représente le *support minimum* des motifs fréquents. Pour cela, nous avons effectué des expérimentations pour différentes valeurs de *minsup*, puis, pour chaque valeur, nous avons lancé l'algorithme *DM* et nous avons calculé le coût d'exécution des requêtes ainsi que la taille des index générés. Les figures 5.21 et 5.22 montrent les résultats obtenus. Il est clair que lorsque *minsup* est petit, beaucoup d'index sont créés

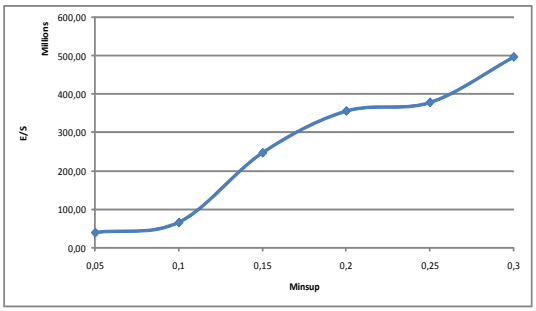


FIG. 5.21 – Effet de *minsup* sur la performance de DM

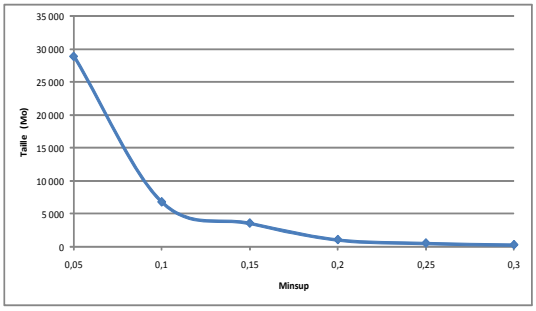


FIG. 5.22 – Effet de *minsup* sur la taille de la configuration sélectionnée par DM

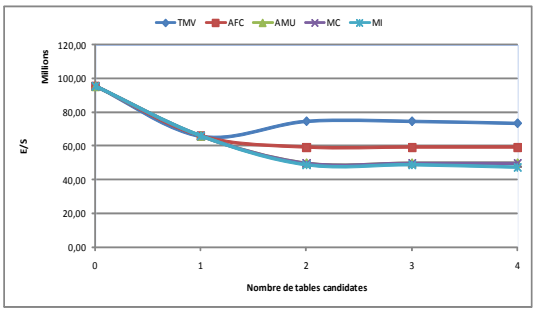


FIG. 5.23 – Nombre de tables de dimension vs performance

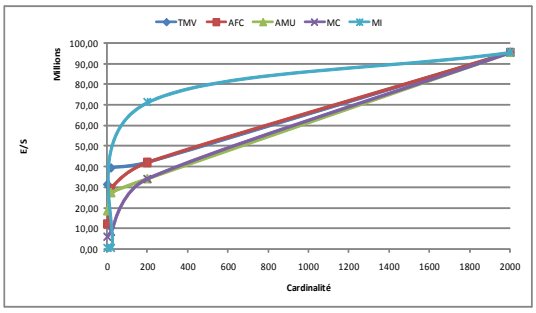


FIG. 5.24 – Cardinalité vs performance

et occupent donc plus d'espace. Cela s'explique par le fait qu'une petite valeur de *minsup* implique que la majorité des motifs sont considérés fréquents. Lorsque *minsup* est grand, peu d'index sont générés, et par conséquent moins d'espace est occupé. Il faut donc choisir une valeur de *minsup* garantissant un bon compromis entre performance et espace de stockage.

Dans la deuxième expérience, nous avons changé le nombre d'attributs candidats pour la sélection d'IJB. Nous avons changé ce nombre de 1 à 12. La figure 5.20 montre la variation du coût d'exécution en fonction du nombre d'attributs candidats. La performance des requêtes augmente lorsque le nombre d'attributs candidats augmente. Cela est du, au fait que, lorsque le nombre d'attributs candidats augmente, les index sélectionnés couvrent plusieurs attributs, par conséquent, les requêtes utilisant ces attributs exploitent les index créés et évitent d'effectuer des jointures entre la table des faits et les tables de dimension de ces attributs. Les résultats montrent aussi que lorsque le nombre d'attributs dépasse un certain nombre (8 dans notre cas), peu d'améliorations sont constatées. Cela est dû à l'espace de stockage qui, une fois saturé, ne permet pas de sélection d'autres index.

Nous avons effectué des expérimentations pour montrer l'effet du nombre des tables de dimension dans la sélection des index. Nous avons choisi 1, 2, 3 ensuite 4 tables et à chaque choix nous exécutons les différents algorithmes. La figure 5.23 montre les résultats obtenus. Généralement, lorsque le nombre de tables augmente, la performance augmente, car plus les tables sont indexées, plus de jointures sont pré-calculées. Nous constatons que l'algorithme *TMV* sort de cette règle, car en utilisant une seule table il donne de meilleurs résultats que 2 ou plus. Cela est dû au fait que lorsque le nombre de tables augmente l'algorithme élimine les tables les moins volumineuses qui sont peut-être intéressantes à indexer. La

performance ne s'améliore pas considérablement à partir d'un certain nombre de tables dû à la contrainte d'espace de stockage, qui ne permet pas de sélectionner d'autres index malgré l'ajout de tables.

Dans la dernière expérience, nous avons modifié la cardinalité des attributs indexables. Nous avons considéré que tous les attributs ont la même cardinalité et nous avons choisi plusieurs valeurs (de 2 à 2000). Pour chaque valeur, nous avons exécuté nos différents algorithmes. La figure 5.24 montre les résultats obtenus. La performance se détériore considérablement avec l'augmentation des cardinalités. Cela est dû à l'augmentation de la taille des index sélectionnés qui provoque la saturation rapide de l'espace de stockage réservé et un coût de chargement énorme des index créés. Lorsque la cardinalité dépasse un certain seuil, soit l'espace de stockage disponible ne permet de sélectionner aucun index, soit les index sélectionnés sont très volumineux, ce qui rend l'utilisation de ces index non avantageuse.

5.2.4.1.2 Validation sous Oracle 10g Pour valider notre approche sur un entrepôt de données réel, nous avons considéré celui issu du benchmark APB-1 avec la même configuration que celle utilisée dans l'évaluation théorique. Plusieurs types de requêtes ont été considérées : les requêtes de type count(*) avec et sans agrégation, les requêtes utilisant les fonctions d'agrégation comme **Sum**, **Min**, **Max**, les requêtes ayant des attributs de dimension dans la clause **SELECT**, etc.

Nous avons appliqué notre approche de sélection avec un espace de stockage de *3 Go*. Nous avons exécuté nos algorithmes pour chaque stratégie de sélection. Pour l'algorithme *DM*, nous avons considéré une solution qui respecte l'espace de stockage disponible (*minsup*=0,20). Le tableau 5.3 montre les index sélectionnés dans chaque stratégie. Pour chaque configuration sélectionnée, nous avons représenté le nombre d'index mono-attributs, multi-attributs ainsi que les attributs de dimension indexés.

Stratégie	Nombre d'IJB sélectionnés	Attributs indexés	Tables de dimension indexées
MV	4 index mono-attributs	GroupLevel, FamilyLevel, LineLevel, DivisionLevel	Prodlevel
DM	3 index mono-attributs	MonthLevel, AllLevel, RetailerLevel	Timelevel, Chanlevel, Custlevel
AFC	4 index mono-attributs, 4 index multi-attributs	QuarterLevel, DividionLevel, YearLevel, GenderLevel	Timelevel, ProdLevel, Custlevel
AMU	3 index mono-attributs	YearLevel, MonthLevel	Timelevel
MC	4 index mono-attributs, 5 index multi-attributs	QuarterLevel, YearLevel, MonthLevel, AllLevel	Timelevel, Chanlevel
MI	9 index mono-attributs	FamilyLevel, LineLevel, DivisionLevel, YearLevel, MonthLevel, QuarterLevel, GenderLevel, CityLevel, AllLevel	Prodlevel, Chanlevel, TimeLevel, Custlevel

TAB. 5.3 – Caractéristiques des configurations générées par chaque stratégie

Nous avons créé physiquement les configurations générées par chaque algorithme puis nous avons exécuté les 60 requêtes pour chaque configuration. Nous avons forcé l'utilisation des IJB en utilisant les *hint*. Pour vérifier que les IJB sont bien utilisés par Oracle pour exécuter une requête donnée, nous avons utilisé l'outil *Explain Plan* fournit avec Oracle. Cet outil permet de visualiser le plan d'exécution détaillé d'une requête. Le tableau 5.3 montre, pour chaque stratégie, la configuration d'IJB sélectionnée ainsi que les tables de dimension indexées. Chaque IJB dans ce tableau est représenté par les attributs de dimension qu'il indexe.

Les figures 5.25 et 5.26 montrent respectivement le temps d'exécution de l'ensemble des requêtes et

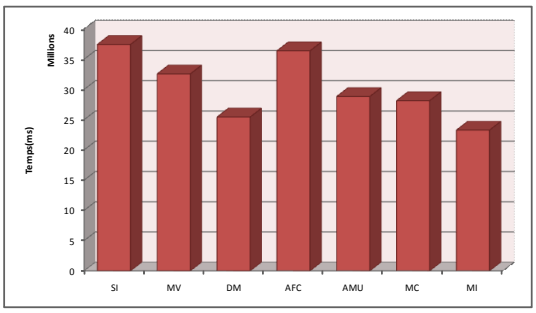


FIG. 5.25 – Temps d'exécution des requêtes sous Oracle

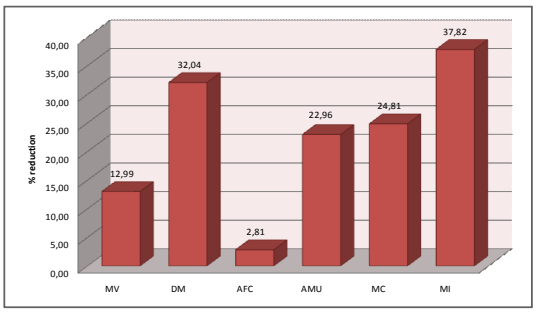


FIG. 5.26 – Pourcentage de réduction du coût d'exécution

le taux de réduction de ce coût en utilisant la configuration d'IJB issue de chaque stratégie.

L'exécution des 60 requêtes sur toutes les configurations d'index nous a permis de confirmer la grande utilité des IJB pour les requêtes de type *Count(*)*. Le coût d'exécution de ces requêtes en utilisant les IJB est négligeable devant le coût sans index. Ces requêtes sont celles les plus bénéficiaires des IJB, car aucun accès aux données n'est effectué (l'accès aux IJB suffit pour répondre à ces requêtes). Les requêtes les moins bénéficiaires des IJB sont celles référençant des attributs de dimension dans la clause SELECT ou celles ayant moins d'attributs indexés, car elles nécessitent des jointures supplémentaires entre les tables de dimension et la table des faits. Nous avons aussi constaté que les IJB multi-attributs sont plus efficaces lorsqu'ils couvrent tous les attributs indexables de la requête. Dans le cas contraire, Oracle utilise souvent des jointures par hachage car l'utilisation des IJB dégrade les performances à cause des jointures supplémentaires. L'utilisation des IJB mono-attributs est généralement bénéfique car elle permet d'utiliser les opérations logiques pour répondre à des requêtes référençant plusieurs attributs indexés.

Les résultats globaux obtenus sous Oracle montrent que l'approche de sélection des IJB mono-attributs ainsi que l'algorithme *DM* donnent les meilleurs résultats. Les index générés par ces deux algorithmes sont tous mono-attributs et couvrent respectivement trois et quatre tables de dimension. Pour l'algorithme de génération d'IJB multi-attributs, la stratégie MC est la plus bénéfique car elle repose sur un modèle de coût et prend en considération plusieurs paramètres. La stratégie *AMU* donne des résultats proches de ceux donnés par MC. Cela est dû au fait que les attributs et la table indexée sont les plus utilisés par les 60 requêtes, ce qui permet d'optimiser une bonne partie d'entre elles. Cela montre d'un coté qu'il est intéressant de considérer la fréquence d'utilisation dans la sélection d'index et de l'autre coté de considérer d'autres paramètres (taille des tables, les cardinalités, etc.). Les résultats montrent aussi que les stratégies TMV et *AFC* donnent des résultats de très mauvaise qualité. La stratégie TMV élimine les tables les moins volumineuses, or leur indexation peut être très bénéfique (la table *Chanlevel* par exemple, est indexée dans *MC*, *MI* et *DM* qui sont les stratégies donnant les meilleures performances). La stratégie *AFC* élimine les attributs de forte cardinalité du processus d'indexation, or indexer un attribut de faible cardinalité rarement utilisé peut être bénéfique à quelques requêtes seulement mais pas pour toute la charge. Par exemple l'algorithme *DM* indexe des attributs *RetailerLevel* et *MonthLevel* ayant une cardinalité importante par rapport aux autres (99 et 12) mais cet algorithme donne de meilleurs résultats car ces attributs sont utilisés par un nombre important de requêtes.

5.2.5 Conclusion

Dans cette partie nous nous sommes intéressés aux index de jointure, nous en avons présenté deux : les index de graphe de jointure adaptés pour optimiser les requêtes réécrites en fonction des vues matérialisées et les index de jointure binaire proposés pour optimiser les requêtes de jointure en étoile.

Nous avons étudié en détails les index de jointure en étoile, vue leur efficacité pour les requêtes décisionnelles et leurs habilités d'être compressés. Nous avons étudié leur problème de sélection et sa complexité. Compte tenue de cette complexité, nous avons proposé deux types d'algorithmes: (1) un algorithme d'élagage des attributs non pertinents pour l'indexation en utilisant une technique de fouille de données enrichie par des caractéristiques physiques de l'entrepôt de données et (2) des algorithmes de sélection des index finaux. Nous avons mené plusieurs expérimentations en utilisant un modèle de coût théorique, puis sous *Oracle* afin de comparer nos algorithmes et leurs stratégies d'élimination.

Chapitre 6

Sélection multiple des structures d'optimisation

Sommaire

1 Exploitation de similitudes entre des index de jointure binaire et la fragmentation pour réduire le coût de maintenance

Publications : [20, 58]

Dans le chapitre précédent, nous avons présenté deux structures d'optimisation, une non redondante : la fragmentation horizontale et une autre redondante : les index de jointure binaires. Nous avons ainsi identifié de fortes similarités entre elles. La fragmentation horizontale primaire des tables de dimension et la dérivée de la table des faits optimisent les sélections et les jointures, respectivement. D'un autre côté, les IJBs optimisent également les jointures et les sélections. Les deux structures sont *concurrentes sur la même ressource* représentant *l'ensemble des attributs de sélection définis sur les tables de dimension* (*EAS*). Deux caractéristiques les différencient : (a) la fragmentation horizontale peut être définie sur

n'importe quel(s) attribut(s) de *EAS*, tandis qu'un IJB est défini sur un ou plusieurs attributs de *EAS*, mais de faible cardinalité. (b) Les IJB sont contraints par l'espace de stockage (ou coût de maintenance) tandis que la fragmentation est contrainte par le nombre de fragments.

Pour montrer cette similitude, nous considérons l'exemple suivant.

Exemple. Soit un entrepôt de données avec trois tables de dimension (*Temps*, *Client* et *Produit*) et une table des faits *Ventes*. La population de cet entrepôt est décrite dans la Figure 6.1. Sur ce schéma, considérons la requête suivante:

```
SELECT Count(*)
FROM Customer, Product P, Time T, Sales S
WHEERE C.City='Poitiers'
AND P.Range ='Beauty'
AND T.Month='June'
AND P.PID=S.PID AND C.CID=S.CID AND T.TID=S.TID
```

Elle contient trois prédicats de sélection définis sur les attributs : *City, Range* et *Month* et trois jointures. Ces attributs sont candidats à la fois pour la définition de schéma de fragmentation et des index de jointure binaires (du fait qu'ils ont une faible cardinalité). Pour optimiser totalement cette requête, deux modes d'exécution sont possibles: (i) l'utilisation de la fragmentation horizontale seule (*FHSEUL*) et (ii) l'utilisation des index de jointure binaires seuls (*IJBSEUL*).

C_RID	CID	Name	City
6	616	Gilles	Poitiers
5	515	Yves	Paris
4	414	Patrick	Nantes
3	313	Didier	Nantes
2	212	Eric	Poitiers
1	111	Pascal	Poitiers

Customer

P_RID	PID	Name	Range
6	106	Sonoflore	Beauty
5	105	Clarins	Beauty
4	104	WebCam	Multimedia
3	103	Barbie	Toys
2	102	manure	Gardening
1	101	SlimForm	Fitness

Product

T_RID	TID	Month	Year
6	11	January	2003
5	22	February	2003
4	33	March	2003
3	44	April	2003
2	55	May	2003
1	66	June	2003

Time

S_RID	CID	PID	TID	Amount
1	616	106	11	25
2	616	106	66	28
3	616	104	33	50
4	545	104	11	10
5	414	105	66	14
6	212	106	55	14
7	111	101	44	20
8	111	101	33	27
9	212	101	11	100
10	313	102	11	200
11	414	102	11	102
12	414	102	55	203
13	515	102	66	100
14	515	103	55	17
15	212	103	44	45
16	111	105	66	44
17	212	104	66	40
18	515	104	22	20
19	616	104	22	20
20	616	104	55	20
21	212	105	11	10
22	212	105	44	10
23	212	105	55	18
24	212	106	11	18
25	313	105	66	19
26	313	105	22	17
27	313	106	11	15

Sales

FIG. 6.1 – Un exemple de Population de l'Entrepôt

– *FHSEUL*: dans ce mode, les tables de dimension *Client*, *Temps* et *Produit* sont fragmentées en utilisant *City*, *Month*, *Range*, respectivement. Dans ce cas, $EAS = \{City, Month, Range\}$. Leurs schémas sont propagés sur la table des faits *Ventes* qui sera décomposée en 90 fragments [16], où chaque fragment des faits $Ventes_i$ $(1 \leq i \leq 90)$ est défini comme suit:

$$Ventes\_i = Ventes \ltimes Client_j \ltimes Temps_k \ltimes Produit_m$$

$(1 \leq j \leq 3)$, $(1 \leq k \leq 6)$, $(1 \leq m \leq 5)$. La figure 6.2c donne la population du fragment *Ventes_BJP*

[16] Supposons que nous avons trois villes, 6 mois et 5 catégories.

correspondant aux ventes de produits de beauté réalisées par les clients vivant à *Poitiers* durant le mois de *Juin*. La requête ci dessous est réécrite comme suit :

SELECT Count(*) FROM SALES *PARTITION*(SALES_BJP)[17]

Pour exécuter cette requête, l'optimiseur charge en mémoire le fragment *Ventes_BJP*. Ce mode permet d'éviter les trois opérations de jointure et de sélection ce qui représente une grande performance.

– IJBSEUL: Dans ce mode, un IJB peut être défini entre la table des faits et les trois tables de dimension sur les attributs *Ville*, *Mois* et *Catégorie* comme suit:

```
CREATE BITMAP INDEX vente_client_ville_prod_cat_time_mois_bjix
ON VENTES(CLIENT.Ville, PRODUIT.Catégorie, TEMPS.Mois)
FROM VENTES V, CLIENT C, TEMPS T, PRODUIT P
WHERE V.CID = C.CID AND V.PID = P.PID AND V.TID = T.TID
```

La Figure 6.2a montre l'index généré. Pour exécuter la requête, l'optimiseur doit accéder aux bitmaps correspondant aux trois valeurs des attributs (*Poitiers*, *Beauty* et *June*), où un "et logique" est utilisé. Aucune jointure n'est exécutée.

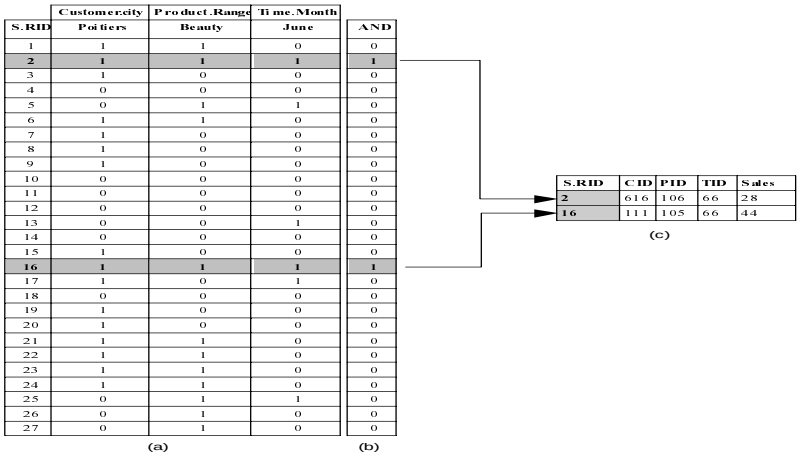


FIG. 6.2 – (a) Index de jointure binaire (b) Le résultat de l'opérateur AND, (c) Fragment des faits

L'exemple montre l'efficacité des deux structures pour optimiser ce type de requêtes. Vue la similarité entre elles, le concepteur peut privilégier la fragmentation horizontale, qui présente un gain de stockage et de mise à jour considérable. D'autres index définis sur d'autres attributs peuvent être sélectionnés afin d'optimiser d'autres requêtes. Sinon, il peut se contenter des index sans fragmenter l'entrepôt. En ignorant ces similarités, le concepteur risque d'avoir le scénario suivant: fragmenter l'entrepôt de données en utilisant l'ensemble EAS, ensuite l'indexer en utilisant l'ensemble des attributs de sélection candidats à l'indexation $EACI$ ($EACI \subseteq EAS$). Les IJB définis sur l'entrepôt non fragmenté (appelés des *IJB globaux*) peuvent être alignés aux fragments générés. Un index est aligné s'il est partitionné de la même façon que la table qu'il référence [191]. Dans ce cas, il est appelé un *index local*.

Dans la thèse de Kamel BOUKHALFA, nous avons proposé une approche pour la sélection multiple de schémas de fragmentation et d'index de jointure binaire.

Avant de la présenter, nous proposons, dans la section suivante une formalisation du problème de la

[17] Nous utilisons une syntaxe d'Oracle.

sélection multiple de schéma de fragmentation et d'indexation comme un problème d'optimisation avec contraintes.

2 Formalisation du problème de sélection des schéma de FH et d'IJB

Étant donné :

- Un entrepôt de données modélisé par un schéma en étoile composé d'une table de faits F et de d tables de dimension $\{D_1, D_2, \cdots, D_d\}$;
- Une charge de requêtes $Q = \{Q_1, Q_2, \cdots, Q_m\}$ où chaque requête Q_i possède une fréquence d'accès f_i;
- Un seuil W représentant le nombre de fragments générés par le processus de fragmentation;
- Un espace de stockage S réservé aux index de jointure binaires.

L'objectif recherché est de sélectionner un schéma de fragmentation horizontale (SF) et une configuration d'IJB (CI) tels que :

- Le coût d'exécution de Q en présence de SF et CI soit réduit;
- Le nombre de fragments générés par le schéma de fragmentation SF soit inférieur ou égal à W ($N_{SF} \leq W$)
- La taille de CI est inférieure ou égale à l'espace de stockage réservé ($Taille(CI) \leq S$).

Une implémentation conjointe pour répondre à ce problème est difficile à mettre en oeuvre vu que les deux sous problèmes (la sélection des schémas de fragmentation et d'indexation) n'ont pas la même contrainte. L'implémentation indépendante est à exclure, car elle n'exploite pas les similarités entre les deux structures. Il reste l'implémentation séquentielle qui peut exploiter ces similarités. Elle consiste d'abord à fragmenter l'entrepôt de données en utilisant EAS. Notons que la fragmentation est contrainte par un seuil représentant le nombre de fragments générés, en conséquence certains attributs de EAS pourraient ne pas être utilisés par le processus de fragmentation. Soit $EASNF$ l'ensemble de ces attributs. Sur cet ensemble, seuls les attributs de faible cardinalité ($EANFI$) sont utilisés pour sélectionner les index de jointure binaires.

Exemple. Dans l'exemple précédent, la fragmentation horizontale a généré 90 fragments. Supposons que le seuil de fragmentation est de 18 fragments des faits. Pour satisfaire cette contrainte, un schéma de fragmentation défini sur deux attributs de dimension, à savoir *Ville* et *Mois* ($FASET = \{Month, City\}$) est sélectionné. En conséquence, l'attribut *Catégorie* de la table de dimension *PRODUIT* n'est pas pris dans le processus de fragmentation. Un index sur *Catégorie* peut être défini sur le schéma fragmenté afin d'assurer une bonne performance de cette requête.

2.1 Analyse de notre approche

La solution proposée permet de réduire l'espace de recherche du problème de sélection des index de jointure binaire. Cette réduction concerne le nombre des attributs indexables [20, 58]. Si un attribut est utilisé dans le processus de fragmentation, il ne le sera pas dans le processus d'indexation. Ce qui représente une réduction importante.

Une autre réduction peut concerner le nombre de requêtes utilisées par le processus d'indexation. Si

une requête est identifiée comme bénéficiaire par le processus de fragmentation, elle peut être écartée par le processus d'indexation. Cette réduction est similaire à la compression de requêtes.

Pour illustrer ces deux réductions, nous proposons une résolution, appelée, FH&IJB, au problème de sélection multiple avec une implémentation séquentielle réduisant le nombre d'attributs indexables et le nombre de requêtes.

2.2 Résolution de sélection multiple de la fragmentation et des IJB

Les étapes de cette résolution sont [56, 58] :

1. l'énumération de l'ensemble des attributs de sélection EAS à partir de l'ensemble de requêtes de départ $\mathbb{Q}$;
2. fragmenter l'entrepôt de données en utilisant EAS en utilisant n'importe quel algorithme satisfaisant la contrainte de maintenance W. Soit $AFSET$ ($AFSET \subseteq AD$) l'ensemble des attributs de sélection réellement utilisés par le schéma final de fragmentation;
3. pour prendre en considération les similarités entre les deux structures, nous identifions les requêtes bénéficiaires de la fragmentation horizontale. Pour déterminer si une requête Q_j ($1 \leq j \leq m$) est bénéficiaire, nous définissons la métrique (appelée, métrique de tuning) de la façon suivante:

$$taux(Q_j) = \frac{C(Q_j, SF)}{C(Q_j, \phi)} \tag{1}$$

   où $C(Q_j, SF)$ et $C(Q_j, \phi)$ représentent le coût d'exécution de la requête Q_j sur un schéma partitionné SF et non partitionné, respectivement. Le concepteur a le droit de paramétrer cette métrique en utilisant un seuil λ défini comme suit:

   Si $taux(Q_j) \leq \lambda$ alors la requête Q_j est bénéficiaire, sinon non bénéficiaire.
4. l'identification des attributs indexables candidats: au lieu de définir les index de jointure binaires sur l'ensemble de requêtes de départ, ils sont définis seulement sur les requêtes non bénéficiaires $Q^{nobenefit} = \{Q'_1, Q'_2, \cdots, Q'_l\}$. A partir de cet ensemble, nous extrairons l'ensemble des attributs indexables candidats, dénoté par ($EnsAIJB$). Cet ensemble ne contient que les attributs de faible cardinalité et qui ne sont pas utilisés par le schéma de fragmentation final.

2.3 Une conséquence de notre sélection multiple : tuning de l'entrepôt de données

Notre approche permet aussi de tuner l'entrepôt de données en utilisant un paramètre λ, dans le cas où le mode de sélection multiple des schémas de fragmentation et d'indexation FH&IJB est utilisé. Ce paramètre permet d'identifier l'ensemble des requêtes non bénéficiaires du processus de fragmentation et, donc, qui nécessite la création d'un ensemble d'IJB pour les optimiser. Ce paramètre couplé avec le seuil W représentant le nombre de sous-schémas en étoile voulu peuvent être utilisés pour partager l'ensemble des attributs candidats entre les deux techniques. Ce partage permet à l'administrateur de privilégier une structure à une autre. Par exemple, si le concepteur veut privilégier la fragmentation horizontale sur les index, il peut fixer W et λ à des grandes valeurs. Cela peut être utile lorsque l'administrateur dispose de peu d'espace disque ou d'une charge de requêtes contenant plusieurs requêtes de mises à jour.

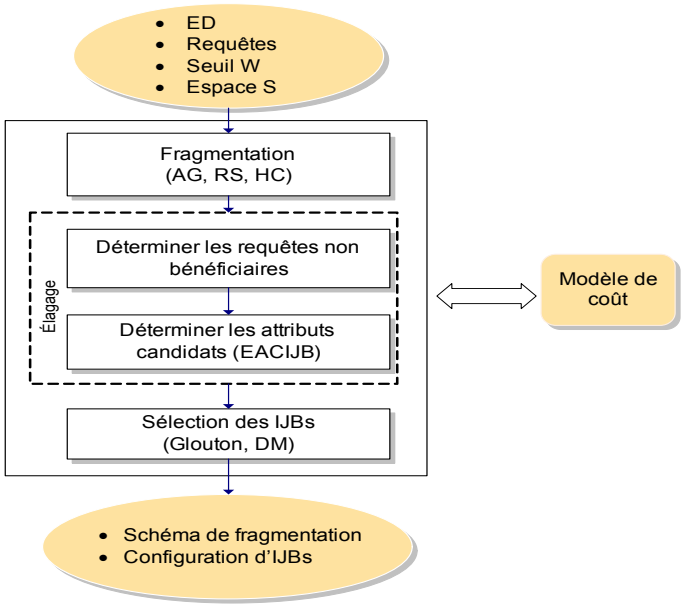


FIG. 6.3 – Notre approche conjointe

2.4 Une autre conséquence : La fragmentation horizontale = Anti-indexation

Dans les travaux de thèse de Kamel BOUKHALFA [56], nous avons présenté une démarche pour sélectionner conjointement les schémas de fragmentation et d'indexation en exploitant les similarités entre les deux structures.

Durant la conférence de DAWAK'09 à Linz en septembre 2009, nous avons eu l'occasion d'assister à la présentation de Dr. Goetz GRAEFE membre de Advanced Database Group at HP Labs, U.S.A. et pionnier de l'optimisation de requêtes, intitulé : *Fast Loads and Fast Queries* [103]. L'idée sous-jacente de son travail est de remplacer les index (efficaces mais coûteux en stockage et en maintenance) par d'autres structures similaires (non redondantes). Ce remplacement ne doit pas détériorer significativement les performances de requêtes. Cette idée est partagée par plusieurs éditeurs de bases de données, comme le système Netezza (www.netezza.com), considéré comme un anti-index. Après une longue discussion avec Goetz GRAEFE, nous pourrions proposer la fragmentation horizontale dérivée et primaire comme une *solution de remplacement des index de jointure binaires*. Cette solution pourrait être utilisée par les applications demandant des mises à jour importantes, comme les entrepôts de données en temps réel. La figure 6.4 illustre cette idée.

2.5 Expérimentation

Nous avons mené des expérimentations pour valider nos propositions. Comme pour les structures précédemment développées dans le cadre de la sélection isolée, nous avons effectué deux types d'expérimentations: une basée sur un modèle de coût et une validation sous Oracle10G. Quatre approches ont été évaluées : (a) sans technique d'optimisation, que nous appelons (a) *SansTO*, (b) *FHSEULE* : seule la fragmentation est utilisée, (c) *IJBSEULS* : seuls les IJB sont sélectionnés sans fragmentation et (d)

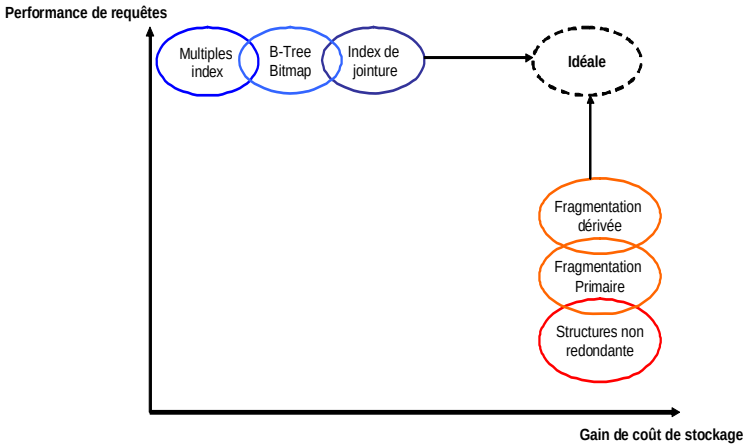


FIG. 6.4 – Compromis entre les structures redondantes et non redondantes

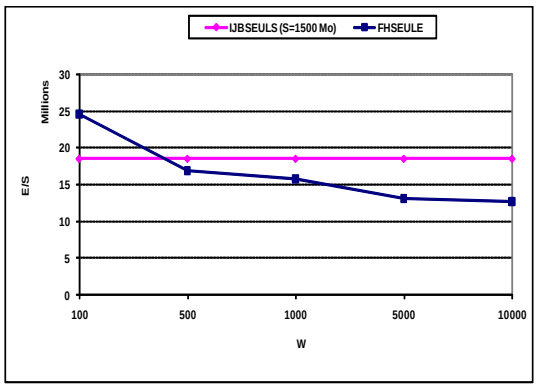


FIG. 6.5 – IJBSEULS vs HPSEULE(Seuil W)

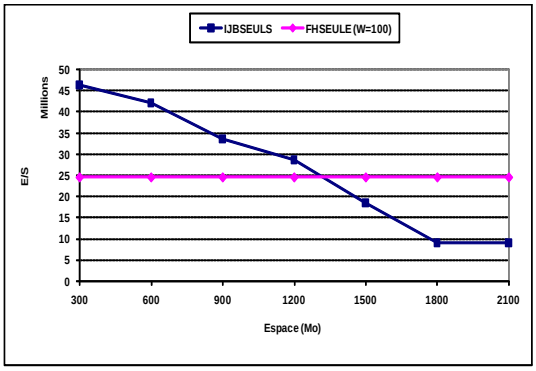


FIG. 6.6 – HPSEULE vs IJBSEULS(Stockage)

FH&IJB : notre approche conjointe.

Pour la sélection d'un schéma de fragmentation horizontal, nous avons utilisé l'algorithme génétique. L'algorithme glouton est utilisé pour sélectionner une configuration d'index mono-attributs.

Dans la première expérimentation, nous comparons les deux approches FHSEULE et IJBSEULS. La figure 6.5 montre la performance des 60 requêtes en termes de coût d'exécution en fonction de l'espace de stockage disponible. Nous avons fait varier cet espace de 300 jusqu'à 2 100 Mo. Nous avons également exécuté l'algorithme génétique pour la fragmentation horizontale avec un seuil de maintenance $W = 100$. Les résultats montrent que FHSEULE est plus performante que IJBSEULS lorsque l'espace de stockage est inférieur à 1 200 Mo. Mais lorsque nous réservons plus d'espace aux IJB, IJBSEULS est plus performante surtout pour les 15 requêtes utilisant la fonction count(*). Cela est dû au fait que ce type de requêtes charge uniquement les IJB et n'accède pas à la table des faits. Cette expérimentation est très intéressante car elle donne des recommandations à l'administrateur pour bien administrer son entrepôt. Pour des requêtes utilisant des opérations count(*) sans agrégation, il est plus bénéfique d'utiliser les IJB seuls.

La figure 6.6 montre la performance de la fragmentation lorsque la contrainte de maintenance est

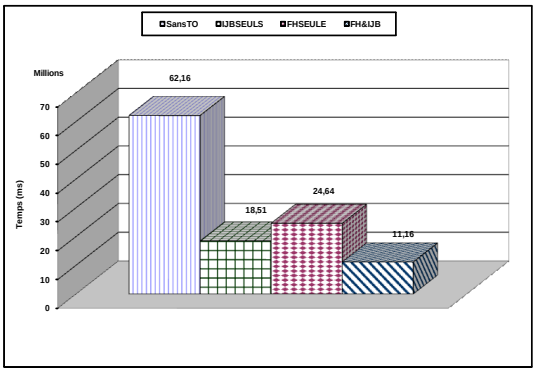


FIG. 6.7 – Comparaison des trois approches

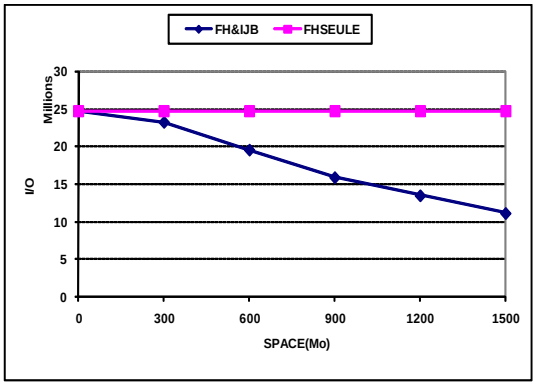


FIG. 6.8 – Effet de S sur notre approche

relâchée. Les IJB ont été sélectionnés avec un espace de 1 000 Mo. L'augmentation de la contrainte de maintenance implique qu'un grand nombre d'attributs de sélection participent dans le processus de fragmentation. Par conséquent, la plupart des requêtes bénéficie du processus de fragmentation. Dans ce cas, FHSEULE est bénéfique pour la majorité des requêtes.

Pour évaluer notre approche FH&IJB, nous considérons le scénario suivant : nous utilisons l'algorithme génétique avec $W = 100$ et nous fixons $\lambda = 0.6$ pour identifier les requêtes bénéficiaires du processus de fragmentation. Parmi les 60 requêtes de départ, 23 sont non bénéficiaires tandis que 37 sont bénéficiaires. Le schéma de fragmentation obtenu fragmente l'entrepôt en utilisant 5 attributs : *ClassLevel*, *FamilyLevel*, *MonthLevel*, *AllLevel*, *RetailerLevel*. Ces attributs ne seront pas considérés par l'algorithme glouton pour la sélection des IJB. Ce dernier utilise 7 attributs : *GroupLevel*, *LineLevel*, *DivisionLevel*, *YearLevel*, *QuarterLevel*, *CityLevel* et *GenderLevel*. L'algorithme glouton utilisé pour sélectionner des IJB prend en entrée ces 7 attributs, les 23 requêtes non bénéficiaires et un espace de stockage de 1500 Mo. Trois IJB ont été sélectionnés par cet algorithme.

La figure 6.7 montre la performance de chaque approche. *FH&IJB* est la meilleure que *SansTO* et *HPSEULE*. Mais elle est légèrement meilleure que *IJBSEULS*, car un nombre important des requêtes de notre banc d'essai possède des fonctions de type count(*) sans agrégations. Rappelons que les IJB sont performants pour ce type de requêtes. Un autre résultat intéressant est le fait que *FH&IJB* est moins gourmande en espace de stockage (1 200 Mo pour le stockage des IJB ce qui représente 20% d'économie d'espace contre 1 500 Mo pour IJBSEULS).

Nous menons une autre expérience pour étudier l'effet de l'espace de stockage sur notre approche *HP&IJB*. Nous varions cet espace de 0 (où aucun IJB n'est sélectionné) à 1500 Mo (la majorité des IJB peuvent être sélectionnés). Nous avons comparé notre approche avec *FHSEULE* (W=100).
La figure 6.8 montre les résultats obtenus. Notre approche est plus performante que *FHSEULE* du fait que la majorité des IJB ont été sélectionnés.

La performance de *FH&IJB* dépend largement du choix de λ qui détermine les requêtes bénéficiaires du processus de fragmentation. Pour mesurer l'impact de ce paramètre, nous varions sa valeur de 0 (toutes les requêtes sont non bénéficiaires) à 1 (toutes les requêtes sont bénéficiaires). La figure 6.9 montre les résultats obtenus. Lorsque λ est égal à 0, HP&IJB réduit le coût généré par FHSEULE de 50% en allouant 741 Mo d'espace pour les index sélectionnés. Lorsqu'il atteint 1, la performance de notre approche

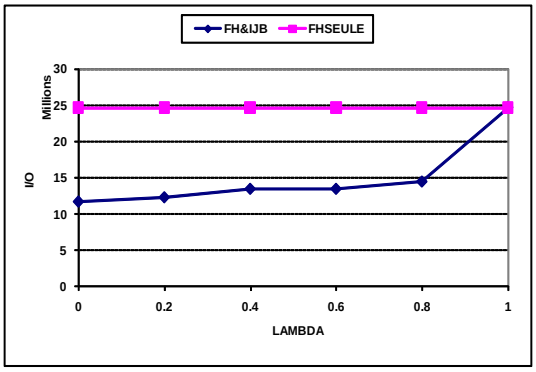


FIG. 6.9 – Effet de λ HP&IJB

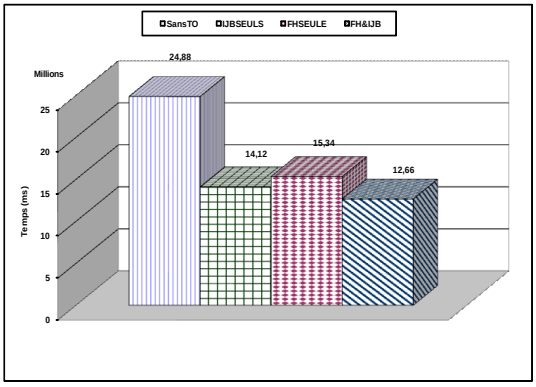


FIG. 6.10 – Validation sous Oracle 10g

est équivalente à celle de FHSEULE (aucun index n'est créé). Cette expérimentation montre la grande importance de ce paramètre durant la conception physique. Il constitue un outil pour *tuner* l'entrepôt de données. L'administrateur peut utiliser ce paramètre de la façon suivante : s'il privilégie la fragmentation (par manque d'espace disque par exemple), il relâche W et donne une valeur à λ proche de 1. Par contre s'il possède assez d'espace de stockage et moins de requêtes de mise à jour, il donne une valeur à λ assez proche de zéro pour prendre en compte le maximum d'attributs candidats pour l'indexation.

Nous avons effectué une validation sous Oracle10G en utilisant les données issues du Benchmark APB-1. Pour calculer le temps d'exécution réel de chaque requête, nous utilisons l'outil Aqua Data Studio[18]. Ce temps d'exécution est multiplié par la fréquence d'accès de la requête correspondante. Nous avons évalué quatre scenarii, FHSEULE, IJBSEULS, FH&IJB et SansTO.

- *SansTO* : nous avons exécuté les 60 requêtes d'origine sur l'entrepôt non fragmenté.
- *HPSEULE* : nous avons fixé $W = 100$ et nous avons lancé l'algorithme génétique. Le schéma de fragmentation obtenu génère 96 sous-schémas en étoile en utilisant 5 attributs. Le module de fragmentation partitionne l'entrepôt de données initial en utilisant le schéma obtenu. Les 60 requêtes sont réécrites sur le schéma fragmenté par le module de réécriture. Les requêtes réécrites ont été exécutées sur l'entrepôt fragmenté.
- *IJBSEULS* : Nous avons lancé l'algorithme glouton avec 12 attributs candidats, l'algorithme sélectionne 9 IJB. Le coût de stockage théorique de ces index est de 3 Go. L'ensemble des 60 requêtes sont exécutées en utilisant les index créés à l'aide de l'ajout des Hints.
- *FH&IJB* : nous avons considéré le schéma de fragmentation obtenu par FHSEULE et nous avons lancé l'algorithme glouton avec 7 attributs candidats, 23 requêtes non bénéficiaires, $\lambda = 0.6$ et une capacité de stockage de 1 Go. Quatre IJB ont été sélectionnés où chaque index est créé sur un attribut indexable. Les requêtes bénéficiaires sont réécrites sur le schéma fragmenté par le module de réécriture et ensuite exécutées sur l'entrepôt fragmenté. Les requêtes non bénéficiaires sont réécrites sur le schéma fragmenté et ensuite exécutées sur l'entrepôt de données fragmenté et indexé en utilisant les Hints[19]. Notons que le buffer est vidé après chaque exécution pour éviter que l'exécution d'une requête n'influence l'exécution de la suivante.

[18] www.aquafold.com

[19] Un hint est un moyen de forcer l'optimiseur de requête à choisir un plan d'exécution. Un exemple d'utilisation d'un hint sous Oracle est : SELECT /*+ INDEX(EMPLOYÉ INDEX_SEXE)*/ nom, adresse FROM EMPLOYÉ WHERE SEXE = 'F'

La figure 6.10 montre les résultats obtenus, ils sont similaires à ceux obtenus dans notre étude théorique. FH&IJB est plus performante que les autres approches et économise 2 Go d'espace de stockage. Cet espace peut être utilisé pour créer d'autres structures redondantes comme les vues matérialisées. IJBSEULS donnent de meilleurs résultats que FHSEULE du fait de la présence d'un grand nombre de requêtes de type count sans agrégations.

2.6 Conclusion

Dans cette partie nous avons clairement identifié la similarité entre les index de jointure binaire et la fragmentation horizontale dérivée. Les deux optimisent les opérations de sélection et de jointure, opérations souvent présentes dans les requêtes décisionnelles. Les problèmes de sélection d'un schéma de fragmentation et d'indexation sont connus comme des problèmes NP-complet. Les travaux actuels sur la sélection conjointe des techniques d'optimisation offrant une forte similarité proposent soit un algorithme parcourant l'espace de recherche des techniques concernées soit une sélection séquentielle. Notre approche de combinaison est un peu différente, car son objectif est d'utiliser la fragmentation horizontale dérivée comme un moyen de réduction de l'espace de recherche du problème de sélection des index de jointure binaires. Rappelons que la fragmentation et l'indexation sont en compétition sur le même ensemble d'attributs de sélection. Dans le cas où la fragmentation utilise un sous ensemble de l'ensemble initial, l'indexation n'utilise que les attributs qui ne sont pas concernés par la fragmentation.

Notre démarche de combinaison consiste d'abord à partitionner l'entrepôt de données, exécuter l'ensemble de requêtes sur l'entrepôt fragmenté, identifier les requêtes non bénéficiaires de la fragmentation, et indexer l'entrepôt fragmenté en ne prenant en compte que les requêtes non bénéficiaires. Nous avons établi un seuil appelé paramètre de tuning déterminant les requêtes bénéficiaires.

Nous avons mené plusieurs expérimentations pour valider notre approche en utilisant d'abord un modèle de coût mathématique. Les résultats obtenus par ce dernier ont ensuite été validés avec Oracle 10g. Les résultats démontrent que notre approche réduit considérablement le coût d'exécution des requêtes et économise l'espace de stockage. Nous avons montré aussi qu'elle peut être utilisée pour tuner l'entrepôt de données en permettant aux administrateurs, à travers un paramètre de tuning, d'améliorer les performances de requêtes non bénéficiaires du processus de fragmentation.

3 Interaction entre les vues matérialisées et les index

Publications : [32].

Dans la partie précédente, nous avons étudié la similarité entre une structure redondante (les index de jointure binaire) et une structure non redondante (la fragmentation horizontale). Dans cette partie, une étude de similarité entre deux structures d'optimisation redondantes qui sont les index et les vues matérialisées est proposée. Ce sont deux structures nécessitant un espace de stockage et causant des problèmes de mise à jour. Notons que lors de la conception physique, le concepteur (ou l'administrateur) a une idée sur l'espace global nécessaire pour stocker ces deux structures. Mais il n'a pas d'idée précise sur le quota réservé à chaque structure. La majorité des travaux sur leur sélection multiple supposent que la répartition de l'espace global entre les vues matérialisées et les index est préalablement faite, mais sans donner les critiques de cette répartition [206].

Dans nos travaux de thèse [10, 32], nous nous sommes intéressés au problème de la gestion de la ressource représentant l'espace disque dédié aux vues matérialisées et les index, autrement dit le problème de distribution de l'espace disque entre les vues et les index, avec pour objectif l'amélioration de la performance des requêtes.

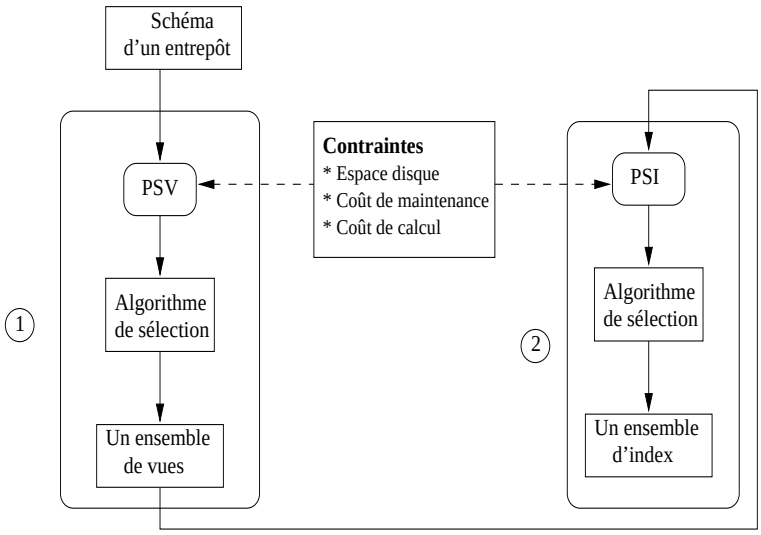


FIG. 6.11 – Le processus de sélection des vues et des index

Pour mieux comprendre ce problème, considérons le scénario suivant: soit S l'espace disque global que l'administrateur possède pour sélectionner un ensemble de vues V et un ensemble d'index I. Pour répartir S entre V et I, l'administrateur a trois possibilités:

1. Attribuer la totalité de l'espace disque S aux vues matérialisées seules. Toutes les requêtes décisionnelles sont alors exprimées en fonction des vues et des relations de base. Cette attribution n'est pas efficace. En effet, plusieurs travaux ont démontré que des vues matérialisées seules sont insuffisantes pour accélérer toutes les requêtes décisionnelles [28, 135, 192]. Cette possibilité est de ce fait écartée.
2. Attribuer la totalité de S aux index seuls. Toutes les requêtes décisionnelles sont ainsi traitées

en fonction des index et des relations de base. Or, interroger un entrepôt de données sans vues matérialisées ne garantit aucune performance [133, 109, 222, 71]. En conséquence, les index seuls sont insuffisants pour garantir une bonne performance des requêtes décisionnelles. Cette possibilité est donc également écartée.

3. La dernière possibilité consiste en un compromis entre les deux premières possibilités qui devrait garantir une bonne performance et de ce fait est l'objet de notre étude.

   Plus formellement, l'administrateur doit déterminer une fraction f ($0 \leq f \leq 1$) tel que le quota d'espace définie par $f \times S$ soit réservé aux vues matérialisées (ou aux index) et le quota d'espace défini par $(1 - f) \times S$ soit réservé aux index (ou aux vues).
   Trois possibilités de distribution sont possibles: (i) une distribution *uniforme*, (ii) une distribution *aléatoire* et (iii) une distribution *calculée*.

   Définition 10

   La distribution uniforme présente une répartition équitable de S entre les vues et les index ($f = 0.5$).

   Définition 11

   La distribution aléatoire présente une répartition quelconque de S.

   Définition 12

   La distribution calculée consiste en l'attribution de l'espace aux vues et aux index, en respectant certains critères.

   Certes, les distributions uniforme et aléatoire sont faciles à obtenir, mais elles présentent plusieurs inconvénients :
   – elles ne prennent pas en compte l'interdépendance mutuelle entre les vues et les index.
   – la distribution aléatoire peut allouer plus d'espace que nécessaire aux vues (ou aux index). Cette allocation se fait alors au détriment de l'espace des index et vice versa. Notons que pour une exécution efficace des requêtes, il est parfois préférable d'avoir plus d'espace pour les vues que pour les index (ou vice-versa).
   – ces deux distributions ne possèdent aucune métrique garantissant la réduction du coût de l'ensemble de requêtes.

Le problème de redistribution d'espace entre les vues et les index doit être considéré après les opérations de mise à jour. Ces dernières interviennent au niveau des tables des sources de l'entrepôt, et les changements correspondants doivent être répercutés sur les vues matérialisées et les index [117]. Ainsi les tailles des vues et des index peuvent augmenter ou diminuer. Il est donc nécessaire de revoir la distribution initiale de l'espace global entre les vues et les index afin de garantir une meilleure performance. Un autre problème pouvant entraîner une redistribution de l'espace est le changement des requêtes de départ (ajout/suppression de requêtes, changement de fréquence d'accès des requêtes, etc). Cela est dû au fait que la plupart des algorithmes de sélection des vues et des index sont basés sur des requêtes connues a priori.

Dans le cadre de nos travaux de thèse, nous avons proposé des solutions au problème de distribution de l'espace entre les vues et les index dans les cas *statique* et *dynamique*. L'intuition sous-jacente de notre solution est la suivante:

Initialement, l'administrateur estime les deux quotas d'espace S^V et S^I pour les vues matérialisées et les index respectivement, ($S = S^V + S^I$). Cette estimation peut être établie en utilisant une distribution uniforme ou aléatoire. En fonction du quota réservé aux vues S^V, l'administrateur sélectionne un ensemble de vues $V = \{V_1, V_2, ..., V_s\}$ en utilisant l'un des algorithmes de sélection de vues contraint par la contrainte d'espace.

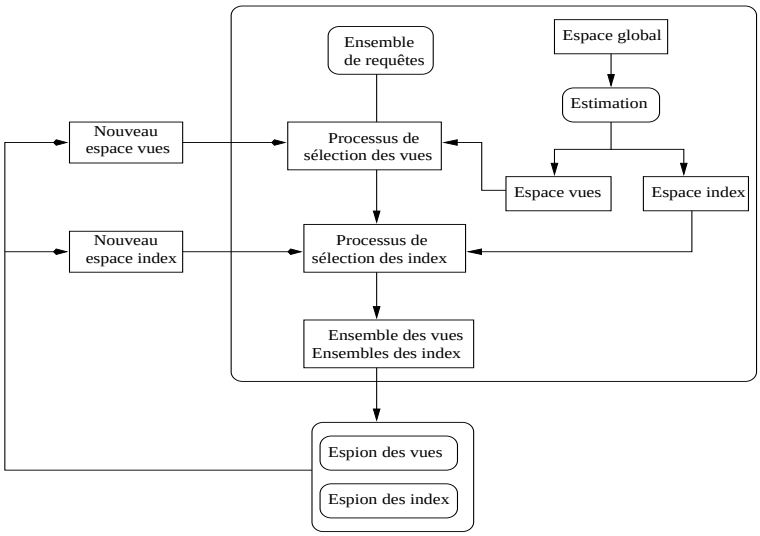


FIG. 6.12 – L'intuition de notre approche itérative

Une fois les vues sélectionnées, un ensemble d'index I est construit $I = \{I_1, ..., I_r\}$ en utilisant le quota réservé aux index (S^I). Rappelons que deux types d'index sont possibles: (i) les index mono table utilisés séparément sur des tables de base ou des vues matérialisées, (ii) les index multi table définis conjointement sur des tables de base et des vues matérialisées.

En conséquence, nous obtenons une solution initiale pour le problème de distribution statique, qui consiste en un ensemble de vues V et un ensemble d'index I. Toutes les requêtes de départ sont donc exécutées en utilisant les deux ensembles (V et I). A l'aide d'un modèle de coût, le coût d'évaluation de l'ensemble des requêtes est calculé.

Le principe de notre approche est de *reconsidérer itérativement* cette solution initiale dans le but de réduire le plus possible le coût d'évaluation des requêtes. Elle s'appuie sur deux agents, *l'espion des index* et *l'espion des vues* dont les rôles sont les suivants:

- L'espion des index a pour tâche de voler de l'espace réservé aux vues. L'espace ainsi récupéré sera utilisé pour créer d'autres index et les vues correspondantes seront supprimées. L'opération est validée si le coût total d'exécution des requêtes diminue.
- D'une manière similaire, l'espion des vues a pour tâche de dérober de l'espace réservé aux index afin de créer d'autre vues dans le but de minimiser le coût total d'exécution des requêtes.

Notre algorithme organise donc une sorte de "combat" entre ces deux espions.

L'algorithme s'achève lorsque les deux espions ne peuvent plus réduire le coût total d'exécution des requêtes. A la fin de cet algorithme, nous obtenons deux résultats principaux :

1. Un ensemble de vues matérialisées $V' = \{V'_1, V'_2, ..., V'_{s'}\}$ et un ensemble d'index $I' = \{I'_1, I'_2, ..., I'_{r'}\}$

garantissant un coût minimal d'exécution des requêtes. Ces deux ensembles peuvent être différents des deux ensembles de départ.

2. De nouveaux quotas d'espace pour les vues matérialisées $S^{V'}$ et les index $S^{I'}$ tels que : $S^{V'} \neq S^{V}$ et $S^{I'} \neq S^{I}$.

3.1 Conclusion

Dans cette partie, nous avons développé une méthode de distribution de l'espace entre deux structures d'optimisation concurrentes sur la même ressource qui est l'espace de stockage. Cette approche exploite la similarité entre les deux structures. Cette répartition est basée sur une approche itérative. Elle commence par une sélection initiale des vues et des index, puis essaye d'améliorer cette dernière en utilisant deux algorithmes gloutons (espion des index et espion des vues).

4 Parallélisation des traitements

Publications : [14, 46, 47]

Au cours de la dernière décennie, la taille des entrepôts de données a augmenté de 5 à 100 Téra octets [84]. Pour optimiser les requêtes décisionnelles sur des entrepôts de données de telle taille, les structures d'optimisation classiques telles que les *vues matérialisées*, les *index avancés*, la *fragmentation* ne sont pas suffisantes. Le traitement parallèle devient alors une solution incontournable pour réduire les coûts de requêtes complexes. Ce dernier n'a pas eu la même attention de la part de la communauté des entrepôts de données contrairement aux techniques classiques; à l'exception des travaux de [200, 201] et de [95]. Dans le cadre de travaux de magister de Soumia BENKRID à l'Ecole Nationale Supérieure d'Alger soutenu en juin 2009, nous nous sommes intéressés à la parallélisation des traitements dans le contexte des entrepôts de données relationnels.

La conception d'un entrepôt de données parallèle passe par cinq phases principales: (i) le choix de l'architecture matérielle, (ii) la fragmentation de l'entrepôt de données, (iii) l'allocation (ou le placement) de fragments générés par le processus de la fragmentation, (vi) la répartition des charges et (v) le traitement des requêtes.

Trois architectures principales existent pour supporter une base de données (ou un entrepôt de données) parallèle: (1) l'architecture à mémoire partagée (shared-memory), (2) l'architecture à disques partagés (shared disk) et (3) l'architecture distribuée (shared nothing). Dans le contexte des entrepôts de données, shared nothing a été adoptée par [95] et fortement recommandée pour les entrepôts de données relationnels modélisés par des schémas en étoile par [84].

Une fois l'architecture matérielle choisie, le concepteur fragmente son entrepôt de données en un ensemble de partitions. La fragmentation est une pré-condition dans la conception des entrepôts parallèles. Cette fragmentation peut être horizontale (selon ses instances) ou verticale (selon ses attributs). La fragmentation horizontale est souvent utilisée pour concevoir des bases de données parallèles [200] et [95].

Le placement des données est le processus affectant les fragments générés par la fragmentation sur les noeuds d'une machine parallèle. L'allocation peut être soit redondante (les fragments sont répliqués sur les sites/noeuds) soit non redondante (chaque fragment réside dans un et un seul noeud/site).

Une fois l'allocation réalisée, les requêtes globales seront réécrites en fonction des fragments. Lors du traitement de requêtes, nous devons vérifier si les noeuds de la machine parallèle sont uniformément chargés.

En nous penchant sur la littérature sur les travaux concernant la conception des bases de données parallèles en général et les entrepôts de données parallèles, en particulier, nous constatons que les deux processus de fragmentation et d'allocation se font d'une manière indépendante et séquentielle : le concepteur partitionne d'abord son entrepôt en utilisant son algorithme favori de fragmentation (le schéma de fragmentation doit optimiser un ensemble de requêtes) ensuite il alloue les fragments sur des noeuds en utilisant un algorithme favori d'allocation (le schéma d'allocation généré doit optimiser le même ensemble de requêtes exécutées sur les différents noeuds). Cette indépendance a fait naître deux communautés de recherche; l'une travaille sur la sélection de schéma de fragmentation [169], [17], [66], [144], [158] et l'autre travaille sur le problème de placement [5], [131], [147], [149]. Cette communauté ne se

soucie pas de la génération des fragments, elle suppose leur existence. L'inconvénient majeur de cette conception est son ignorance de l'interdépendance entre la fragmentation et l'allocation. Étant donné que les fragments générés par le processus de fragmentation sont l'une des entrées du problème d'allocation et que la fragmentation et l'allocation cherchent souvent à optimiser un ensemble de requêtes, nous proposons dans cette section une approche traitant conjointement le problème de fragmentation et de placement. Un algorithme d'équilibrage de charge entre les noeuds est proposé. A notre connaissance, ce travail est le premier qui offre une solution complète de conception d'un entrepôt de données parallèles: fragmentation, allocation et équilibrage de charge.

4.1 Positionnement

Dans cette section, nous décrivons les principaux travaux sur la fragmentation et l'allocation proposés dans le cadre des entrepôts de données parallèles [95], [200] et [201].

Dans [200], les auteurs proposent une approche de construction et d'exploitation d'un entrepôt de données sur une machine parallèle de type shared disk ayant K disques. L'entrepôt considéré est modélisé par un schéma en étoile caractérisé par une table de faits volumineuse et un nombre de tables de dimension de petite taille. Ils ont proposé une approche de fragmentation qui décompose la table des faits en utilisant une méthode de partitionnement appelée *Fragmentation Hiérarchique Multidimensionnelle*. Elle consiste à fragmenter la table de faits en utilisant plusieurs attributs de tables de dimension. Chaque table de dimension est fragmentée virtuellement en utilisant le mode *intervalle* (Range partitioning) sur des attributs appartenant à des *niveaux plus bas de la hiérarchie*. Les tables dimensions et leurs index (B*-arbre) sont répliquées sur chaque disque de la machine parallèle. Pour accélérer les requêtes, des index de jointure en étoile entre les tables de dimension et la table des faits sur des attributs appartenant à des *niveaux plus hauts de la hiérarchie* sont définis.

Le processus d'allocation concerne alors les fragments de la table des faits et les index de jointure définis. Notons que le nombre de fragments générés N peut être largement supérieur au nombre de disques K. Pour soutenir un degré de parallélisme élevé et un bon équilibrage de la charge, une allocation circulaire des fragments de la table des faits sur les disques est utilisée. Les index de jointure binaires définis sur le même fragment sont placés consécutivement sur les disques afin de permettre un parallélisme intra-requêtes. Par exemple, si le fragment $frag_i$ de la table des faits est placé sur le disque j, tous les k index de jointure qui lui sont associés sont placés sur les disques $j, j+1, ..., j+k-1$ *modulo* d. Certaines directives pour les concepteurs d'entrepôts de données parallèles ont été recommandées: exclure tous les schémas de fragmentation qui ne respectent pas les contraintes suivantes: (i) le nombre des index binaires autorisés que l'administrateur souhaite matérialiser, (ii) le nombre de fragments des tables des faits générés que l'administrateur souhaite avoir, (iii) la taille de disque de chaque noeud, (vi) éviter que le nombre de fragments générés soit inférieur au nombre de disques. Malheureusement, ces recommandations n'ont pas toutes été prises en considération dans leurs algorithmes de fragmentation et d'allocation. Dans le Warlock [201], certaines recommandations, comme le nombre des fragments de la table des faits ont été prises en compte.

Dans [95]; une stratégie de placement de données consistant à partitionner horizontalement la table des faits et les grosses tables de dimension ensuite les allouer sur les noeuds en utilisant une distribution circulaire ou aléatoire est proposée. Les tables de petite taille sont répliquées sur tous les noeuds. Les

tables de dimension plus importantes sont fragmentées par hachage selon leurs clés primaires. La table des faits est à son tour partitionnée par hachage en utilisant les clés étrangères. Cette fragmentation ne prend pas en considération des exigences de requêtes de jointure en étoile ((i) des sélections sur les tables de dimension et (ii) des jointures entre la table des faits et les tables de dimension). [144] proposent une approche de fragmentation des entrepôts de données XML. Les fragments générés sont ensuite alloués sur des grilles de calcul. [140] proposent une approche de conception d'un entrepôt relationnel sur des clusters de données. Un cluster de base de données est un ensemble de machines, chacune ayant son propre SGBD, inter connectées par une couche middleware. Elles gèrent une base de données et exécutent des requêtes. Le problème traité par [140] est l'allocation des données sur les différentes machines. Deux solutions existent : répliquer totalement la base de données sur toutes les machines, ou partitionner les données sur les machines. Leur contribution consiste à combiner la fragmentation et la réplication. La table des faits est partitionnée et les tables de dimensions sont totalement répliquées. Les auteurs ne contrôlent pas le nombre de fragments générés comme dans [200].

Dans les travaux cités, les processus de partitionnement et d'allocation se font *d'une manière séquentielle (itérative)* (Figure 6.13.

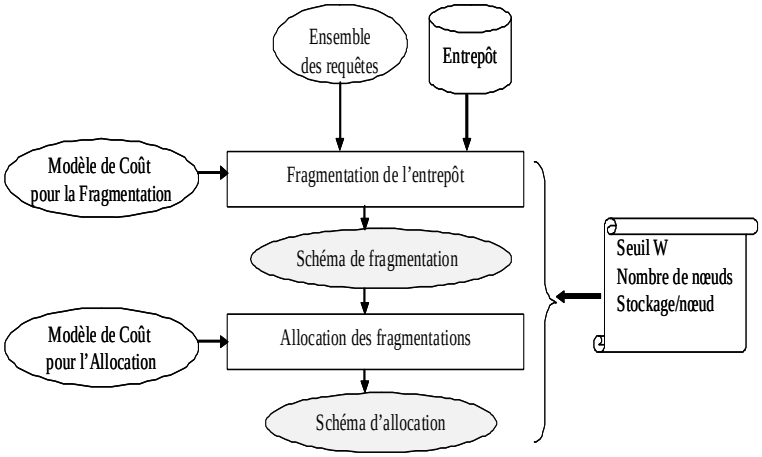


FIG. 6.13 – Les étapes de l'approche de conception itérative

Dans le cadre du travail de magister de Soumia BENKRID [14] soutenu en juin 2009, nous avons développé une approche de conception d'un entrepôt de données parallèle, où la décision d'allocation se fait pendant la fragmentation.

4.1.1 Formalisation de Notre Problème

Étant donné:

– Un entrepôt de données composé de d tables de dimension $\{D_1, ..., D_d\}$ et d'une table de faits F. Comme dans [95, 140], nous supposons que toutes les tables de dimension sont répliquées sur tous les noeuds et résident dans leurs mémoires centrales.

– Un ensemble de charge de requêtes $Q = \{Q_1, Q_2, ..., Q_n\}$ exécuté sur une machine parallèle Shared Nothing à M noeuds $N = \{N_1, N_2, ..., N_M\}$. Chaque noeud N_i $(1 \leq i \leq M)$ possède une capacité de

stockage ST_i.

– Une contrainte de maintenance W représentant le nombre de fragments de l'entrepôt que le DBA considère pertinents pour le processus d'allocation. Ce nombre doit être largement supérieur au nombre de noeuds.

Le problème de conception d'un entrepôt de données parallèle consiste à fragmenter la table des faits en N fragments et à les allouer simultanément afin de réduire le coût d'exécution de requêtes sur les noeuds de la machine parallèle. La figure 6.14 illustre les étapes de notre approche. Nous constatons l'existence d'un seul modèle de coût contrairement à l'approche itérative.

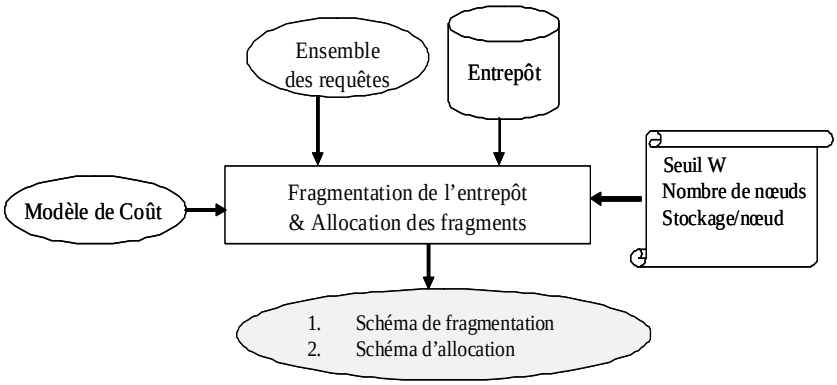


FIG. 6.14 – Les étapes de l'approche de conception conjointe

Pour répondre à ce problème, nous avons proposé un algorithme de fragmentation et d'allocation [14] Nous avons effectué des expérimentations afin de valider notre proposition. Les résultats préliminaires obtenus montrent l'intérêt de la conception multiple des entrepôts de données parallèles. Pour plus de détails voir [14].

4.2 Conclusion

Le traitement parallèle est devenu une nécessité pour répondre aux requêtes complexes. La conception d'un entrepôt de données parallèle passe par cinq étapes : le choix de l'architecture matérielle, la fragmentation, l'allocation des données, le traitement des requêtes et la répartition de la charge entre les noeuds. La majorité des travaux sur la conception des entrepôts de données parallèles traite la fragmentation et l'allocation d'une manière séquentielle. Dans cette partie, nous avons montré l'intérêt de traiter ces deux problèmes d'une manière simultanée. Ensuite nous avons proposé un algorithme composé de deux procédures, une pour fragmenter l'entrepôt de données et l'autre pour allouer les fragments générés. La décision de placement se fait lors de la fragmentation. Les algorithmes proposés sont validés en utilisant les données du banc d'essai APB1 [75]. Les résultats obtenus nous encouragent à poursuivre dans cette direction. Soumia BENKRID inscrite en thèse (la cotutelle est en cours de finalisation) à l'école nationale d'informatique d'Alger travaille sur le même thème.

Chapitre 7

Simulation de la conception physique

Lors de la sélection des schémas d'optimisation, nous avons identifié un besoin de développement d'un outil assistant les concepteurs dans la phase d'exploitation. Pour satisfaire ce besoin nous avons développé un outil, appelé ParAdmin dans le cadre de la thèse de Kamel BOUKHALFA [56].

7.1 Etude de l'existant

Afin de positionner notre outil, nous avons fait un état des lieux des outils commerciaux : Oracle SQL Tuning Advisor [77], DB2 Design Advisor [225] et Microsoft Database Tuning Advisor [2].

Oracle SQL Tuning Advisor est un outil permettant de générer des conseils pour optimiser une charge de requêtes afin d'améliorer leurs performances. Les conseils se présentent sous forme de recommandations, chacune avec le bénéfice qu'elle apporte. L'administrateur a le choix soit d'accepter les conseils soit de les compléter. Les recommandations concernent trois structures d'optimisation : les vues matérialisées, les index définis sur une seule table et la fragmentation horizontale primaire. Ces recommandations sont établies en utilisant les modèles de coût basés sur des statistiques sur la base de données. Cet outil utilise souvent une sélection isolée.

DB2 Design Advisor est une amélioration de l'outil *DB2 Index Advisor tool* [208] défini initialement pour automatiser la sélection des index. *DB2 Design advisor* permet de générer des recommandations sur quatre structures d'optimisation : les vues matérialisées, les index, la fragmentation horizontale primaire et le groupement. La fragmentation horizontale est limitée au mode de Hachage dans une architecture parallèle où plusieurs processeurs sont interconnectés à travers un réseau. Cet outil propose une sélection combinée de ces quatre techniques en exploitant l'interdépendance entre ces techniques.

L'outil *Microsoft Database Tuning Advisor* fait partie intégrante de *SQL Server 2005*. Les recommandations générées par cet outil concernent quatre techniques d'optimisation: la fragmentation horizontale primaire, la fragmentation verticale, les index et les vues matérialisées. Contrairement à l'outil d'Oracle, Database Tuning Advisor utilise l'optimiseur de requêtes pour évaluer la qualité des techniques sélectionnées et les différentes alternatives. Pour réduire le coût des appels de l'optimiseur, l'outil utilise des serveurs de tests pour estimer la qualité des différentes structures.

La conclusion que nous pouvons tirer de cette étude des outils est qu'il n'y a pas un consensus sur

les structures d'optimisation utilisées, c'est-à-dire que chaque outil génère des recommandations sur ses propres techniques d'optimisation. Même pour les techniques consensuelles, chaque outil les implémente à sa façon. Prenons l'exemple de la fragmentation horizontale primaire. Elle est implémentée sous Oracle en utilisant plusieurs modes (*range, list, hash, range-range, range-liste*, etc) tandis que sous DB2 elle est implémentée en n'utilisant que le mode *Hash*. De plus, la fragmentation horizontale dérivée n'est supportée par aucun de ces outils.

Ces outils ont été développés dans le seul souci de réduire le temps qu'un administrateur passe à tuner son entrepôt de données. Par exemple, le gain de temps pour les administrateurs est un des arguments de vente de l'outil d'Oracle. De même, dans [72], nous pouvons lire que l'outil de réglage d'Oracle10G permet aux utilisateurs d'avoir un gain de 76% du temps de réglage et d'administration passé sous SQL Server 2000. Enfin, ces outils ne donnent pas accès aux choix des algorithmes utilisés pour la sélection des structures d'optimisation.

Le tableau 7.1 montre une comparaison des structures d'optimisation supportées par chacun de ces outils.

Outil	Index	FHP	FHD	VM	Groupement
Oracle Access Advisor	Oui	Oui (tous les modes)	Non	Oui	Non
DB2 Design Advisor	Oui	Oui (Hash)	Non	Oui	Non
Database Tuning Advisor	Oui	Oui(Range)	Non	Oui	Oui

TAB. 7.1 – Comparaison des outils commerciaux

Ces outils ont été développés dans le cadre de l'auto-administration des bases de données (zéro administration par l'administrateur de la base de données). Le principal risque d'utilisation de ces outils est leur incapacité à fournir des recommandations *robustes* [98], cela est dû au fait que celles-ci étaient établies sans le contrôle et l'intervention de l'administrateur. Une recommandation non robuste peut détériorer les performances des requêtes au lieu de les augmenter. Cela peut être causé par le fait que les hypothèses prises par l'optimiseur ne sont pas valides ou le fait que la charge de requête n'est pas représentative [98]. Ces outils sont propriétaires et supposent que la base de données est opérationnelle. Ils n'offrent pas la possibilité de considérer d'autres algorithmes de sélection de schémas de structures d'optimisation et les paramétrer. Peu d'outils existent pour exploiter les similarités entre les structures d'optimisation.

7.2 Conception et réalisation de l'outil d'assistance ParAdmin

Publications : [19, 59, 18]

7.2.1 Analyse des besoins

En examinant les différents choix qu'un administrateur est amené à faire, nous avons identifié un ensemble de besoins d'administration que *ParAdmin* doit satisfaire. Ces besoins concernent les aspects suivants :

- Les structures d'optimisation supportées ;
- Les modes de sélection des techniques d'optimisation ;
- Les algorithmes de sélection et leur paramétrage ;
- La possibilité de tuner l'entrepôt.
- La possibilité des retour en arrière pour reconsidérer les choix en cas d'insatisfaction. Après la sélection des structures d'optimisation, l'administrateur peut ne pas être satisfait de ces structures. Cette insatisfaction peut être due à la qualité des structures sélectionnées ou à certaines tables ou attributs que l'administrateur préfère ne pas utiliser dans ces structures. *ParAdmin* doit pouvoir donner la possibilité à l'administrateur de revenir en arrière pour reconsidérer ses choix et améliorer la qualité des structures sélectionnées.
- La possibilité de personnalisation de l'administration. La plupart des outils proposés par les éditeurs de SGBD entrent dans le cadre de l'auto-administration (zéro administration) où l'administrateur délègue tous ses choix à l'outil. Nous avons vu que ces outils ne garantissent pas une solution robuste, vu que cette solution a été faite sans l'intervention de l'administrateur. Or, ce dernier, grâce à son expérience pourra personnaliser son administration en ciblant par exemple les attributs et tables candidats pour chaque technique d'optimisation. La personnalisation pourra toucher les algorithmes de sélection utilisés ainsi que la configuration de leurs paramètres. Par exemple, l'administrateur peut éliminer une table de dimension de petite taille du processus de fragmentation et préférer utiliser les IJB pour optimiser les requêtes accédant à cette table. Par conséquent, *ParAdmin* doit non seulement, donner la possibilité à l'administrateur d'effectuer une administration non personnalisée (par laquelle l'outil prend tous les choix à la place de l'administrateur) mais aussi lui proposer de personnaliser son administration (l'administrateur effectuera alors tous les choix que nous avons présentés plus haut).

7.2.2 Conception

Pour réaliser cet outil que nous avons baptisé ParAdmin, nous avons considéré trois structures d'optimisation : la fragmentation horizontale primaire, la fragmentation dérivée et les index de jointure binaires. Les objectifs principaux de *ParAdmin* sont :

- de permettre la visualisation de l'état courant de l'entrepôt de données : la structure de l'entrepôt (schéma, attributs, taille de chaque table, définition de chaque attribut, etc.) et la charge définie sur l'entrepôt (le nombre de sélections, le nombre de jointures, attributs de sélection, les facteurs de sélectivité de chaque prédicat, etc.)
- d'ffrir deux modes de sélection de techniques d'optimisation : *personnalisée* et *non personnalisée*. Dans la sélection non personnalisée, nous supposons que l'administrateur n'a pas assez de connaissances sur la technique sélectionnée. L'outil fait alors le choix (par défaut) de l'algorithme de sélection et propose la solution générée par cet algorithme. Tandis que la sélection personnalisée donne plus de liberté à l'administrateur pour le choix de l'algorithme, des paramètres, des

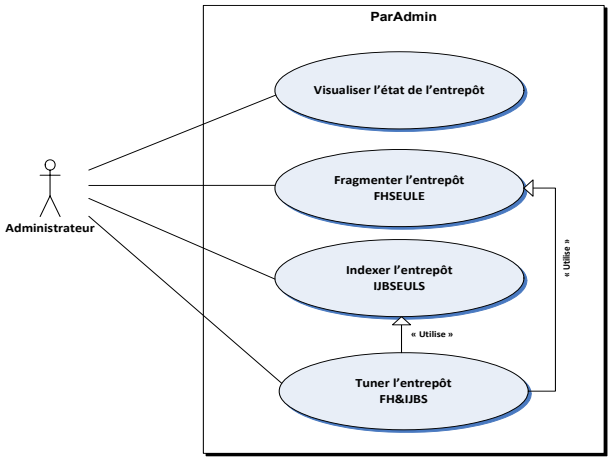


FIG. 7.1 – Le modèle des cas d'utilisation de ParAdmin

attributs et des tables sur lesquels il souhaite sélectionner la technique d'optimisation.
- d'offrir une sélection mono-structure de son choix (FHSEULE, IJBSEULS).
- d'offrir une sélection multi-structure (FH&IJB).
- de permettre à l'administrateur de revoir ses choix s'il n'est pas satisfait.
- de proposer la génération des scripts implémentant les techniques d'optimisation.
- de permettre la visualisation de la qualité de chaque technique d'optimisation proposée et de ses critères de performance : le coût d'entrées/sorties de requêtes, le taux d'amélioration apporté par la technique, etc.

A partir de ces objectifs, nous avons identifié quatre cas d'utilisation de *ParAdmin* : la visualisation de l'état de l'entrepôt, la fragmentation de l'entrepôt (FHSEULE), l'indexation de l'entrepôt (IJBSEULS) et le tuning de l'entrepôt (FH&IJBS). La figure 7.1 représente le modèle des cas d'utilisation correspondant.

7.2.2.1 Modèles de tâches pour la conception de ParAdmin

Le développement des systèmes interactifs nécessite la conception d'une interface conviviale capable de répondre aux besoins d'interactivité entre le système et l'utilisateur. La validation des interfaces est coûteuse en temps. Un point important de la validation ergonomique d'applications interactives est son acceptation par ses utilisateurs. Afin d'accroître l'acceptation d'applications interactives et diminuer le coût de validation, des études cherchent à comprendre les besoins des utilisateurs depuis l'étape de conception. Ces études proposent l'utilisation des *modèles de tâches* pour modéliser les besoins de l'utilisateur en termes d'interactivité [167]. Ces modèles sont développés en associant les utilisateurs de manière à représenter la façon dont ils effectuent des activités. Parmi les modèles de tâches existant dans la littérature, nous pouvons citer CTT (Concurent Task Trees) [174], Diane+ [207], GTA (Groupware Task Analysis) [209] et K-MAD (Kernel of Model for Activity Description) [87]. Certains modèles de tâches intègrent un outil permettant d'aider le concepteur à mettre en oeuvre un modèle de tâche. Par exemple, CTT, Diane+, GTA et K-MAD intègrent respectivement les outils CTTE, TAMOT, EUTERPE et K-MADe. Ces outils intègrent le plus souvent un outil de simulation qui produit les scenarii d'utili-

sation. Ces derniers sont utilisés pour permettre la validation par l'utilisateur de l'ordonnancement des tâches du logiciel en conception. Ce travail a été effectué en collaboration avec Cybille CAFFIAU, une doctorante au laboratoire LISI au sein de l'équipe Interface Homme Machine.

Malgré une syntaxe différente, ces modèles utilisent les mêmes concepts de modélisation qui peuvent se résumer dans les concepts suivants :

- *le but* qui représente l'état du système à atteindre;
- *la tâche* qui représente ce qu'il y a à faire ou est fait pour atteindre un but. Il s'agit toujours d'un verbe d'action;
- *la manière d'atteindre le but*, exprimée par l'ordonnancement des tâches et leur décomposition hiérarchique;
- *les conditions* dans lesquelles la tâche est réalisée, exprimées sous forme de pré et post-conditions.
- *l'opérateur* exécutant la tâche qui peut s'agir de l'utilisateur ou du système.

Nous avons utilisé le modèle de tâches K-MAD et son outil K-MADe pour modéliser *ParAdmin*. Ce choix est lié principalement au fait que ce dernier est développé en partenariat avec l'équipe IHM de notre laboratoire et l'INRIA. De plus, il permet la définition de conditions logiques (pré, post, itération) formelles prises en compte lors de la simulation augmentant de ce fait le niveau de vérification.

7.2.2.2 Architecture fonctionnelle de ParAdmin

La modélisation des tâches nous a permis d'identifier les différents besoins d'utilisation de *ParAdmin*. Nous avons validé l'ordonnancement des tâches à l'aide des scenarii afin de produire une interface permettant de les accomplir selon les besoins de l'administrateur. K-MADe permet de générer les scenarii en fonction des tâches modélisées, de leurs natures (optionnelles, obligatoire, etc.) et de leurs exécutions (séquentielles, parallèles, etc.). K-MADe donne la possibilité à l'utilisateur d'exécuter ou non les tâches optionnelles (la personnalisation de l'administration par exemple) et d'instancier certaines variables utilisées dans la suite du scénario (par exemple l'algorithme choisi algorithme génétique, recuit simulé, hill climbing). La gestion des post et pré-conditions est très importante dans le déroulement des scenarii. Par exemple, si l'administrateur choisit l'administration personnalisée, alors une variable booléenne ($Perso$) est instanciée automatiquement à *vrai* (post-condition). Cette instanciation permet d'exécuter ou non certaines tâches dépendantes de cette variable. Par exemple, la tâche *choix des attributs* est exécutée seulement si l'administration personnalisée a été choisie, elle est donc gardée par la pré-condition $Perso = Vrai$. Par conséquent, si l'administration personnalisée n'a pas été choisie (*Perso=Faux*), alors l'outil n'affiche pas la tâche *Choix des attributs* parmi les tâches disponibles. La génération des scenarii pourra nous aider à identifier les tâches qui doivent être disponibles dans *ParAdmin* à n'importe quel moment en fonction des choix effectués par le concepteur ou l'administrateur. Par exemple dans le cas précédant, lorsque le concepteur choisit une administration non personnalisée, *ParAdmin* doit automatiquement désactiver la possibilité de choisir les attributs et les tables candidates.

L'utilisation du modèle de tâches K-MAD nous a permis d'obtenir l'architecture globale du fonctionnement de *ParAdmin* (voir la Figure 7.2). Cette figure montre que l'administrateur commence par visualiser l'état actuel de l'entrepôt selon trois types d'informations : informations sur l'entrepôt, sur les requêtes et sur le système physique. Les informations concernant l'entrepôt peuvent être obtenues automatiquement en accédant aux catalogues de l'entrepôt. Après la visualisation, l'administrateur peut

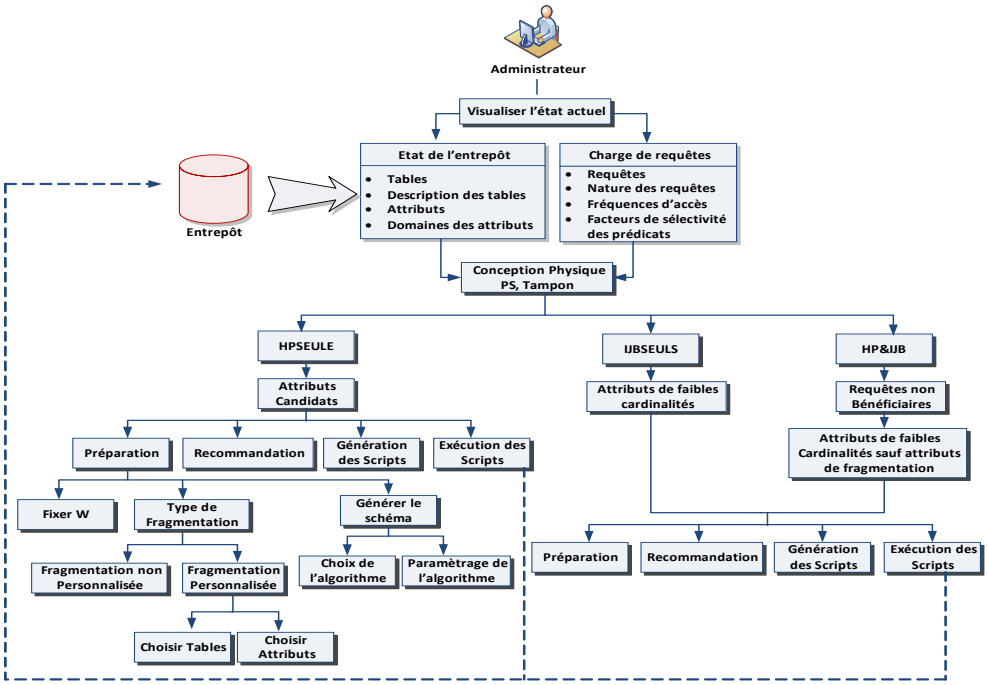


FIG. 7.2 – Architecture générale du fonctionnement de *ParAdmin*

fragmenter l'entrepôt, l'indexer ou les deux en même temps. Par exemple, pour fragmenter l'entrepôt, l'administrateur passe par quatre étapes, la préparation de la fragmentation, la génération des recommandations, la génération des scripts et l'exécution des scripts. La préparation de la fragmentation consiste à choisir le seuil *W* (représentant le nombre final de fragments de la table des faits) et le type d'administration (personnalisée ou non). Si la fragmentation personnalisée a été choisie, l'administrateur peut choisir les attributs, les tables ainsi que l'algorithme de sélection et éventuellement les valeurs des paramètres de cet algorithme. *ParAdmin* recommande un schéma de fragmentation et affiche toutes les informations concernant ce schéma. Si l'administrateur est satisfait, il lance la génération des scripts qu'il pourra exécuter pour fragmenter physiquement l'entrepôt.

Cet outil a été développé sous l'environnement Visual Studio 2005 et validé par des tests effectués par les étudiants du laboratoire. Figure 7.3 montre l'interface globale de l'outil.

L'outil *ParAdmin* que nous proposons permet d'aider l'administrateur dans sa tâche de conception physique et de réglage. Il lui permet de choisir les techniques d'optimisation, le mode de leur sélection, les algorithmes utilisés, les paramètres relatifs à chaque algorithme ainsi que les tables et les attributs pris en compte pour la génération des recommandations. Contrairement aux autres outils, *ParAdmin* permet de recommander à la fois la fragmentation primaire et dérivée. *ParAdmin* permet aussi de combiner la fragmentation horizontale et les index de jointure binaire selon une approche qui permet d'utiliser la fragmentation horizontale pour élaguer l'espace de recherche des index de jointure binaire et par conséquent réduire sa complexité. Notre outil peut être utilisé pour optimiser n'importe quelle base de données ou entrepôt de données. Cela est possible grâce à *Statistics API* développée dans [134]. Cette API permet d'accéder aux métadonnées d'une base de données et de récupérer les différentes

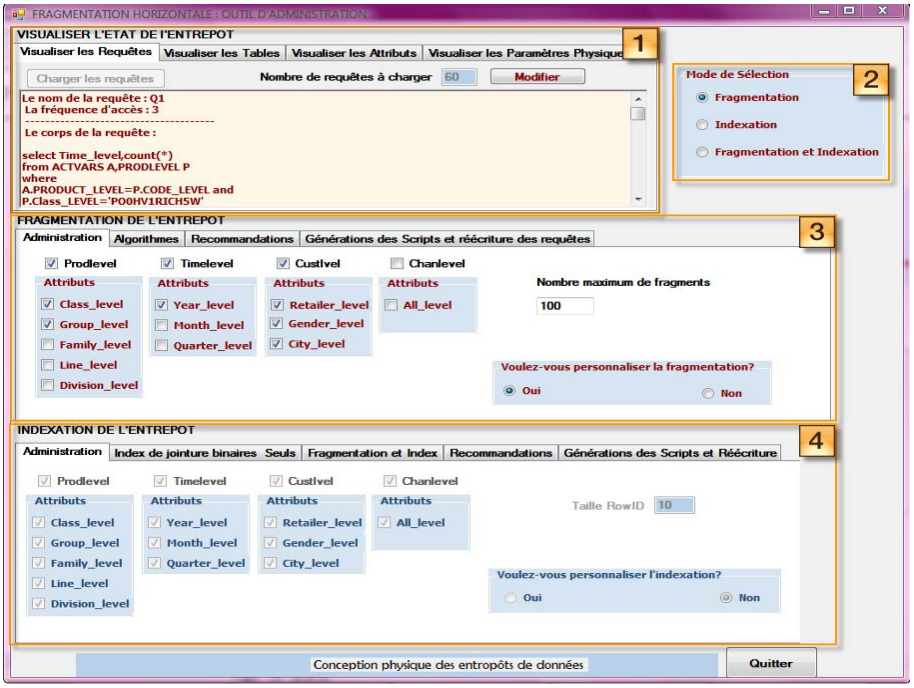


FIG. 7.3 – Interface globale de l'outil ParAdmin

statistiques : cardinalités des tables, tailles moyenne des n-uplets, facteurs de sélectivité des prédicats, etc. Les statistiques retournées par l'API sont utilisées par notre modèle de coût pour estimer la qualité des différentes techniques d'optimisation sélectionnées.

7.3 Conclusion

Les entrepôts de données ont fait naître un nouveau besoin d'administration et de tuning. Cela est dû à leurs caractéristiques : la volumétrie, la complexité des requêtes, les exigences de temps de réponse raisonnable et la gestion de l'évolution de l'entrepôt. Dans cet environnent, nous avons mis en évidence les difficultés qu'un administrateur peut rencontrer durant les phases de conception physique et de tuning. Ces difficultés sont multiples, car elles concernent plusieurs niveaux de conception: le choix des techniques d'optimisation pertinentes pour l'ensemble de requêtes à optimiser, le choix de la nature de sélection des techniques d'optimisation et le choix des algorithmes et leurs paramètres. Vu ces difficultés, nous avons identifié le besoin de développer un outil d'assistance à l'administrateur permettant de répondre aux besoins en termes de choix possibles. Nous avons proposé un outil, appelé *ParAdmin*, offrant trois techniques d'optimisation : la fragmentation horizontale primaire, la fragmentation horizontale dérivée et les index de jointure binaire. Il permet à l'administrateur de choisir les différents algorithmes et leurs paramètres. Il peut alors utiliser ces techniques d'une manière isolée ou combinée. Une autre particularité de *ParAdmin* est de proposer des sélections personnalisées et non personnalisées des structures d'optimisation. Après chaque sélection, *ParAdmin* propose des recommandations à l'administrateur permettant de visualiser la qualité de la (des) technique(s) qu'il a choisie(s). Il peut alors soit valider ses

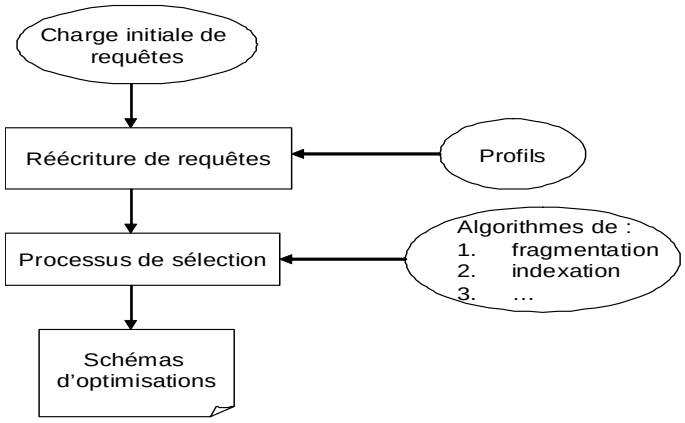


FIG. 7.4 – Sélection personnalisée des structures d'optimisation

choix en générant et exécutant les scripts ou les revoir et en faire d'autres.

7.4 Personnalisation d'un système d'intégration

Publications : [24, 154, 23].

Nous avons effectué nos premiers travaux sur la personnalisation dans le cadre d'un entrepôt de données décisionnel, modélisé par un cube de données. Ces travaux ont été effectués dans le cadre de la thèse de Hassina MOULOUDI [153] encadrée par Arnaud Giacometti et Patrick Marcel du laboratoire d'informatique de l'université de Tours.

Nos travaux de personnalisation consistent à offrir aux décideurs des éléments importants contenus dans un cube de données et qui, d'une part, doivent correspondre à une requête spécifique et aux choix d'affichage souhaité, et d'autre part, prendre en compte les contraintes liées au support de visualisation (écran, téléphones portables, pda, etc.) vue le développement des supports mobiles. Cependant lorsque les résultats de requêtes à afficher sont très importants, il n'est parfois plus possible de tout afficher et il faut alors rechercher des résultats correspondant le mieux aux souhaits du décideur tout en considérant les contraintes matérielles. Pour réduire la taille des résultats, une piste consiste à incorporer les profils des utilisateurs dans les requêtes. Pour aborder ce travail, nous avons effectué un état de l'art sur les travaux de personnalisation dans le contexte des bases de données relationnelles et la recherche d'information.

La première proposition liée à ce travail était l'exploitation des profils des utilisateurs pour partitionner un entrepôt de données, car un profil peut être exprimé sous forme d'un prédicat de sélection (Figure 7.4).

La deuxième proposition consiste en une formalisation du problème de personnalisation comme étant un problème d'optimisation avec contraintes. Il s'agit de trouver une visualisation intéressante pour l'utilisateur tout en satisfaisant un ensemble de contraintes. Les travaux de ont été validés sur une plateforme mobile, appelé MobileOlap simulant des requêtes sur un téléphone mobile. Les premières expériences menées sur le banc d'essai APB-1 [75].

Troisième partie

Perspectives

Chapitre 8

Conclusion & perspectives

8.1 Bilan général

Les travaux présentés dans ce mémoire ont été menés chronologiquement au sein de quatre laboratoires de recherche : TIMC-IMAG - Grenoble, LIMOS - Clermont Ferrand, Database and Knowledge Management laboratory of Hong Kong University of Science and Technology et le laboratoire LISI/ENSMA Poitiers. Ces travaux portent sur trois domaines complémentaires : la gestion sémantique de données hétérogènes effectués pleinement au laboratoire LISI/ENSMA, la conception physique des bases de données volumineuses effectués dans les quatre laboratoires et la personnalisation de requêtes effectuée au LISI/ENSMA avec une collaboration avec le Laboratoire d'Informatique de Tours.

Dans le domaine de la gestion sémantique de données hétérogènes, nous avons proposé des solutions pour gérer et rendre persistantes les données à base ontologique. Cette persistance est assurée par le modèle OntoDB développé dans le cadre de la thèse d'Hondjack DEHAINSALA dirigée conjointement par Guy PIERRA et moi même. Ce modèle est constitué de quatres parties. (1) La partie *données* permet de représenter les données à base ontologique. Les données de chaque classe définie au niveau de l'ontologie, y sont représentées dans une relation. Cette relation peut être une simple table, ou une vue constituée de plusieurs tables. Chaque attribut de cette relation correspond à une propriété définie, au niveau ontologique, comme applicable à la classe correspondant à la relation. (2) La partie *méta-base (catalogue)* est une partie classique des bases de données. Elle permet de représenter la structure globale de la base de données. (3) La partie *ontologie* permet de représenter les ontologies sous forme de données d'un méta-modèle. En effet, toutes les descriptions usuelles des ontologies sont réalisées par l'intermédiaire de méta-modèles. (4) La partie méta-schéma spécifique à notre architecture. Elle permet de représenter le méta-modèle d'ontologie utilisé dans la base de données sous forme d'instance d'un méta-méta-modèle. Elle vise à permettre que notre architecture supporte les changements ou les évolutions du formalisme d'ontologie. Notre méta-schéma a la particularité d'être réflexif, c'est-à-dire qu'il s'auto-représente dans la base de données. Cette architecture à 3 niveaux est analogue à l'architecture du MOF où la partie *données* représente des données (M_0) dans une structure de niveau M_1 soit M_1/M_0. La partie ontologie représente le niveau M_2/M_1. Enfin la partie méta-schéma représente les niveaux M_3/M_2 et M_3/M_3.

Autour de cette architecture, un travail sur l'intégration des sources de données hétérogènes s'est greffé dans le cadre de la thèse de Dung NGUYEN XUAN dirigée conjointement par Guy PIERRA et

moi même. Nous avons proposé une nouvelle approche d'intégration de sources de données autonomes et évolutives par articulation a priori d'ontologies. Elle suppose que chaque source de données possède sa propre ontologie qui référence une ontologie de domaine conceptuelle partagée. Un travail de formalisation des sources de données à base ontologique et du problème d'intégration a été mené. Plusieurs scénarii d'intégration ont été proposés. Ces propositions ont été validées sur des ontologies réelles issues du domaine de l'ingénierie automobile. Notre approche d'intégration de données permet la gestion des évolutions des sources et de leurs ontologies. Cette gestion est assurée par le principe de *continuité ontologique*. Il stipule qu'une ontologie ne peut infirmer un axiome qui se trouvait vérifié dans une version antérieure de l'ontologie. Ce principe nous a permis de proposer un mécanisme, dit de *version flottante*, qui permet de ne conserver dans le système d'intégration matérialisé qu'une seule version de chaque classe et chaque propriété, appelée version courante (en fait, la plus grande version connue). Ceci permet de consolider entre les versions courantes toutes les relations ayant existé entre les différentes versions des différents concepts. Nous avons également proposé une approche pour l'historisation des instances figurant dans les différentes sources. Afin d'identifier les instances ontologiques bien qu'elles puissent, au cours du temps, être décrites par des ensembles différents de propriétés, nous avons proposé la notion de *clé sémantique*. La présence de cette clé, permet d'éviter toute duplication dans le système d'intégration.

Il existe des cas où notre approche *a priori* ne peut s'appliquer. C'est le cas s'il n'existe pas d'ontologie partagée au moment où certaines sources sont créées. C'est également le cas s'il existe plusieurs ontologies qui se chevauchent. Enfin, cette approche d'intégration est impossible si les correspondances entre ontologies locales et partagées sont plus complexes que celles supportées dans la méthode d'articulation *a priori*. Nous avons proposé, dans ce dernier cas une approche d'intégration qui consiste à représenter sous forme de modèle, au sein de chaque BDBO, les correspondances entre l'ontologie locale et une ou plusieurs ontologies partagées. Ces modèles sont construits par réification des opérateurs algébriques de transformation. Nous avons montré que cette intégration automatique restait possible dans une approche entrepôt de données.

Ces travaux sur la gestion sémantique des données ont été à l'origine de beaucoup d'autres travaux dont, en particulier, les deux thèses menées en collaboration avec les chercheurs du LISI. Ils nous (Guy PIERRA, Yamine AIT AMEUR et moi-même) ont également motivé pour lancer un *numéro spécial* sur les ontologies et leurs contributions dans la conception des systèmes d'information avancés, dans la revue *Data & Knowledge Engineering (DKE), Elsevier*.

Dans le domaine de la conception physique des bases de données et des entrepôts de données, nous nous sommes intéressés à l'exploitation efficace des données volumineuses en proposant des structures d'optimisation adaptées aux requêtes complexes. Nos premiers travaux sur ce domaine datent de 1996. Ils ont débuté par la proposition d'algorithmes de fragmentation horizontale et verticale dans le cadre des bases de données orientées objet réparties. Un nouveau type d'algorithme pour la fragmentation horizontale a été proposé, appelé *fragmentation dirigée par modèle de coût*. Cet algorithme permet de quantifier la qualité de schéma de fragmentation obtenu. Avec l'arrivée de la technologie d'entreposage de données, demandeuse de structures d'optimisation avancées, nous nous sommes penchés sur le développement de démarches pour la sélection des schémas d'optimisation. Nous avons identifié deux approches pour la conception de structures d'optimisation : la sélection isolée et la sélection multiple. Dans la sélection isolée, une seule structure d'optimisation est sélectionnée. Tandis que dans la sélection multiple, plusieurs sont sélectionnées pour agir simultanément. Pour la sélection isolée, nous avons étu-

dié tout particulièrement la fragmentation horizontale (dérivée et horizontale) dans les bases de données orientées objet et les entrepôts de données relationnels ainsi que les index de jointure binaire. La raison de cet intérêt particulier réside dans les caractéristiques spécifiques qu'offrent les deux structures : la fragmentation horizontale ne demande ni coût de stockage ni coût de mise à jour et les index de jointure binaires sont adaptés à la compression. Ces deux structures ont d'ailleurs été largement adoptées par les éditeurs commerciaux de bases de données. Récemment, Oracle a mis en évidence la compression de données (souvent appliquée aux index binaires) et le partitionnement comme un critère pour lancer sa version 11g Release2. Pour la sélection multiple, nous nous sommes intéressés à la sélection simultanée de schémas de fragmentation et d'indexation en exploitant leurs similarités. Nos travaux ont montré que la fragmentation horizontale peut remplacer les index de jointure binaire tout en garantissant une meilleure optimisation. La fragmentation horizontale peut être utilisée pour tuner un entrepôt de données afin de prendre en considération les évolutions de son schéma et de ses instances.

Ce deuxième axe de travail a également initié de nombreux travaux, réalisés à travers d'autres collaborations nationales et internationales. On peut citer la sélection des index de jointure binaire en utilisant les techniques de fouille de données avec Prof. Rokia MISSAOUI de l'Université de Québec à Outaouais, Canada ; la proposition d'une démarche de fragmentation pour les entrepôts de données relationnels avec Karlapalem KAMALAKAR de Hong Kong University of Science and Technology et Mukesh MOHANIA d'IBM - Inde ; la fragmentation horizontale dérivée dans les bases de données orientées objet avec Prof. Qing LI de City University of Hong Kong, Chine et l'étude de l'effet de la personnalisation sur la sélection des structures d'optimisation avec Arnaud GIACOMETTI et Patrick MARCEL de l'Université de Tours et Dominique LAURENT de l'Université de Cergy-Pontoise.

Récemment, nous nous sommes penchés sur le problème de conception des entrepôts de données parallèles. Etant donné que nous avons déjà effectué des travaux sur la fragmentation et l'allocation d'une manière itérative (lors de mon séjour à Hong Kong) [27], nous avons proposé dans le cadre du Magister de Soumia BENKRID (soutenu en juin 2009) une approche qui combine les deux structures. L'intérêt pour cette problématique est né lors de l'édition d'un numéro spécial en avril 2009 dans *Distributed and Parallel Database Journal - Springer* sur le thème : "*New Trends in Physical Data Warehouse Design*". Nous avons en effet reçu un nombre important d'articles en relation avec le parallélisme.

Pour toutes les contributions que nous avons proposées, nous nous sommes toujours efforcés d'effectuer une double validation : théorique en utilisant des modèles de coût mathématiques pour la sélection des structures d'optimisation (la fragmentation, l'indexation, allocation, etc.) que nous avons développés, et réelle en utilisant le SGBD Oracle 10.

Ayant étudié plusieurs structures d'optimisation et ayant proposé un nombre important d'algorithmes pour leurs sélection, il nous est apparu un besoin d'outils d'aide à l'administration destinés à être utilisés avant le déploiement des systèmes d'intégration. Un tel outil, qui intègre plusieurs de nos contributions a été développé dans le cadre de la thèse de Kamel BOUKHALFA effectuée au LISI sous mon entière responsabilité.

8.2 Perspectives

Nos travaux ont ouvert de nombreuses pistes que nous nous proposons de développer. Dans cette section nous présentons succinctement celles qui nous paraissent être les plus prometteuses. Elles sont divisées en deux catégories : des perspectives à court terme concrétisées par des travaux en cours et des perspectives à moyen ou long terme.

8.2.1 Travaux en cours

Nos travaux en cours concernent les trois phases de notre démarche de développement de système d'intégration : la construction, l'exploitation et la personnalisation.

8.2.1.1 Conception : les sources de données à base ontologique

Le travail réalisé dans le cadre de la thèse de Hondjack DEHAINSALA est un travail fondateur sur la faisabilité de création de BDBO efficace et de grande taille. L'exploitation intensive de la BDBO OntoDB au sein de nombreux projets de recherche, incluant en particulier : le projet ANR Dafoe Differential and Formal Ontology Editor (http://dafoe4app.fr/) et le projet ANR RNTL Environmental Web Ontology Knowledge Hub (E-wok hub), ont permis d'identifier de nombreuses voies à explorer et d'améliorations à étudier. Celles-ci peuvent être classées selon trois catégories : les optimisations, les évolutions et les applications.

1. *Optimisations*. Les tests de performance effectués sur la partie *ontologie*, ont mis en évidence des temps de réponse élevés pour les ontologies contenant de nombreuses classes et propriétés. Ces temps de réponse s'expliquent par la complexité du schéma logique de la partie *ontologie* directement définie à partir du modèle d'ontologie PLIB qui comporte plus de 200 entités. Le modèle logique est donc composé de nombreuses tables, en majorité des tables d'aiguillages (supportant le polymorphisme), impliquant, pour certains accès, de nombreuses jointures. Pour améliorer la performance de notre prototype, une nouvelle approche de représentation des ontologies est actuellement en cours de développement. Elle consiste à utiliser les techniques de transformation de modèles afin de simplifier le modèle d'ontologie PLIB avant de le traduire en modèle relationnel. Cette simplification permet d'aplatir les hiérarchies d'entités et/ou de fusionner des entités liées par des relations de composition ou des agrégations en exploitant certaines contraintes d'intégrité du modèle. Ces transformations devraient améliorer considérablement les performances de la partie *ontologie* de notre architecture.
2. *Évolutions*. Notre premier prototype de BDBO est orienté vers un modèle d'ontologie unique (PLIB). Or les types de connaissances représentables dans une BDBO dépendent des caractéristiques du modèle d'ontologie utilisé. Dans nos exploitations d'OntoDB dans différents domaines (l'ingénierie, la médecine, la conception d'ontologies), nous nous sommes aperçus qu'il existait de très nombreuses constructions ontologiques possibles, et que chaque application avait ses besoins propres. D'où le besoin de ne pas supporter un formalisme unique (par exemple, PLIB ou OWL) mais un modèle flexible qui pourrait être étendu au gré des besoins de chaque application. Ceci

demande de concevoir différemment les BDBO pour rendre leur modèle plus flexible et d'identifier les primitives ontologiques hétérogènes qu'il pourrait s'avérer intéressant de rapprocher dans un même système. Les primitives que nous avons pour l'instant identifiées sont les suivantes :

– les dépendances fonctionnelles entre les propriétés définies dans chaque classe. Ceci permettrait par la suite de normaliser les schémas logiques des différentes classes ;
– les dépendances algébriques entre valeurs de propriétés. Il s'agirait d'exprimer les relations mathématiques existantes entre les propriétés des classes ce qui permettrait de calculer les valeurs de certaines propriétés dynamiquement à partir de la valeur d'autres propriétés ;
– Les opérateurs d'équivalence entre classes. Il s'agirait d'introduire en particulier certains des constructeurs des logiques de description s'appliquant sur les classes, tels que les constructeurs de restriction de classes. Ces constructeurs de classes impliquent la définition des mécanismes pour le calcul des instances des classes définies par ces constructions. Nous envisageons de réaliser ceci par l'intermédiaire de vues SQL.

Soulignons que ces deux dernières extensions consistent à passer d'ontologies conceptuelles canoniques à des ontologies conceptuelles non canoniques. Une partie du traitement des requêtes sera alors effectuée de façon intentionnelle, au niveau ontologique, avant d'être traitée de façon extensionnelle au niveau de la partie *données*. Les travaux visant à traiter les problèmes d'optimisation et de flexibilité du modèle sont actuellement très avancés à travers la thèse de Chimène KANKAM encadrée conjointement avec Guy PIERRA.

Concernant les rapports entre niveau conceptuel et logique au sein de la base de données, un grand nombre de contraintes d'intégrités, dont les contraintes d'unicité et de cardinalité, ainsi que les dépendances fonctionnelles sont souvent de nature ontologique. Nous envisageons d'exploiter automatiquement toutes les contraintes définies au niveau ontologique pour générer les contraintes d'intégrités au niveau de la base de données. Ces travaux entrent dans le cadre de la thèse de Chedlia CHEKROUN (actuellement en première année de thèse) que j'encadre conjointement avec Prof. Yamine AIT AMEUR.

3. *Applications*. L'architecture OntoDB a été largement utilisée dans plusieurs projets ANR, comme le projet EWOK-HB, et industriels (Airbus, Renault). Récemment, elle a été sélectionnée pour être utilisée dans le développement d'une plateforme technique pour concevoir des ontologies à partir de textes dans le cadre du projet ANR Dafoe (Differential and Formal Ontology Editor : http://dafoe4app.fr/). Henry Valéry TEGUIAK doctorant en deuxième année de thèse CIFRE est en charge de cette étude. Il est encadré conjointement par Yamine AIT AMEUR et moi-même.

8.2.1.2 Exploitation efficace d'un système d'intégration de données

En ce qui concerne l'exploitation efficace d'un système d'intégration, nos travaux portent sur quatre volets : (1) la sélection personnalisée des structures d'optimisation, (2) le développement de nouveaux algorithmes de sélection de schémas d'optimisation, (3) la prise en compte de l'évolution des entrepôts de données dans la sélection des schémas d'optimisation et (4) l'extension de nos travaux à de nouvelles architectures.

1. *La sélection personnalisée des structures d'optimisation.* Nous travaillons actuellement sur une sélection personnalisée des structures d'optimisation dans les entrepôts de données. Notons que

nos approches pour la sélection de schémas de fragmentation et d'indexation suivent une *démarche globale*, c'est-à-dire que leurs algorithmes considèrent tous les attributs de sélection de la charge de requêtes dans les processus de sélection. Dans une telle démarche, toutes les tables de dimension sont supposées avoir la même probabilité d'être utilisées, ce qui n'est pas toujours le cas dans la réalité, car les tables ne possèdent pas la même taille, la même fréquence d'utilisation, etc. Il serait donc intéressant d'offrir aux administrateurs/concepteurs la possibilité de choisir d'abord les tables qu'ils considèrent pertinentes pour la sélection, avant de choisir leurs attributs. Le choix des tables de dimension peut se faire manuellement par les administrateurs en se basant sur leur expérience, ou automatiquement en prenant en considération des paramètres liés à l'optimisation de la jointure. Récemment, nous avons proposé quelques pistes de recherche pour faciliter la sélection des tables de dimension pour fragmenter une table des faits dans *ACM Twelfth International Workshop on Data Warehousing and OLAP* [43]. Lors de l'étude de ce problème, nous avons identifié une similarité entre le problème de sélection des tables de dimension pertinentes pour la fragmentation de la table des faits et le problème de l'ordre de la jointure connu comme un problème difficile.

2. *Le développement de nouveaux algorithmes de sélection de schémas d'optimisation.* Actuellement, dans le cadre d'une collaboration entre le Laboratoire d'Informatique de Tours (équipe *Ordonnancement et Conduite*) et le laboratoire LISI (financement des Conseils Scientifiques de l'ENSMA et l'Université de Tours), nous explorons d'autres types d'algorithmes issus de la recherche opérationnelle (comme la programmation linéaire) pour sélectionner des schémas d'optimisation.
3. *La prise en compte de l'évolution d'un entrepôt de données dans la sélection des schémas d'optimisation.* Lors de mon séjour cet été au *laboratoire de Recherche sur l'Information Multimédia* de l'Université de Québec à Outaouais, Canada, Rokia MISSAOUI, Léonard KWUIDA (actuellement à *Zurich University of Applied Sciences*, Suisse) et moi même, nous avons identifié le besoin de développer de nouveaux algorithmes de sélection des index de jointure qui prennent en compte l'évolution d'un 'entrepôt de données. Dans notre première collaboration, nous avions proposé des algorithmes basés sur les techniques de fouille de données pour sélectionner des schémas d'indexation dans le *cas statique*. Pour prendre en compte l'*aspect évolutif d'un entrepôt de données*, une piste directe consiste à faire adapter les algorithmes incrémentaux de fouille de données pour la sélection des index.
4. *Vers d'autres environnements architecturaux.* Les environnements architecturaux ont une incidence sur les deux phases de construction d'un système d'intégration : la conception et l'exploitation.
   - Dans la thèse de Dung NGUYEN XUAN, nous avons proposé des approches d'intégration dans une architecture d'entrepôt. Il serait intéressant d'adapter nos solutions dans une architecture de médiation, en mettant en évidence les différentes composantes du médiateur, le traitement de requêtes (réécriture de requêtes, localisation de sources pertinentes, etc.), les structures d'optimisation de requêtes et le passage à l'échelle. Tout particulièrement, la méthode d'intégration par réification des correspondances entre ontologies semble se prêter de façon intéressante à une approche de type médiateur.
   - La majorité des solutions d'optimisation que nous avons proposées reposent sur un contexte centralisé. Les progrès technologiques de la dernière décennie ont permis l'amélioration significative de la puissance de calcul des ordinateurs. Néanmoins, cette puissance n'est souvent pas suffisante pour l'exécution de nouvelles applications très gourmandes en ressources de calcul

comme les entrepôts de données. Par conséquent, les processeurs sont combinés et reliés par un réseau de communication, formant ainsi des architectures parallèles (comme les clusters de bases de données ou les grilles de calcul). La majorité des travaux sur la parallélisation ou la répartition des données considèrent que les sites ou les machines possèdent le même support de stockage (disque) avec la même capacité. Avec l'évolution des dispositifs de stockage (les disques externes, les disques flashes, etc.), cette hypothèse doit être remise en cause. Il serait intéressant de proposer ou d'adapter des algorithmes de fragmentation et de placement de données dans cet environnement. Un autre problème sur lequel nous nous penchons (avec l'équipe Temps Réel du laboratoire LISI avec qui nous avons collaboré dans le cadre de la thèse de Kamel BOUKHALFA) est la proposition de stratégies d'ordonnancement des requêtes sur des architectures composées de ressources dynamiques et ayant de dispositifs de stockage variés. Ce travail pourrait intéresser l'équipe Ordonnancement et Conduite du laboratoire d'Informatique de Tours avec qui nous avons une collaboration en cours.

8.2.2 Perspectives à moyen terme

L'équipe Ingénierie de Données du laboratoire LISI est un acteur majeur de la recherche sur la modélisation, la transformation et la persistance des données fortement structurées provenant de la conceptualisation de domaines complexes (ingénierie automobile, avionique, etc.). Les grandes perspectives qui s'offrent à nous dans ce cadre sont de quatre ordres : (1) le développement de modèles, méthodes et outils permettant de traiter telles données de façon efficace, (2) la proposition de solutions de persistance des instances ontologiques issues de ces modèles, (3) la proposition d'une optimisation sémantique des requêtes et des applications et (4) l'accès personnalisé aux instances ontologiques.

1. *La gestion de modèles ontologiques* : Avec l'émergence d'un nombre rapidement croissant d'ontologies de domaine, les travaux que nous avons effectués sur la gestion et l'intégration de données à base ontologique suggèrent d'abord une nouvelle méthode de conception des bases de données à base ontologique. Dans cette méthode, le modèle conceptuel est défini à partir d'un sous-ensemble de l'ontologie de domaine, complété ensuite par spécialisation pour couvrir l'ensemble des exigences de l'application cible. Le développement d'outils de génie logiciel supportant cette approche, comme pour les bases de données traditionnelles avec le modèle ANSI/SPARC, devient une nécessité pour satisfaire les applications nécessitant des bases de données à base ontologique.

   Une telle approche ontologique de conception peut être utilisée pour modéliser les applications décisionnelles. Rappelons que la plupart des travaux sur la conception des entrepôts de données se concentre principalement sur les niveaux logique et physique. Générer directement, à partir d'un cahier des charges, des schémas logiques sans passer par une modélisation conceptuelle pose des problèmes de communications entre les concepteurs et les utilisateurs. L'utilisation des ontologies, récemment développées dans le domaine décisionnel (http://www.cyc.com/cyc), dans une approche ontologique permettrait d'offrir une conception complète des entrepôts de données décisionnels.

2. *La persistance des instances ontologiques* : La représentation de plus en plus fréquente de toute connaissance sous forme digitale qui est une caractéristique de notre société de communication, ne nécessite pas seulement de représenter explicitement la connaissance véhiculée par un ensemble de

données et de connaissances, elle nécessite aussi de pouvoir gérer de façon efficace des ensembles de données de plus en plus grands. De tels ensembles ne pouvant être traités en mémoire centrale, les systèmes étudiés dans ce thème devront permettre l'exploitation efficace de bases de données de grandes tailles, dont les données seront modélisées par référence à des ontologies. De nombreux efforts restent à faire notamment pour identifier les primitives ontologiques adaptées aux différents domaines d'application, et pour définir des architectures adaptées à la montée en charge et enfin pour développer des structures d'optimisation exploitant la notion d'ontologie.

3. *L'optimisation sémantique des requêtes* : deux dimensions de recherche peuvent concerner l'optimisation sémantique : une optimisation des bases de données à base ontologique et l'utilisation des ontologies dans la sélection de structures d'optimisation. Concernant la première dimension, nous n'avons pas encore étudié l'optimisation des requêtes sur les données à base ontologique de notre architecture OntoDB. Nos travaux sur la sélection de structure d'optimisation peuvent facilement être adaptés à la partie *données* de ce modèle. Par ailleurs, étant donné que l'ontologie locale de chaque source est stockée sous forme objet, les travaux sur l'optimisation de requêtes dans les bases de données orientées objet peuvent être réutilisés. Rappelons que l'optimiseur de requêtes utilise une approche dirigée par un modèle de coût (cost-based optimization) afin de répondre aux requêtes sur la partie donnée. Le développement de ce type d'optimisation serait également souhaitable pour la partie ontologie.

   En ce qui concerne la deuxième dimension de l'optimisation sémantique, les ontologies utilisées dans le processus d'intégration doivent être exploitées pour proposer des structures d'optimisation comme les schémas de fragmentation et d'indexation. A partir des besoins de l'application cible, les concepteurs peuvent indiquer les propriétés sur lesquelles ils souhaitent par exemple fragmenter ou indexer. Une génération directe d'un modèle logique, fragmenté en utilisant nos algorithmes, serait alors possible.

8.2.2.1 La gestion sémantique de la personnalisation

Le travail sur la personnalisation que nous avons effectué dans le cadre de la thèse de Hassina MOULOUDA en collaboration avec Arnaud GIACOMETTI, Patrick MARCEL et Dominique LAURENT portait sur l'exploitation de profil pour la sélection des structures d'optimisation et la proposition des résultats en respectant les contraintes de visualisation d'un dispositif d'affichage. Dans le cadre de la thèse de Dilek TAPUCU effectuée au sein du laboratoire LISI et sous la direction de Yamine AIT AMEUR, des solutions liant les ontologies aux préférences ont été proposées et validées sous l'architecture OntoDB. Nous considérons l'idée d'étendre cette solution au contexte d'entreposage de données afin de rendre persistant les profils de décideurs et les exploiter pour définir des structures d'optimisation avancées.

8.2.2.2 Prise en compte des propriétés non fonctionnelles dans notre démarche de développement de système d'intégration

Les propriétés non fonctionnelles se distinguent des propriétés fonctionnelles dans le fait qu'elles ne portent pas sur le résultat des applications qui vont être exécutées sur le système d'intégration, mais concernent d'autres critères de qualité tels que le temps de réponse, la qualité de service, la fraicheur de

données, la consommation d'énergie, les ressources nécessaires, la robustesse des structures d'optimisation, la sécurité, etc. Les propriétés non fonctionnelles jouent un rôle majeur dans la manière de concevoir les systèmes d'intégration impliquant de nombreuses sources. Chaque phase de notre démarche de développement doit prendre en compte, voire optimiser ses propres propriétés non fonctionnelles. Une première perspective consisterait à modéliser et à évaluer les propriétés non fonctionnelles pour chaque phase de notre démarche de conception de système d'intégration : la construction, l'exploitation et la personnalisation. Une autre perspective consisterait à étendre notre outil d'assistance afin de prendre en compte les propriétés non fonctionnelles (performance de requêtes, réplication de données, facilité de tuning, robustesse de la sélection de structure d'optimisation, etc.). Actuellement, avec notre modèle de coût (estimant le nombre d'entrées sorties nécessaires pour exécuter un ensemble de requêtes), notre outil permet d'évaluer la qualité des structures d'optimisation proposées, mais il devrait prendre en considération l'aspect réparti des sources, leurs capacités de stockage, la charge réseaux, etc.

Quatrième partie

Curriculum vitae

Ecole Nationale Supérieure de Mécanique et d'Aérotechnique et Université de Poitiers

Laboratoire d'Informatique Scientifique et Industrielle (E.A. 1232)

ENSMA — Téléport 2 — 1 Avenue Clément Ader — B.P. 40109 — 86961 FUTUROSCOPE CHASSENEUIL Cedex — FRANCE
Tél : (+33/0) 5 49 49 80 63 — Fax : (+33/0) 5 49 49 80 64 Web : www.lisi.ensma.fr

Résumé du rapport de synthèse en vue de l'obtention d'une

HABILITATION à DIRIGER DES RECHERCHES
Section CNU 27

Contributions à la conception et l'exploitation de systèmes d'intégration de données

Présentée par

Ladjel BELLATRECHE

Maître de Conférences

LISI/ENSMA – Université de Poitiers
Téléport 2, 1 avenue Clément Ader
B.P. 40109
86960 FUTUROSCOPE Cedex
France

bellatreche@ensma.fr

http://www.lisi.ensma.fr/members/bellatreche

Table de Matières

1. ETAT CIVIL

Nom patronymique	BELLATRECHE
Prénom	Ladjel
Date de Naissance	11 décembre 1968
Situation	Marié, 3 enfants
Adresse Personnelle	13, rue des Libellules Vouneuil Sous Biard 86580 France
Téléphone	(+33) 05 49 49 80 72 (bureau) (+33) 06 98 84 98 51 (mobile) (+33) 05 49 43 33 02 (domicile)
Fax	(+33) 05 49 49 80 64
Fonction actuelle	Maître de Conférences, titulaire, classe normale, 3ème échelon
Adresse professionnelle	Téléport 2 1, Avenue Clément Ader 86960 Futuroscope
Mél.	bellatreche@ensma.fr
URL	http://www.lisi.ensma.fr/members/bellatreche

2. TITRES UNIVERSITAIRES

Année	Diplôme	Etablissement
97 – 00	Doctorat en Informatique	Université Blaise Pascal Clermont – Ferrand II
93 – 94	Diplôme d'Etudes Approfondies (DEA) Informatique Fondamentale et Parallélisme Mention : Assez Bien	Université Paul Sabatier – Toulouse III
92-93	Maîtrise Ingénierie Mathématique	Université Paul Sabatier – Toulouse III
87 – 92	Ingénieur Informatique Option Système d'Information Mention : Bien Major de Promotion	Institut National d'Informatique de Sidi Bel Abbès, Algérie
87	Baccalauréat Série Mathématiques et Physique Mention : Assez Bien	Lycée Ain Tedelès Algérie

97 – 00 : Doctorat en Informatique
Université Blaise Pascal, Clermont-Ferrand II, France
Titre : *Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique d'un entrepôt de données.*

Soutenue : le 18 décembre 2000 à Clermont Ferrand, devant le jury composé de :
Président : M. Djamel ZIGHED, Professeur à l'Université Lyon 2
Rapporteurs : M. Henri BRIAND, Professeur à l'Université de Nantes
M. Jacques KOULOUMDJIAN, Professeur à l'INSA de Lyon
Directeur de thèse : M. Michel SCHNEIDER, Professeur à l'Université Blaise Pascal, Clermont-Ferrand II
Co-directeur de thèse : M. Michel SIMONET, Chargé de Recherche CNRS
Université Joseph Fourier, Grenoble
Mention : Très honorable

93 – 94 : D.E.A. Informatique Fondamentale et Parallélisme
Université Paul Sabatier, Toulouse III
Titre de mémoire de D.E.A. : *Contribution des codes correcteurs d'erreurs à la transmission de la parole*
Lieu : Institut de Recherche en Informatique de Toulouse (I.R.I.T.)
Directeur : Professeur Alain POLI

87 –92 : Ingénieur en Informatique
Option **:** Systèmes d'Informations
Lieu : Institut National d'Informatique de Sidi Bel Abbès - Algérie
Titre du mémoire d'ingénieur : *Conception et mise en place d'un système d'informations pour la gestion des activités portuaires.*

3. EXPERIENCES PROFESSIONNELLES

Sept-02 – Présent	Maître de Conférences à l'Université de Poitiers. Membre du Laboratoire d'Informatique Scientifique et Industrielle (LISI) de l'Ecole Nationale Supérieure de Mécanique et d'Aérotechnique (E.N.S.M.A.) Equipe d'Accueil (EA-1232)
Sept-01 – Août-02	Professeur Chercheur (Contrat à Durée Indéterminée) Ecole d'Ingénieur en Informatique et Robotique (IMERIR) Perpignan – France
Juin 01 – Août 01	Chercheur invité Departement of Computer Science Purdue University U.S.A.
Sept-00 – Août-01	A.T.E.R. (Attaché Temporaire d'Enseignement et de Recherche) Université Joseph Fourier – Grenoble
Sept-99 – Août-00 :	Demi A.T.E.R. - Université de Savoie – Chambéry
Fév-97 – Sept-99 :	Chercheur - doctorant Département d'Informatique University of Science and Technology – Hong Kong - CHINA
94 – 96	Ingénieur de Recherche & Développement Laboratoire Traitement de l'Imagerie, de la Modélisation et de la Cognition (TIMC-IMAG) Université Joseph Fourrier – Grenoble
93 – 94	Enseignant Vacataire Université de Paul Sabatier – Toulouse

4. PRIMES

- 2003 - 2007 : Prime d'Encadrement Doctoral et de Recherche (PEDR) (1ère demande)

- 2008 – 2011 : Renouvellement de la PEDR.

5. ACTIVITES DE RECHERCHE

5.1. Mon parcours

- **Juin 1992** : ingénieur en Informatique option «système d'informations» à l'Institut d'Informatique de Sidi Bel Abbès (Algérie).
- **1993** : Maîtrise Ingénierie Mathématique à l'Université Paul Sabatier - Toulouse
- **1994** : D.E.A. en Informatique Fondamentale et Parallélisme à l'Université Paul Sabatier Toulouse. Je n'ai pas pu continuer en thèse faute de financement.
- **1994-1996** : ingénieur de recherche au Laboratoire Traitement de l'Imagerie, de la Modélisation et de la Cognition (TIMC-IMAG - Grenoble) sous la direction de Michel SIMONET.

  J'ai alors développé mes premiers travaux de recherche sur le partitionnement dans les bases de données orientées objet.
- **1997-2000** : inscrit en thèse sous la direction de Michel SCHNEIDER du Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS) de l'université Blaise Pascal – Clermont Ferrand.

  Ces travaux de thèse entrent dans le cadre d'une collaboration avec Prof. Kamalakar KARLAPALEM de Hong Kong University of Science and Technology (HKUST). Durant cette période, j'ai effectué un séjour de deux ans à HKUST, où j'ai participé à trois projets traitant de : (i) la conception des bases de données orientées objet réparties, (ii) bases de données réparties et (iii) entrepôts de données.
- **2000-2001** : ATER à l'université de Joseph Fourier Grenoble.

  Mes travaux de recherche ont été effectués au Laboratoire TIMC-IMAG.
- **2001-2002** : Professeur à l'Ecole d'Ingénieur en Informatique et Robotique (IMERIR)
- **2002-Présent** : Maître de Conférences à l'Université de Poitiers.

  J'ai intégré l'équipe « Ingénierie des Données » (IDD) du Laboratoire LISI. Les thèmes scientifiques développés au sein de cette équipe étaient complémentaires réalisés avant mon recrutement. Cela m'a permis dès la première année de co-encadrer conjointement avec Prof. PIERRA une thèse de doctorat alliant l'intégration à base ontologique (compétence développée au sein de l'équipe IDD) et les entrepôts de données (compétences que j'ai apportées). Ces travaux en équipe ont facilité mon intégration et ont engendré d'autres travaux comme la *gestion de la persistance des données à base ontologique*, *l'étude des performances dans les bases de données à base ontologique*, *l'exploitation des données à base ontologique*, la *conception des bases de données à base ontologique*, etc. En parallèle à ces travaux, je poursuis les travaux sur l'optimisation physique des bases de données entamés pendant ma thèse de doctorat.

5.2. Travaux effectués au sein du TIMC-IMAG en qualité d'ingénieur

Mes premiers travaux sur la fragmentation verticale et horizontale dans les bases de données orientées objet ont été initiés au laboratoire TIMC-IMAG. Ce travail entre dans le cadre du projet OSIRIS. OSIRIS est un système de représentation et de gestion de données et de connaissances (SGBD-BC) orienté objet. En tant que gestionnaire de données, il permet le

partage des données par diverses catégories d'utilisateurs, à travers les vues, ainsi que la manipulation de grandes quantités de données persistantes. En tant que base de connaissances, il permet l'expression des propriétés logiques attachées aux classes d'objets, ici représentées par des vues, et le classement automatique d'instances dans les vues en fonction des propriétés de l'instance considérée. L'intégration de ces deux mondes (données et connaissances) est réalisée grâce aux propriétés que l'utilisateur exprime sur les vues. Il existe plusieurs types de propriétés : une spécialisation de vues (ou héritage), une définition d'attributs et de contraintes sur ces attributs. Les contraintes sur les attributs peuvent être exprimées par un prédicat quelconque sur les attributs. Toutefois, une catégorie particulière de contraintes, appelée assertions est fortement optimisée et sert de base à une partition de l'espace des objets qui est le fondement de l'implantation du système Osiris. Les assertions d'Osiris sont des clauses de Horn dont les littéraux sont de la forme 'Attribut appartient-à Domaine'. Les restrictions de domaine définies généralement par des prédicats (attribut θ valeur) en sont un cas particulier. La nature de ces assertions permet d'effectuer une partition de l'espace des objets en zones de stabilité vis-à-vis des assertions. La notion de partition est proche de la fragmentation horizontale définie dans les années 80 dans le contexte des bases de données relationnelles. Ce qui nous a conduit à étudier la fragmentation dans le contexte des bases de données orientées objet. Nous avons ensuite développé des algorithmes de fragmentation horizontale et verticale.

Projet associé	TIMC-IMAG : OSIRIS
Principales publications associées à ces travaux	[RI-6], [CI-27] [WI-13]

5.3. Travaux de doctorat

Mes travaux de doctorat ont traité deux thèmes : l'un à HKUST a abordé le partitionnement dans les bases de données orientées objet réparties et les entrepôts de données relationnels et l'autre au laboratoire LIMOS de Clermont Ferrand a traité la sélection multiple des structures d'optimisation dans les entrepôts de données relationnels. Ces travaux ont été effectués sous la responsabilité conjointe de Prof. KARLAPALEM de Hong Kong University of Science & Technology et Prof. M. SCHNEIDER du laboratoire LIMOS.

5.3.1. Travaux effectués à HKUST

J'ai poursuivi mon travail, entamé à TIMC, sur le problème de fragmentation dans les bases de données orientées objet (BDOO) au Computer Science Department of Hong Kong University of Science & Technonlogy. Ces travaux ont été lancés pour la première fois par le Prof. KARLAPALEM en 1994. Nous avons développé des concepts, des implémentations et des algorithmes de fragmentation horizontale. Deux types d'algorithmes de fragmentation horizontale ont été proposés: *des algorithmes dirigés par les affinités de prédicats* et *des algorithmes dirigés par un modèle de coût*. Le premier type d'algorithme utilise la fréquence d'accès des prédicats pour générer des fragments horizontaux. Le deuxième utilise un modèle de coût comprenant les différents paramètres logiques et physiques de la base de données pour sélectionner les fragments. Ce même modèle de coût est utilisé pour évaluer la qualité de schéma de fragmentation généré par tout algorithme de fragmentation.

Durant cette période, un autre travail conséquent sur la fragmentation horizontale dérivée dans les BDOO a été réalisé avec la collaboration de Prof. Qing LI (http://www.cs.cityu.edu.hk/~csqli/) du Département d'Informatique de City University of Hong Kong. Dans ce travail, nous avons montré l'intérêt de la fragmentation primaire et

dérivée dans les BDOO dans le but d'optimiser les traitements. Ce travail a fait l'objet d'une publication au *17th International Conference on Conceptual Modeling (ER'98)* [CI-22].

Nous nous sommes également intéressés au problème de l'allocation de fragments issus du processus de partitionnement sur des sites d'un système de base de données réparti. Nous avons proposé des algorithmes d'allocation prenant en considération les instances et les méthodes des fragments.

Avec la naissance des entrepôts de données caractérisés par le volume des données et la complexité de requêtes, nous nous sommes intéressés à la problématique de l'optimisation physique. Notre premier travail consistait à proposer une méthodologie de fragmentation dans les entrepôts de données relationnels prenant en considération les exigences des requêtes décisionnelles, ainsi que les relations entre les tables de dimension et la table des faits. Nous avons étudié les différents scénarii de fragmentation d'un entrepôt de données relationnel. Cette étude nous a permis de proposer un scénario de fragmentation qui permet de partitionner totalement ou partiellement les tables de dimensions d'un schéma en étoile de l'entrepôt de données en utilisant la fragmentation primaire et propager leurs schémas de fragmentation sur la table des faits (qui sera fragmentée en utilisant la fragmentation dérivée).

Nous avons également étudié le problème de l'indexation des vues matérialisées. Nous avons proposé un nouveau type d'index appelé *index de graphe de jointure* adapté pour optimiser les jointures entre les vues matérialisées.

Projet associé	Research Grants Council – RGC CERG HKUST/747/96E
Principales publications associées à ces travaux	[RI-6], [CI-25], [CI-26], [CI22], [CI-23]

5.3.2. Travaux effectués au LIMOS

La plupart des travaux que j'ai menés au TIMC-IMAG et HKUST concerne la sélection isolée de structure d'optimisation dédiée aux requêtes de jointure en étoile. Nous avons identifié une limite de cette sélection. En conséquence, nous nous sommes intéressés à la sélection multiple des structures d'optimisation. Elle peut concerner des structures redondantes (comme la fragmentation horizontale), non redondantes (comme les vues matérialisées et les index) ou les deux. La sélection multiple est plus complexe que la sélection isolée (qui consiste à sélectionner un seul schéma d'optimisation), vue la complexité de son espace de recherche englobant ceux des structures concernées. En plus de cette complexité, une autre s'ajoute qui concerne la gestion des similarités (ou dépendances) entre certaines structures. Prenons l'exemple de deux structures d'optimisation : les vues matérialisées et les index. Elles génèrent deux schémas d'optimisation redondants, partageant la même ressource qui est l'espace de stockage et nécessitent des mises à jour régulières. Nous avons d'abord étudié le problème de la répartition de l'espace de stockage entre les vues matérialisées et les index de jointure. Ensuite, nous avons proposé une sélection conjointe des schémas de fragmentation, d'indexation et des vues matérialisées pour optimiser les requêtes paramétrées.

5.3.3. Résumé de la thèse de doctorat

Un entrepôt de données est une collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour supporter un processus d'aide à la décision. La taille des entrepôts de données peut être très grande. Le processus d'analyse de données s'appuie souvent sur des requêtes de type OLAP qui regroupent et filtrent des données de différentes

formes. Compte tenu de la nature interactive des entrepôts, la nécessité d'avoir un temps de réponse rapide est un objectif critique. Sans *technique d'optimisation de requêtes,* l'interrogation d'un entrepôt serait complexe, et le traitement de ces requêtes pourrait prendre des heures.

Dans ce contexte, nos travaux se sont intéressés à diverses structures d'amélioration des performances des entrepôts de données pour favoriser au mieux les requêtes. Ils interpellent deux niveaux de la conception des entrepôts : le *niveau logique* et le *niveau physique*.

Au niveau logique, nous avons suggéré une méthodologie de fragmentation des structures de données de l'entrepôt. Durant cette période la fragmentation horizontale n'était pas considérée comme un aspect de la conception physique. Nous avons justifié que la *fragmentation horizontale* est la plus appropriée.

Au niveau physique, nous nous sommes intéressés :

1. à la définition et à la sélection d'index de jointure en présence des vues matérialisées et
2. à la distribution de l'espace disque entre les vues matérialisées et les index.

En ce qui concerne l'indexation, nous avons proposé une technique d'indexation de jointure originale appelée *index de graphe de jointure*. Ce type d'index est spécifique aux entrepôts de type ROLAP. Ils peuvent être utilisés sur les vues, les tables de dimensions et la table des faits. Ils permettent de réduire considérablement le coût d'exécution des requêtes. Une stratégie d'exécution des requêtes en présence des index de graphe de jointure a été décrite, et un modèle de coût évaluant le coût d'exécution d'un ensemble de requêtes a été développé. Ensuite, nous formalisé le problème de sélection d'index de jointure en présence d'une contrainte d'espace disque et nous proposons trois algorithmes de résolution optimaux ou quasi-optimaux (un algorithme exhaustif et deux algorithmes gloutons).

Le problème de distribution de l'espace entre les vues matérialisées et les index a été motivé par le fait que ces derniers se définissent comme des structures physiques accélérant les performances des requêtes. Elles présentent certaines similitudes. Il s'agit de structures redondantes, qui partagent la *même ressource (espace disque)*. Toutes deux peuvent entraîner des surcharges causées par les opérations de mise à jour.

Le processus de sélection des vues et des index est contraint par la capacité de stockage que l'administrateur attribue à chacun. La tâche de distribution de l'espace entre les vues matérialisées et les index dans le but d'améliorer les performances des requêtes est très difficile pour l'administrateur. Cette difficulté est due à plusieurs facteurs :

1. la nécessité d'avoir une métrique pour décider de la distribution d'espace entre les vues et les index ;
2. l'interdépendance mutuelle entre les vues et les index ;
3. la nécessité de tenir compte des répercussions des opérations de mise à jour sur les vues et les index ;
4. la nécessité de tenir compte de l'évolution des fréquences des requêtes qui peuvent modifier l'intérêt de certaines vues ou certains index et par conséquent entrainer une redistribution de l'espace.

Dans cette thèse, nous avons répondu à la question suivante: comment *distribuer d'une manière automatique l'espace entre les vues matérialisées et les index afin de garantir une meilleure performance d'exécution d'un ensemble de requêtes décisionnelles?*

Nous avons formalisé le problème de distribution de l'espace entre les vues matérialisées et les index dans le cas statique (où tous les paramètres de l'entrepôt sont connus a priori) et dans le cas dynamique (certains des paramètres de l'entrepôt doivent être réévalués après les opérations de mises à jour). Nous avons proposé un algorithme approché de résolution basé sur l'interaction entre deux agents, l'un opérant pour le compte des vues et l'autre pour le

compte des index. Finalement, nous avons évalué la performance de notre approche et son utilité sur un schéma en étoile.

Principales publications associées à ces travaux	[CI-16], [CI-18], [CI-20]

5.4. Travaux effectués au LISI/ENSMA en qualité de maître de conférences

Depuis mon recrutement en tant que maître de conférences à l'Université de Poitiers, j'ai intégré l'équipe « Ingénierie des Données» du laboratoire LISI de l'ENSMA et l'Université de Poitiers.

Compte tenu de mes compétences en bases de données et entrepôts de données, aussitôt recruté, j'ai développé des travaux portant sur l'intégration de sources structurées dans le cadre du projet *OntoDB* lancé par Prof. PIERRA au début de l'année 2002. L'objectif de ce projet est de permettre la gestion, l'échange, l'intégration et l'interrogation de données structurées associées à des ontologies formelles. En collaborant avec les membres de l'équipe IDD, nous avons développé une approche ascendante d'intégration de bases de données à base ontologique dans un entrepôt de données avec pour application, les bibliothèques de composants industriels (thèse de Nguyen Dung). Ces travaux ont un lien étroit avec mes travaux antérieurs.

Intégration de données hétérogènes en utilisant des ontologies conceptuelles.

L'intégration de données permet de fournir aux utilisateurs une interface d'accès transparente à ces données. Nous avons d'abord identifié trois principales difficultés lors du développement de notre système d'intégration: (1) l'hétérogénéité des données, (2) l'autonomie des sources, et (3) l'évolution des sources. L'hétérogénéité de données concerne à la fois la structure et la sémantique. L'hétérogénéité structurelle provient du fait que les sources de données peuvent avoir différentes structures et/ou différents formats pour stocker leurs données. De nombreux travaux concernant ce type d'hétérogénéité ont été proposés dans les contextes des bases de données fédérées et des multi-bases de données. L'hétérogénéité sémantique, par contre, il présente un défi majeur dans le processus d'élaboration d'un système d'intégration. Elle est due aux différentes interprétations des objets du monde réel. En effet, les sources de données sont conçues indépendamment par des concepteurs différents ayant des objectifs applicatifs différents. Chacun peut donc avoir un point de vue différent sur le même concept. Afin de réduire cette hétérogénéité, nous avons proposé une approche à base ontologique permettant d'associer aux données des ontologies formelles qui en définissent le sens. L'autonomie de sources de données fait référence à la nécessité, pour les sources, de garder leur autonomie tant au niveau de leur schéma qu'au niveau de leur gestion malgré leur implication dans un système d'intégration. L'évolution des sources vise à supporter l'évolution asynchrone, au cours du temps, des différents aspects de chaque source: données, schémas, et le cas échéant, ontologie locale. Cette évolution asynchrone doit également pouvoir concerner, lorsqu'elle existe, l'ontologie partagée.

Le système d'intégration à base ontologique que nous avons proposé dans la thèse de Nguyen DUNG permet (1) une intégration automatique de sources de données, (2) une grande autonomie des sources et (3) une évolution asynchrone des schémas et des ontologies. Nous supposons l'existence d'une ontologie partagée. Chaque source locale référence cette ontologie a priori. L'architecture que nous avons choisie pour réaliser cette intégration est de type « entrepôt », où les données des sources sont dupliquées dans le système d'intégration. Trois scénarii d'intégration (*FragmentOnto*, *ExtendOnto* et *ProjOnto*) ont été proposés et validés en utilisant des ontologies normalisées développées à

l'initiative du laboratoire dans le domaine de l'ingénierie. Cette validation a été effectuée sur le langage EXPRESS dans l'environnement ECCO et JAVA. Elle a été mise en œuvre pour intégrer des ensembles de catalogues de composants de données puis à gérer son évolution à travers les mises à jour des catalogues tant au niveau des définitions contenues (ontologies) qu'au niveau des composants décrits (données).

Projets associés	OSI : PLIB, LISI : OntoDB
Stagiaires de Master 2 Recherche	N. DUNG, A. RAHNI, H. TOUIHRI, N. BELAID
Thèse de doctorat	N. DUNG
Principales publications associées à ces travaux	[RI-5], [RN-3], [RN-4], [RN-6], [CI-8], [CI-12], [CI-15]

Conception de bases de données à base ontologique.

Le deuxième thème de recherche étudié concerne l'étude de la persistance des données ontologiques. Ce thème a été motivé par l'émergence des modèles d'ontologies stables dans différents domaines, OWL dans le domaine du Web sémantique, PLIB dans le domaine technique. De plus en plus de données (ou de méta données) sont décrites par référence à ces ontologies. La taille croissante de telles données rend nécessaire de les gérer au sein de bases de données originales, que nous appelons bases de données à base ontologique (BDBO). Une particularité des BDBOs est de représenter, outre les données, les ontologies qui en définissent le sens. Le travail de thèse de Hondjack DEHAINSALA a porté cette étude. Nous avons commencé par un état des lieux sur les différentes architectures de BDBOs existantes. Les schémas qu'elles utilisent pour la représentation des données sont soit constitués d'une unique table de triplets de type (sujet, prédicat, objet), soit éclatés en des tables unaires et binaires respectivement pour chaque classe et pour chaque propriété. Si de telles représentations permettent une grande flexibilité dans la structure des données représentées, elles ne sont ni susceptibles de passer à grande échelle lorsque chaque instance est décrite par un nombre significatif de propriétés, ni adaptée à la structure des bases de données usuelles, fondée sur les relations n-aires. C'est ce double inconvénient que vise à résoudre le modèle OntoDB que nous avons proposé dans le cadre de cette thèse. L'architecture de BDBO proposée analogique à l'architecture du MOF pour le W3C, est constituée de quatre parties : les deux premières parties correspondent à la structure usuelle des bases de données : données reposant sur un schéma logique de données, et méta-base décrivant l'ensemble de la structure de tables. Les deux autres parties, originales, représentent respectivement les ontologies, et le méta-modèle d'ontologie au sein d'un méta-schéma réflexif. Des mécanismes d'abstraction et de nomination permettent respectivement d'associer à chaque donnée le concept ontologique qui en définit le sens, et d'accéder aux données à partir des concepts, sans se préoccuper de la représentation des données. Cette architecture permet à la fois de gérer de façon efficace des données de grande taille définies par référence à des ontologies, mais aussi d'indexer des bases de données usuelles au niveau connaissance en leur adjoignant les deux parties : ontologie et méta-schéma. Le modèle OntoDB a été validé par un prototype opérationnel implanté sur PostgreSQL avec le modèle d'ontologies PLIB. Nous avons effectué des tests sur des bancs d'essai pour valider nos propositions.

Projet associé	ANR : EWOK-HB, ANR : DAFOE4APP, LISI : ONTODB
Stagiaires de Master 2 Recherche	C. FANKAM*, M. BACHIR, HONDJACK

Thèses de doctorat	C. FANKAM*, HONDJACK
Publications associées à ces travaux	[CI-9], [CN-19], [RN-6]

* Chimène FANKAM doit soutenir sa thèse en novembre 2009.

Conception de bases de données à partir d'une ontologie de domaine

Vu les similarités entre les modèles conceptuels et les ontologies de domaine, nous avons proposé une approche de conception de bases de données à partir d'une ontologie de domaine. Cette approche, appelée *SISRO* pour Spécialisation Importation Spécifique et Représentation des Ontologies, utilise une ontologie locale pour assurer une meilleure autonomie de la base de données par rapport à l'ontologie de domaine. Elle représente l'ontologie locale et les ontologie(s) partagée(s) soit par un mécanisme de vues, soit par une représentation explicite des ontologies au sein de la base de données. Les classes de l'ontologie locale sont différentes des classes des ontologies partagées avec lesquelles elles sont articulées par des relations de subsomption. Elles importent à travers cette relation celles des propriétés applicables à leur classe subsumante qui s'avèrent pertinentes pour le cahier des charges à satisfaire. Notre méthodologie comporte quatre étapes principales. Dans l'étape 1, une ontologie locale est construite par identification des concepts et propriétés des ontologies partagées pertinents par rapport au cahier des charges. Le concepteur dispose d'une grande liberté dans la structuration des classes (qui représentent les concepts retenus) et des propriétés. Cette ontologie locale peut alors être spécialisée par les concepts nécessaires qui ne figuraient pas dans les ontologies partagées, puis étendue par ajout de nouvelles propriétés. Dans la seconde étape, le modèle conceptuel de l'application est défini à partir de l'ontologie locale. Le concepteur peut ne choisir qu'un sous ensemble de l'ontologie locale pour prévoir par exemple les extensions futures. Cette différence entre ontologie locale et modèle conceptuel de données est justifiée par le caractère descriptif des ontologies et la nature prescriptive des modèles conceptuels. Dans la troisième étape, le modèle conceptuel est traduit en modèle logique, puis en modèle physique à partir d'un ensemble de règles de transformation suivant le formalisme utilisé. Enfin, la quatrième étape fournit un accès aux données au niveau ontologique afin de permettre l'intégration automatique. Dans cette étape, le concepteur a le choix entre deux possibilités : (1) utiliser des vues pour représenter les classes des ontologies concernées ; (2) représenter dans la même base de données à la fois les données, les concepts ontologiques qui en définissent le sens (ontologie locale, ontologie partagée), mais aussi le lien entre ontologie et données au travers des identifiants de concepts.

Dans ce dernier cas, une interface graphique interactive et générique peut être développée pour permettre à un utilisateur de parcourir les concepts de la base au niveau connaissance, et un langage de requête fournissant l'accès aux données par le niveau ontologique peut être rendu disponible. La méthode SISRO est entièrement outillée par une suite d'outils qui permettent d'en réaliser toutes les étapes. Ces outils sont déjà mis en œuvre dans diverses applications d'ingénierie.

Projets associés	ANR : EWOK-HB, ANR : DAFOE4APP, LISI: ONTODB
Stagiaires de Master 2 Recherche	C. CHAKROUN[1]

[1] Chedlia CHEKROUN actuellement en première année de thèse encadrée conjointement par Yamine AIT AMEUR et moi même.

Thèse de doctorat	C. CHAKROUN
Principales publications associées à ces travaux	[RN-3], [CI-6]

Conception physique des entrepôts de données

Parallèlement à mes travaux sur les ontologies et les bases de données à base ontologique, j'ai poursuivi mes travaux portant sur la conception physique des entrepôts de données (suite de mes travaux de thèse). La thèse de BOUKHALA soutenue en juillet 2009 entre dans ce cadre. Nous avons commencé par analyser les différentes techniques d'optimisation existantes. Cette analyse nous a permis de proposer une classification de ces techniques en deux catégories : les techniques redondantes comme les vues matérialisées et les index du fait qu'elles nécessitent un espace de stockage et un coût de maintenance et les techniques non redondantes comme la fragmentation et le traitement parallèle. Pour élaborer une conception physique d'un entrepôt de données relationnel, nous nous sommes concentrés sur deux techniques d'optimisation, à savoir la fragmentation horizontale et les index de jointure binaires. Ce choix a été motivé par l'existence des similarités entre les deux techniques. En ce qui concerne la fragmentation, nous avons d'abord proposé une méthodologie pour partitionner un entrepôt de données. Nous avons ensuite formalisé le problème de sélection de schéma de fragmentation. En collaboration avec Pascal RICHARD, membre de l'équipe *Temps Réel* du LISI, nous avons montré son NP-complétude. Pour le résoudre, nous avons proposé trois algorithmes de sélection : un algorithme glouton, un algorithme génétique et un algorithme de recuit simulé.

Pour comparer ces algorithmes, nous avons développé un modèle de coût à base mathématique qui permet d'estime le nombre d'entrées sorties nécessaires pour exécuter un ensemble de requêtes. Les expérimentations ont montré des résultats encourageants sur l'utilisation de la fragmentation horizontale. Nous avons mené une validation des schémas de fragmentation proposés par les trois algorithmes sous ORACLE10G. Cette validation a confirmé les résultats théoriques.

Pour les index de jointure en étoile, nous avons d'abord formalisé le problème de sélection des index comme un problème d'optimisation à contrainte représentant l'espace de stockage. Pour le résoudre nous avons ensuite proposé deux types d'algorithmes : un algorithme glouton et un algorithme dirigé par les techniques de fouille de données. Ce dernier a été développé en collaboration avec Rokia MISSAOUI professeur à l'Université du Québec en Outaouais (http://w3.uqo.ca/missaoui/), lors de son séjour scientifique en France.

Stagiaires de Master 2 Recherche	K. WOAMENO
Elève Ingénieur (ENSMA) Magister	Y. PANDELEY
Etudiant Magister	T. BIDI, S. BENKRID, R. BOUCHAKRI
Thèse de doctorat	K. BOUKHALFA
Principales publications associées à ces travaux	[RI-1], [RI-2], [RI-3], [RI-4], [RN-2], [RN-5], [CI-1], [CI-2], [CI-11]

Personnalisation de requêtes OLAP

Les contacts établis avec Prof. Arnaud GIACOMETTI, Dr. Patrick MARCEL et Hassina MOULOUDI du Laboratoire d'Informatique (LI) de l'Université de Tours lors de la

conférence BDA'04 à Montpellier ont permis de lancer une collaboration concrétisée par la soutenance de la thèse de Hassina MOULOUDI sur l'application des techniques de recherche d'informations pour personnaliser les requêtes OLAP. Cette personnalisation est cruciale pour les utilisateurs finaux des outils d'aide à la décision qui comptent fortement sur la visualisation des réponses de requêtes de leurs analyses interactives sur des quantités importantes de données. Très souvent, ces réponses ne peuvent pas être visualisées entièrement sur le dispositif d'affichage (écran d'ordinateur, PDA, etc.). En conséquence, l'utilisateur doit naviguer longuement afin de trouver l'information recherchée. L'idée que nous avons eue consiste à personnaliser les requêtes décisionnelles en prenant en compte les profils des utilisateurs. Pour ce faire, chaque utilisateur (décideur) de l'entrepôt de données est invité à donner ses préférences (son profil) et une contrainte de visualisation. Cette contrainte peut être, par exemple, les limitations imposées par le dispositif utilisé pour afficher la réponse d'une requête. Ce travail a abouti à la thèse de Mme MOULOUDI soutenue en Décembre 2007. Cette dernière a validé son travail en réalisant un prototype « Mobile OLAP » pour les applications mobiles, où l'entrepôt de données a été stocké et géré par le SGBD ORACLE.

Stagiaires de Master 2 Recherche	A. YAGOUB
Thèse de doctorat	H. MOULOUDI
Principales publications associées à ces travaux	[WI-6], [CN-14], [CN-18]
Organisation Conjointe de Conférence	EDA07
Collaboration entre le laboratoire d'informatique de Tours et le LISI/ENSMA	Voir Section Contrats : 2 500 € (2009-2010)

6. ENCADREMENT

Thèses soutenues	3
Thèses en cours	3
Stages de Master Recherche	10
Stages de Master Professionnel	4
Stages Ingénieur ENSMA + CNAM	2
Stages de Magister	7

6.1. Encadrements de thèses

Actuellement, je co-encadre 3 étudiants en thèse. J'ai encadré 3 thèses et en moyenne 2 stages de DEA/Master 2 Recherche par an.

6.1.1. Thèses soutenues

2003 – 2006 Mr. Nguyen XUAN – Bourse MRT

- Titre : Une approche ascendante d'intégration des bases de données à base ontologique : Application aux bibliothèques de composants industriels.
- Soutenue le 22 Décembre 2006.
- Mention : Très honorable.
- Encadrement à 50% (co-encadrée avec Prof. PIERRA)
- Composition de Jury:

    - Prof. Anne DOUCET - Université de Pierre et Marie Curie : **Rapporteur**
    - Prof. Chantal REYNAUD – Université de Paris XI : **Rapporteur**
    - Prof.Michel SCHNEIDER – Université de Blaise Pascal – Clermont Ferrand
    - Dr. Marie Christine LAFAYE – Université de La Rochelle
    - Prof. Yamine AIT AMEUR – ENSMA
    - Prof. Guy PIERRA – ENSMA
    - Dr. Ladjel BELLATRECHE – Université de Poitiers
- Devenir : Création d'une entreprise à Hanoi Vietnam : http://www.vietinfotech.net

2003-2007 Mr. Hondjack DEHAINSALA – Bourse CIFRE

- Titre : Explicitation de la sémantique dans les bases de données : base de données à base ontologique et le modèle OntoDB.
- Soutenue le mai 2007.
- Mention : Très honorable.
- Encadrement à 50% (co-encadrée avec Prof. PIERRA)
- Composition du Jury :
    - Dr. Jean CHARLET –Université Paris 6 : **Rapporteur**
    - Prof. Mohand-Said HACID – Université de Lyon 1 : **Rapporteur**
    - Prof. Mokrane BOUZEGHOUB - Université de Versailles
    - Prof. Michel SCHNEIDER Université de Blaise Pascal – Clermont Ferrand
    - Dr. Jean Yves LAFAYE – Université de La Rochelle
    - Prof. Guy PIERRA – ENSMA
    - Dr. Ladjel BELLATRECHE – Université de Poitiers
- Devenir : Actuellement Ingénieur R&D à Mondeca – Paris : http://www.mondeca.com

2005 : 2009 Kamel BOUKHALFA – Bourse Franco-algérienne

- Titre : De la Conception Physique aux Outils d'Administration et de Tuning des Entrepôts de Données
- Soutenue 2 juillet 2009
- Mention : Très honorable.
- Encadrement à 90% (co-encadrée avec Prof. PIERRA)
- Composition du Jury :
    - Prof. Arnaud GIACOMETTI, Université de Tours : **Rapporteur**
    - Prof. Jérôme DARMONT, Université de Lyon 2 : **Rapporteur**
    - Prof. Zohra BELLAHSENE, Université de Montpellier 2
    - Prof. Guy PIERRA, ENSMA
    - Prof. Pascal RICHARD, Université de Poitiers
    - Dr. Ladjel BELLATRECHE – Université de Poitiers
- Devenir : Maître Assistant à l'Université des Sciences et de la Technologie Houari Boumediene (USTHB), Alger, Algérie.

6.1.2. Thèses en cours

2006 – 2009 Chimène FANKAM (Thèse en cours) – Bourse MRT

- Sujet : OntoDB2 : un modèle efficace de gestion des bases de données à base ontologique pour les données techniques et le Web sémantique.
- Co-encadrement à 40% avec Guy PIERRA.
- Soutenance prévue décembre 2009.

- Lieu : LISI/ENSMA
- Projet associé : ANR : EWok Hub

2008 - 2010 : Henry Valéry TEGUIAK (Thèse en cours) – Bourse CIFRE
- Sujet : Construction d'ontologies à partir de textes: une approche basée sur l'Ingénierie Dirigée par les Modèles.
- Co-encadrement à 50% avec Guy PIERRA.
- Soutenance prévue septembre 2011.
- Lieu : LISI/ENSMA
- Projet associé : ANR : Dafoe

2009 – 2012 : Chedlia CHAKROUN (Thèse en cours) – Bourse Région Poitou Charente
- Sujet : Une démarche de conception de Base de données à base ontologique.
- Co-encadrement à 50% avec Yamine AIT AMEUR.
- Soutenance prévue septembre 2012.
- Lieu : LISI/ENSMA

6.2. Encadrements de Master Recherche

- 2008 – 2009 :
  - Chedlia CHAKROUN
    - Sujet : OntoLMDesigner : Génération de modèle logique de données normalisé à partir d'une ontologie de domaine.
    - Master 2 Informatique Télécommunication. Parcours Bases de Données Technologies Web - Université de Poitiers.
    - Encadrement à 100%.
  - Bery MBAIOSSOUM
    - Sujet : Benchmarking des systèmes de gestion des bases de données à base ontologique.
    - Master 2 Informatique Télécommunication. Parcours Bases de Données Technologies Web - Université de Poitiers.
    - Encadrement à 60% avec Dr. Stéphane JEAN.
    - Publication associée [RN-1].
  - Komla WOAMENO
    - Vers la simulation de la conception physique d'un entrepôt de données.
    - Master 2 Informatique Télécommunication. Parcours Bases de Données Technologies Web - Université de Poitiers - Université de Poitiers.
    - Encadrement à 100%.
    - Publications associées [RI-1], [WI-1].

- 2006-2007 **:**
  - Nabil BELAID
    - Sujet : Panorama de travaux autour de l'intégration de données hétérogènes, autonomes et évolutives.
    - Master 2 STIC – Université de Poitiers.
    - Encadrement à 50% avec Prof. Guy PIERRA.
    - Publication associée [CN-10]

- 2005-2006 **:**
  - Mounira BACHIR

        - Master 2 STIC – Université de Poitiers.
        - Encadrement à 50% avec Prof. Guy PIERRA.
        - Hichem TOUARI : Master 2

- 2004 – 2005 :
    - Ahmed RAHNI
        - Sujet : AMIDHA : une Approche Médiateur d'Intégration de Sources de Données Hétérogènes et Autonomes.
        - Master 2 STIC – Université de Poitiers
        - Encadrement à 50% avec Prof. Guy PIERRA.
    - Aghilas YAGOUB
        - Sujet : contribution de la visualisation à l'optimisation de requêtes OLAP.
        - Master 2 STIC – Université de Poitiers
        - Encadrement avec Dr. Patrick MARCEL du Laboratoire d'Informatique de Tours.

    - Dalal SHOUFAN
        - Sujet : Conception d'une architecture de recherche sémantique dans une base de données de vidéo.
        - Master2 STIC – Université de Poitiers.
        - Encadrement avec M. Chaker LARABI du Laboratoire SIC (Signal Image Communication) – Université de Poitiers.

- 2003 – 2004 **:**
    - Hichem ZAIT
        - Sujet : Conception d'une Architecture d'Intégration de Sources de Données Hétérogènes Basée sur Ontologie.
        - D.E.A. Traitement de l'Information : Informatique, Image, Automatique (T3IA) – Université de Poitiers.
        - Encadrement à 50% avec Prof. Guy PIERRA.

- 2002 – 2003 :
    - Dung NGUYEN XUAN
        - Sujet : Intégration de données hétérogènes basée sur des ontologies PLIB.
        - D.E.A. Traitement de l'Information : Informatique, Image, Automatique (T3IA) – Université de Poitiers.
        - Encadrement à 50% avec Prof. Guy PIERRA.
        - Publications associées : [RI-5], [RN-4], [CI-10], [CI-12], [CI-15], [CI-17]

6.3. Encadrements Master Professionnel

- 2008-2009
    - Développement d'un outil de génie logiciel de conception de bases de données à partir d'une ontologie de domaine.
        - Master 2 Informatique Télécommunication - Université de Poitiers.

- 2007-2008
    - Exploitation des techniques de data mining pour l'auto administration des entrepôts de données sous ORACLE 10g.
        - Master 2 Fondements et Ingénierie de l'Informatique et de l'Image (F3I) Poitiers.

- 2006-2007
  - Personnalisation des requêtes OLAP sous ORACLE 10g.
    - Master 2 Fondements et Ingénierie de l'Informatique et de l'Image (F3I) Poitiers.
- 2005-2006
  - Techniques de data mining et détection des intrusions.
    - DESS Risque d'Informatique Niort – IRIAF.

6.4. Encadrements d'élèves Ingénieurs

- 2009-2010 :
  - Mathieu SERVAIS
    - Sujet : Programmation linéaire du problème de sélection de schéma de fragmentation horizontale.
    - Ingénieur 3ème année ingénieur informatique à Polytech'Tours
    - Co-encadrement avec Pascal Richard du LISI/ENSMA et Vincent T'KINDT Laboratoire d'informatique de l'Université de Tours.

- 2006-2007
  - Yvan PANDELEY **:**
    - Sujet **:** Etude empirique pour la sélection de schéma de fragmentation horizontale d'un entrepôt de données relationnel
    - Ingénieur 2ème année ENSMA

- 2007-2008
  - Chedlia CHAKROUN
    - Sujet **:** Gestion de concepts non canoniques dans les bases de données ontologiques.
    - Ingénieur de l'Ecole Nationale des Ingénieurs à Sousse – Tunisie.
    - Lieu : LISI/ENSMA

6.5. Encadrement d'Ingénieur CNAM

- **2002-2003**
  - Hervé LORINQUER :
    - Sujet : Sélection d'index, de vues matérialisées et de partitions pour l'amélioration des performances dans les entrepôts de données.
    - Ingénieur CNAM – Université de Blaise Pascal Clermont Ferrand.
    - Co-encadrement avec Prof. Michel SCHNEIDER (LIMOS)
    - Publication associée : [CI-16]

6.6. Encadrements de Magister – Algérie

6.6.1. Magisters soutenus

- Décembre 07- Janvier 2009 :
  - Tahar BIDI :
    - Sujet : Optimisation de requêtes OLAP.
    - Lieu : Ecole Nationale Supérieure d'Informatique (ESI) d'Alger
    - Soutenu 28/01/2009
  - Amel BOUSSIS

        - Sujet : Intégration de sources de données à base ontologique dans un environnement pair à pair.
        - Lieu : ESI
        - Soutenu 28/01/2009
    - Zoubir OUARET :
        - Sujet : Conception logique et physique d'un entrepôt de données XML.
        - Lieu : ESI
        - Soutenu 27/01/2009.
        - Co-encadrement avec Omar BOUSSAID (Laboratoire Eric Université de Lyon 2).
        - Publication associée [CN-11].

- Septembre 07 – Juin 2009 :
    - Soumia BENKRID
        - Sujet : Conception d'un entrepôt de données parallèle.
        - Lieu : ESI
        - Soutenue le 24/06/2009.
        - Publications associées : [CI-2], [WI-2], [CN-1]

6.6.2. Magisters en cours

- Septembre 2008 …
    - Selma KHOURI
        - Sujet : Une approche ontologique de conception d'un entrepôt de données.
        - Lieu : ESI
        - Soutenance prévue fin décembre 2009.
        - Publication associée : [CN-2].
    - Lamia SADEG
        - Sujet : Optimisation des requêtes complexes par étude de similarité entre plans.
        - Lieu : École Militaire Polytechnique (EMP) Alger
        - Soutenance prévue fin décembre 2009
    - Ryma BOUCHAKRI
        - Sujet : Tuning des entrepôts de données.
        - Lieu : ESI
        - Soutenance prévue fin décembre 2009.
        - Co-encadrement avec Kamel BOUKHALFA – USTHB – Algérie.

6.7. Participation à des Jurys de Thèse de Doctorat (extérieurs au LISI)

- Mars 2008
    - Arpita AGGARWAL
    - Titre : Evaluation of adaptive hypermedia systems
    - Lieu: University of Delhi – Inde
    - Rapporteur

- 21 Décembre 2007 :
    - Hassina MOULOUDI
    - Titre : Personnalisation de requêtes et visualisations OLAP sous contraintes.
    - Lieu : Université de Tours
    - Examinateur

- 27 juin 2007
  - Mohamed SENOUCI
    - Titre : Une approche analytique pour la reconnaissance de l'écriture manuscrite »
    - Lieu : Université d'Oran Algérie
    - Rapporteur
  - Janvier 2008 :
    - Zakaria ELBERRICHI
    - Titre : Text Mining.
    - Lieu : Université de Sidi Bel Abbès Algérie
    - Rapporteur.

7. PUBLICATIONS

Nature de publications	Nombre
Thèse de Doctorat	**1**
Edition de Livre	**3**
Articles parus dans une revue internationale avec comité de lecture	**6**
Chapitres de Livre	**6**
Articles parus dans une revue nationale avec comité de lecture	**6**
Articles parus dans des actes de conférences internationales avec comité de lecture	**27**
Articles parus dans des actes de workshops internationaux avec comité de lecture	**12**
Articles parus dans des actes de conférences nationales à comité de lecture	**23**
Rapports de Recherche	**2**

7.1. Thèse de doctorat

[TD-1] Bellatreche, L. Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique d'un entrepôt de données. Université de Clermont-Ferrand II. 18 décembre 2000. Nombre de pages : 189.

7.2. Livres

[L-1] Bellatreche, L. New Trends in Physical Data Warehouse Design, Special issue of Distributed and Parallel Database Journal, Springer, 2009, nombre de pages: 123. ISBN: 0926-8782.
[L-2] Ladjel Bellatreche, Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction, Information Science Reference, August 2009. Nombre de pages: 336. ISBN : 978-1-60566-756-0.
[L-3] Bellatreche L., Marcel, P. and Giacometti A. Entrepôts de Données et Analyse en ligne (EDA'07), Editions Cépaduès, Juin 2007. Nombre de pages: 190. ISBN: 1764-1667

7.3. Revues internationales avec comité de lecture

[RI-1] Bellatreche, L., Boukhalfa, K., Richard, P. and Woameno, K. Y. Referential Horizontal Partitioning Selection Problem in Data Warehouses: Hardness Study and Selection Algorithms, in International Journal of Data Warehousing and Mining, 5(4), pp. 1-23, 2009.
[RI-2] Bellatreche, L. New Trends in Physical Data Warehouse Design, in Distributed and Parallel Database Journal, Springer Verlag, pp. 1-3, 25(1-2), February 2009.

[RI-3] Boukhalfa, K., Bellatreche, L. and Alimazighi, Z. HP&BJI: A Combined Selection of Data Partitioning and Join Indexes for Improving OLAP Performance, Annals of Infomation Systems special issue on new trends in Data Warehouse and OLAP, Vol. 3, November 2008, pp. 179-2001, Springer Verlag.
[RI-4] Bellatreche L., Missaoui R., Necir H. and Drias H. Data Mining Driven Approach for Pruning and Selectiong Bitmap Join Indexes, in Journal of Computing Science and Engineering, 2(1), pp. 206-223, December, 2007.
[RI-5] Bellatreche, L., Dung, N. X., Pierra G. and Hondjack, D., Contribution of Ontology-based Data Modeling to Automatic Integration of Electronic Catalogues within Engineering Databases, in Computers in Industry Journal, Elsevier, 57 (8-9), pp. 711-724, 2006.
[RI-6] Bellatreche, L., Karlapalem, K. and Simonet, A. Algorithms and Support for Horizontal Class Partitioning in Object-Oriented Databases. Distributed and Parallel Databases Journal, Springer Verlag, 8(2), pp. 155 - 179, April 2000.

7.4. Chapitres de livre

[CL-1] Bellatreche, L. Optimization and Tuning in Data Warehouses, Encyclopedia of Database Systems (EDS), Edited by Ling Liu and Tamer Özsu, Springer, 2009, pp. 69-76.
[CL-2 Bellatreche, L. Bitmap join indexes vs. Data Partitioning, in Encyclopedia of Data Warehousing and Mining, edited by John Wang, IGI Group Publishing.
[CL-3] Bellatreche, L. A Genetic Algorithm for Selecting Horizontal Fragments, in Encyclopedia of Data Warehousing and Mining, IGI Group Publishing.
[CL-4] Amine, A., El Berrichi, Z., Simonet, M., Bellatreche, L. and Malki, M. SOM-based Clustering of Textual Documents Using WordNet, in Handbook of Research on Text and Web Mining Technologies, IGI Group Publishing.
[CL-5] Bellatreche, L., Karlapalem and K. Mohania, M. Some Issues in Design of Data Warehousing Systems**,** in Developing Quality Complex Data Bases Systems: Practices, Techniques, and Technologies, Edited by Dr. Shirley A. Becker, IDEA Group Publishing, 2001, pp. 125-172.
[CL-6] Bellatreche, L**.** and Mohania, M., Physical Data Warehousing Design'', in Encyclopedia of Data Warehousing and Mining, edited by John Wang, IDEA Group Publishing, 2005, pp. 906-912.

7.5. Revues nationales avec comité de lecture

[RN-1] Fankam, C., Jean, S., Pierra, G., Bellatreche, L. and Mbaiossoum, B. Raisonnement Numérique sur les Relations d'Ordre pour le Web Sémantique, à apparaître dans Ingénierie des Systèmes d'Information (ISI).
[RN-2] Boukhalfa, K., Bellatreche, L. and Caffiau, S. De l'optimisation de requêtes aux outils d'administration des entrepôts de données. Ingénierie des Systèmes d'Information (ISI), 13(6/2008), 2008, pp. 33-62.
[RN-3] Fankam, C. Bellatreche, L., Dehainsala, H, Ait Ameur, Y. and Pierra, G. SISRO : conception de bases de données à partir d'ontologies de domaine. Technique et Science Informatiques (TSI), 28, 2009, pp. 1-29.
[RN-4] Dung, N. X., Bellatreche, L. and Pierra G. OntoDaWa : Un système d'intégration à base ontologique de sources de données autonomes et évolutives. Revue d'Ingénierie des Systèmes d'Information (ISI), 13(2), pp. 97-125, 2008
[RN-5] Boukhalfa, K. and Bellatreche, L. Sélection de schéma de fragmentation horizontale dans les entrepôts de données : formalisation et algorithmes, Revue d'Ingénierie des Systèmes d'Information (ISI), 11(6/2006), pp. 55-82, Novembre, 2006.

[RN-6] Pierra, *G.*, Dehainsala, H., Aït-Ameur, Y. and Bellatreche, L., Base de données à base ontologique : principe et mise en œuvre, Ingénierie des systèmes d'information (ISI), 10(2), pp. 91-116, 2005.

7.6. Conférences internationales avec comité de lecture

[CI-1] Bellatreche, L, Boukhalfa, K. and Alimazighi, Z. SimulPh.D.: A Physical Design Simulator. Proceedings of 20th International Conference on Database and Expert Systems Applications (DEXA09), pp. 263-270, LNCS 5690, Springer, September 2009, Linz, Austria.
[CI-2] Bellatreche, L. and Benkrid, S. Proceedings of 11th International Conference on Data Warehousing and Knowledge Discovery (DAWAK09), pp. 99-110, LNCS 5691 Springer, September 2009, Linz, Austria.
[CI-3] Fankam, C., Jean, S., Pierra, G., Bellatreche, L. and Aït Ameur, Y., Towards Connecting Database Applications to Ontologies. Proceedings of the First International Conference on Advances in Databases (DB'09) , March, 2009.
[CI-4] Bellatreche, L., Boukhalfa, K. and Richard, P. Horizontal Partitioning in Data Warehouse: Hardness Study, Selection Algorithms and Validation on ORACLE10G. Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2008), pp. 87-96, LNCS 5182, Springer, September, 2008
[C-5] Abdelmalek, A., Elberrichi, Z., Bellatreche, L., Simonet, M. and Malki, M. Concept-based clustering of textual documents using SOM. Proceedings of the 6th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'08), pp. 156-163, Doha, Qatar, March 2008.
[CI-6] Fankam, C., Jean, S., Bellatreche, L. and Aït Ameur, Y. Extending the ANSI/SPARC Architecture Database with Explicit Data Semantics: An Ontology-Based Approach. Proceedings of the 2nd European Conference on Software Architecture (ECSA'2008), LNCS 5292 Springer, September, 2008, pp. 318-321.
[CI-7] Bellatreche L., Boukhalfa, K. and Mohania, M. Pruning Search Space of Physical Database Design. Proceedings of the 18th International Conference on Database and Expert Systems Applications (DEXA'07) , pp. 479-488, LNCS 4653 Springer, September, 2007,
[CI-8] Bellatreche L., Missaoui R., Necir H. and Drias H. Selection and Pruning Algorithms for Bitmap Index Selection Problem using Data Mining. Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'07), pp. 221-230, LNCS 4654 Springer, September, 2007
[CI-9] Dehainsala H., Pierra G. and Bellatreche L. OntoDB: An Ontology-Based Database for Data Intensive Applications. Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07), LNCS 4443 Springer, Bangkok - Thailand, April, 2007, pp. 497-508.
[CI-10] Jean S., Dehainsala H., Xuan, D. N., Pierra G., Bellatreche L. and Aït-Ameur, Y. OntoDB: It is Time to Embed your Domain Ontology in your Database. Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07), LNCS 4443 Springer, Bangkok - Thailand, April, 2007, pp. 1119-1122
[CI-11] Bellatreche, L., Boukhalfa, K. Abdalla, H. SAGA: A Combination of Genetic and Simulated Annealing Algorithms for Physical Data Warehouse Design. Proceedings of 23rd British National Conference on Databases (BNCOD'06), LNCS 4042 Springer, Belfast, Northern Ireland, July, 2006, pp. 212-219.
[CI-12] Dung, N. X., Bellatreche, L., Pierra G., A Versioning Management Model for Ontology-Based Data Warehouses. Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'06), LNCS 4081 Springer, September 2006, pp. 195-206

[CI-13] Dehainsala H., Pierra, G. and Bellatreche, L. Benchmarking Data Schemes of Ontology Based Databases. Proceedings of the 5th International Confenrence on Ontologies, Databases, and Applications of Semantics (ODBASE'06), pp. 48-49, LNCS 4277, Springer, 2006, Montpellier.
[CI-14] Bellatreche, L. and Boukhalfa, K., An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse Environment. Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery (DAWAK'05), LNCS 3589 Springer, pp. 115-125, 2005.
[CI-15] Bellatreche, L., Pierra, G. Xuan, D. N., Hondjack, D., Aït Ameur, Y., An a Priori Approach for Automatic Integration of Heterogeneous and Autonomous Databases. Proceedings of the 15th International Conference on Database and Expert Systems Applications (DEXA04), LNCS, 3180 Springer, pp. 475-485, Zaragosa, September 2004.
[CI-16] Bellatreche, L., Schneider, M., Lorinquer, H. and Mohania, M., Bringing Together Partitioning, Materialized Views and Indexes to Optimize Performance of Relational Data Warehouses. Proceedings of the International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2004), LNCS 3181 Springer, pp. 15-25.
[CI-17] *Bellatreche, L.*, Pierra G., Dung, N. X. and Hondjack, D., An Automated Information Integration Technique using an Ontology-based Database Approach, in Proceedings of Concurrent Engineering Conference - a Special Track on Data Integration in Engineering (DIE'2003), pp. 217–224, July 2003, Madera.
[CI-18] Bellatreche, L., Schneider, M., Mohania, M. and Bhargava, B. PartJoin: An Efficient Storage and Query Execution for Data Warehouses. Proceedings of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2002), LNCS, 2454 Springer, September 2002, Aix-en-Provence, France.
[CI-19] Mohania, M., Kambayashi, Y., Tjoa, A. M., Wagner, R. and Bellatreche, L., Trends in Database Research, Invited paper in 12th International Conference on Databases and Expert Systems, (DEXA'01), pp. 984-988, LNCS 2113 Springer, September, 2001, Munich, Germany.
[CI-20 Bellatreche, L., Karlapalem and Schneider M. On Efficient Space Distribution among Materialized Views and Indices in Data Warehousing Environments. Proceedings of the ACM International Conference on Information and Knowledge Management (ACM-CIKM'2000), pp. 397 – 404, November 2000, Virginia.
[CI-21] Bellatreche, L., Karlapalem and Li, Q. Evaluation of View Indexing in Data Warehousing Environments. Proceedings of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2000), pp. 57 – 66, LNCS 1874 Springer, September, 2000, London
[CI-22] Bellatreche, L., Karlapalem, K. and Qing L. Derived Horizontal Class Partitioning in OODBSs: Design Strategy, Analytical Model and Evaluation. Proceedings of 17th International Conference on Entity Relationship Approach (ER'98), LNCS 1507 Springer, pp. 465-479, November 1998, Singapore.
[CI-23] Bellatreche, L., Karlapalem, K. and Qing, L. An Iterative Approach for Rules and Data Allocation in Distributed Deductive Database Systems. Proceedings of the 7th ACM International Conference on Information and Knowledge Management (ACM CIKM'98), pp. 356-363, November 1998, Washington D.C.
[CI-24] Bellatreche, L., Karlapalem, K. and Qing, L. Complex Methods and Class Allocation in Distributed Object-Oriented Databases. Proceedings ot the 5th International Conference on Object Oriented Information Systems (OOIS'98), pp. 239-256, September 1998, Paris.
[CI-25] Bellatreche, L., Karlapalem, K. and Basak, G. B. Query-Driven Horizontal Class Partitioning in Object-Oriented Databases. Proceedings of the 9th International Conference on

Databases and Expert Systems (DEXA'98), LNCS 1460 Springer, pp. 692-701, August 199, Vienna, Austria.
[CI-26] Bellatreche, L., Karlapalem, K. and Simonet, Horizontal Class Partitioning in Object-Oriented Databases. Proceedings of the 8th International Conference on Databases and Expert Systems (DEXA'97), LNCS 1308 Springer, pp. 58-67, Toulouse, September 1997.
[CI-27] Bellatreche, L. and Simonet, A. A Horizontal Fragmentation Algorithm in Object Database Design. Proceedings of the third International Conference of the Austrian Center for Parallel Computation (ACPC'96), with Special Emphasis on Parallel Databases and Parallel I/O, LNCS 1127 Springer, pp. 223-226, Klagenfurt, Austria, September 1996.

7.7. Workshops internationaux avec comité de lecture

[WI-1] Bellatreche, L. and Woameno, K. Y. Dimension Table driven Approach to Referential Partition Relational Data Warehouses, to appear in ACM Twelfth International Workshop on Data Warehousing and OLAP (DOLAP'09), November, 2009, Hong Kong
[WI-2] Benkrid, S., Bellatreche, L. and Drias, H. A Combined Selection of Fragmentation and Allocation Schemes in Parallel Data Warehouses. Proceedings of 4th International Workshop on Data Management in Global Data Repositories (GREP'08), edited by IEEE Computer Society Press, pp. 370-374, Septembre, 2008,
[WI-3] Bellatreche L. Selection of Redundant and non Redundant Optimization Structures in VLDBs. Proceedings of 3rd International Workshop on Data Management in Global Data Repositories (GRep' 2007), pp. 819-824, edited by IEEE Computer Society Press, September, 2007.
[WI-4] Bellatreche L. and Pierra G. OntoAPI: An Ontology-based Data Integration Approach by an a Priori Articulation of Ontologies. Proceedings of 4th International Workshop on P2P Data Management, Security and Trust (PDMST'07), pp. 799-803, edited by IEEE Computer Society Press, September, 2007.
[WI-5] Ziyati, E., Boukhalfa, K. and Bellatreche, L. La contribution des structures d'optimisation non redondantes dans la conception physique des entrepôts de données. Proceedings of 8th International Symposium on Programming and Systems (ISPS'2007), pp. 173-184, Mai, 2007.
[WI-6] Bellatreche, L., Giacometti, A., Marcel, P., Laurent, D. and Mouloudi, H., A Personalization Framework for OLAP Queries. Proceedings of ACM 8th International Workshop on Data Warehousing and OLAP (DOLAP'05), pp. 9-18, Bremen, Germany.
[WI-7] Bellatreche, L., Karlapalem, K. Mohania, M., Schneider, M. What Can do Partitioning for Data Warehouses? Proceedings of the International Database Engineering and Applications Symposium (IDEAS'2000), IEEE Computer Society Press, pp. 437-445, November 2000, Japan.
[WI-8] Bellatreche, L. and Karlapalem, K. Logical and Physical Design of a Data Warehouse'', in EDBT Ph.D**.** Workshop, March 2000, Konstanz, Germany.
[WI-9] Bellatreche, L., Karlapalem, K. and Mohania, M. OLAP Query Processing for Partitioned Data Warehouses. Proceedings of the International Symposium on Database Applications in Non-Traditional Environments, pp. 35-42, November 1999, IEEE Computer Society Press, Japan.
[WI-10] Bellatreche, L. Partitioning Strategies in Distributed Object-Oriented Database Systems. Proceedings of the 3rd International Symposium Programming and Systems (ISPS'97), pp. 360-373, Algiers, April 1997.
[WI-11] Bellatreche, L., An Algorithm for Vertical Fragmentation in Distributed Object Database Systems. Proceedings of the 2nd International Baltic Workshop on Databases and Information Systems (BALT'96), pp. 65-74, Tallin, June 1996.

[WI-12] Bellatreche, L., Simonet, A. and Simonet, M. An Algorithm for Vertical Fragmentation in Distributed Object Database Systems with Complex Attributes and Method. Proceedings of 7th International Conference on Databases and Expert Systems, (DEXA'96), pp. 15-21, Zurich, September 1996, IEEE Computer Society.

7.8. Conférences nationales avec comité de lecture

[CN-1] Benkrid, S. and Bellatreche, L. Une démarche conjointe de fragmentation et de placement dans le cadre des entrepôts de données parallèles, in 5ème Journées Francophones sur les Entrepôts de Données et analyse en Ligne (EDA'09), Revue des Nouvelles Technologies de l'Information, RNTI B-5, pp. 91-106, Montpellier, Juin 2009. Sélectionné pour la revue Technique et Science Informatiques (TSI).

[CN-2] Khouri, S., Bellatreche, L. and Fankam, C. SISROM2C : Un outil de modélisation conceptuelle à base ontologique d'un entrepôt de données. 4ème journées Francophones sur les Entrepôts de Données et analyse en Ligne (EDA'09), Revue des Nouvelles Technologies de l'Information, RNTI B-5, pp. 123-138, Montpellier, Juin 2009.

[CN-3] Bellatreche, L., Boukhalfa, K. and Caffiau, S. ParAdmin: Un Outil d'Assistance à l'Administration et Tuning d'un Entrepôt de Données. 4ème journées Francophones sur les Entrepôts de Données et analyse en Ligne (EDA'08), Revue des Nouvelles Technologies de l'Information, RNTI-B-4, pp. 99-114, Juin, 2008

[CN-4] Boukhalfa, K., Bellatreche, L. and Richard, P. Fragmentation primaire et dérivée : étude de complexité, algorithmes de sélection et validation sous Oracle10g. 4ème journées Francophones sur les Entrepôts de Données et analyse en Ligne (EDA'08), Revue des Nouvelles Technologies de l'Information, RNTI-B-4, p. 123-140, Juin, 2008

[CN-5] Fankam, C., Jean, S., Pierra,G. and Bellatreche, L. Enrichissement de l'architecture ANSI/SPARC pour expliciter la sémantique des données: une approche fondée sur les ontologies. 2ème Conférence Francophone sur les Architectures Logicielles (CAL'08), Revue des Nouvelles Technologies de l'Information, RNTI-L-2, pp. 47-61, Montréal, Mars 2008.

[CN-6] Abdelmalek, A. Elberrichi, Z., Bellatreche, L., Simonet, M. and Malki, M., Classification Automatique Non supervisée de Documents Textuels basés sur Wordnet. 8èmes journées Extraction et Gestion des Connaissances (EGC'08) Sophia-Antipolis, France, Revue des Nouvelles Technologies de l'Information, pp. 227-228.

[CN-7] Dehainsala H., Pierra G., Bellatreche L. and Aït Ameur Y. Conception de bases de données à partir d'ontologies de domaine : Application aux bases de données du domaine technique. 1ères Journées Francophones sur les Ontologies (JFO'07), pp. 215-230, Octobre, 2007.

[CN-8] Necir H., Bellatreche L. and Missaoui R. DynaClose : Une approche de fouilles de données pour la sélection des index de jointure binaires dans les entrepôts de données, 3ème journées Francophones sur les Entrepôts de Données et analyse en Ligne (EDA'07), Revue des Nouvelles Technologies de l'Information, RNTI-B-3, pp. 83-98, Juin, 2007.

[CN-9] Pierra G., Aït Ameur, Y., Bellatreche, L. Dehainsala, H., Jean, S., Fankam, C. and Xuan, D. N. Données à base ontologique: gestion, interrogation, intégration, Premières journée francophone sur les ontologies (papier invitée), 2007.

[CN-10] Belaid, N., Bellatreche, L., Ait Ameur, Y. and Pierra, G. Intégration de sources à base ontologique : architecture en réseau VS architecture en étoile. Plate-Forme AFIA: Atelier Thématique GDR I3 sur Ontologies et Gestion de l'hétérogénéité sémantique (OGHS), Juillet, 2007, pp. 9-20.

[CN-11] Ouaret, Z., Bellatreche, L. and Boussaid, O., Conceptual and Logical designs of XML Warehouses. Troisième Atelier sur les Systèmes Décisionnels (ASD'08), pp. 137-149, Octobre, Maroc.
[CN-12] Dung, N. X., Bellatreche, L. and Pierra G., Un modèle a base ontologique pour la gestion de l'évolution asynchrone des Entrepôts de Données. Conférence Internationale de Modélisation, Optimisation et Simulation des Systèmes : Défis et Opportunités (MOSIM'06), pp. 1682-1691, Rabat, Maroc, 3 - 5 avril, 2006.
[CN-13] Bellatreche, L. Boukhalfa, K., Combinaison des algorithmes génétique et de recuit simulé pour la conception physique des entrepôts de données, in the Proceedings of the Congrès d'Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'2006), pp. 673-686, May 2006.
[CN-14] Mouloudi, H. Bellatreche, L. Giacometti A. and Marcel, P., Personalization of MDX Queries. Bases de Données Avancées (BDA'06), Octobre, 2006.
[CN-15] Dung, N. X., Bellatreche, L. and Pierra G., Ontology Evolution and Source Autonomy in Ontology-based Data Warehouses, 2ème Journée sur Entrepôts de Données et Analyse en Ligne (EDA'06), Revue des Nouvelles Technologies de l'Information, pp. 55-76, Juin 2006.
[CN-16] Bellatreche, L. and Boukhalfa, K., Une répartition statique et dynamique de l'espace entre les vues matérialisées et les index dans les entrepôts de données, International Symposium on Programming and Systems (ISPS'05), Alger, 2005
[CN-17] Bellatreche, L. and Boukhalfa, K., La fragmentation dans les entrepôts de données : une approche basée sur les algorithmes génétiques, 1ère Journée sur Entrepôts de Données et Analyse en Ligne (EDA'05), Revue des Nouvelles Technologies de l'Information (EDA'2005), RNTI-B-1, pp. 141-160, Juin, 2005, Lyon.
[CN-18] Bellatreche, L., Giacometti, A., Marcel, P., Laurent, D. and Mouloudi, H., A Framework for Combining Rule-Based and Cost-Based Approaches to Optimize OLAP Queries, 1ère Journée Francophone sur les Entrepôts de Données et Analyse en Ligne (EDA'2005). Revue des Nouvelles Technologies de l'Information, pp. 177-196, 2005, Lyon.
[CN-19] Pierra, G., Dehainsala,H., Ait Ameur, Y., Bellatreche, L., Chochon, J. and Mimoume, M., Base de Données à Base Ontologique : le modèle OntoDB, 20èmes Journées Bases de Données Avancées (BDA'2004), 2004, pp. 263-286, Montpellier.
[CN-20] Bellatreche, L., Pierra G., Dung, N. X. and Hondjack, D., Integration de sources de données autonomes par articulation a priori d'ontolgie, in the proceedings of the Congrès d'Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID'2004), pp. 283-298, May 2004
[CN-21] Bellatreche, L. Techniques d'optimisation des requêtes dans les data warehouses, in proceedings of the 6th International Symposium on Programming and Systems (ISPS'2003), pp. 81-98, May 2003, Algeria.
[CN-22] Bellatreche, L. Une Méthodologie de Fragmentation dans les Entrepôts de Données, in the 18ème Congrès d'Informatique des Organisations et Systèmes d'Information et de Déçision (INFORSID'0**0**), pp. 245-260, Lyon, France
[CN-23] Bellatreche, L., Karlapalem, K. and Simonet, A. Query Optimization using Horizontal Class Partitioning in Object Oriented Databases, in the 16ème Congrès d'Informatique des Organisations et Systèmes D'Information et de Décision (INFORSID'98), pp. 405-422, May 12-15, 1998, Montpellier, France.

7.9 Rapports techniques

[RT1] Bellatreche, L. and K. Karlapalem. Spy vs. Spy: On Efficient Space Distribution Among Materialized Views and Indices in Data Warehousing Environments", Technical Report, HKUST-CS99-10, Department of Computer Science, HKUST, July, 1999.

[RT3] L. Bellatreche, K. Karlapalem and Q. Li, Algorithms for Graph Join Index Problem in Data Warehousing Environments, Technical Report, HKUST-CS99-07, Department of Computer Science, HKUST, March, 1999.

7.10. Séminaires, conférences invitées

- Juillet 2009 : "*Les Ontologies dans les Systèmes d'Information*" : Laboratoire de Recherche sur l'Information Multimédia (LARIM) - Université du Québec en Outaouais Canada
- Juin 2007 : ''*Perspectives de recherche dans les entrepôts de données*" : Laboratoire d'informatique Université de Tours.
- Juin 2008: "*From Designing Data Warehouse Applications to Tuning*", International Conference on Web and Information Technologies (ICWIT'08) Conférence Invitée, Sidi Bel Abbès, Algérie.
- Mai 2006 : "*La conception physique des data warehouses*", Conférence sur l'Informatique et ses Applications, papier invité, Saida, Algérie.
- Août 2001 : "*Horizontal Fragmentation in Very Large Databases*", Departement of Computer Science, Purdue University, U.S.A.

8. RAYONNEMENT SCIENTIFIQUE

8.1. Collaboration

Durant mes activités de recherche, j'ai eu l'opportunité de collaborer tant avec des universitaires qu'avec des industriels.

8.1.1. Milieu industriel

- Institut Français du Pétrole (IFP) (Contacts : Jean François Raynaud)
- IBM Inde (Contact: Prof. Mukesh Mohania).
- EADS (contact: Anne Monceaux)

8.1.2. Milieu universitaire

8.1.2.1 National

- Laboratoire d'Informatique de Tours (contacts: A. Giacometti, P. Marcel, V. T'KINDT).
- Laboratoire d'informatique de La Rochelle (contact : J. Y. LAFAYYE)
- Laboratoire LIRMM (contact: Z. BELLAHSENE)
- Laboratoire TICM-IMAG (contact: M. SIMONET)

- Laboratoire ERIC-Unversité Lyon 2 (contact: O. BOUSSAID)

8.1.2.2 International

- Département d'Informatique de City University of Hong Kong – Chine (contact : Qing LI)
- Nanyang Technological University – Singapore (Contact :V. GOPALKRISHNAN)
- Laboratoire de Recherche sur l'Information Multimédia – Canada (Contact R. MISSAOUI)
- Département d'Informatique – Purdue University (contact : B. K. BHARGAVA)
- International Institute of Information Technology India (Contact: K. KAMALAKAR)
- Université des Sciences et de la Technologie Houari Boumediene (USTHB) (contacts : H. DRIAS, Z. ALIMAZIGHI)
- Ecole Nationale d'Informatique (contact : T. TEBIBEL)
- Université de Sidi Bel Abbès (contact : M. MALKI)
- Institute of Computing Science Poznań University of Technology Pologne (contact: R. WREMBEL).
- Institute of High Performance Computing and Networking, Italian National Research Council, Italie (contact : A. CUZZOCREA).
- Multimedia, InfoRmation systems and Advanced Computing Laboratory – Sfax, Tunisie (contact: Faiez Gargouri)

8.2. Participation à des groupes de recherche

- GDR I3 du CNRS : Modèles de Données et Conception de Systèmes d'Information
  - GT 3.1 : Infrastructures de médiation – Namur, 2006
- Groupe Données de la Fédération Pôle Régional de Recherche en Images, Données et Systèmes (Prides)

8.3. Organisation de manifestations nationales et internationales

- EDA'07: 3èmes journées francophones sur les Entrepôts de Données et Analyse en ligne à Poitiers Juin 2007.
- JFO'09 : 3èmes journées francophones sur les Ontologies (JFO'09), à Poitiers décembre 2009.
- ISOLA: ISoLA Workshop On Leveraging Applications of Formal Methods, Verification and Validation.
- DIE'03: Data integration in Engineering - Special track in 10th ISPE International Conference on Concurrent Engineering: Research and Applications (CE'2003)
- ISO: 41st International meeting of ISO TC184/SC4 and Working Groups "Industrial Data", Futuroscope, 26-31 October 2003

8.4. Présidence de comités de programme de conférences nationales

- EDA'07: 3èmes journées francophones sur les Entrepôts de Données et Analyse en ligne à Poitiers Juin 2007.
- JFO'09 : 3èmes journées francophones sur les Ontologies (JFO'09), à Poitiers décembre 2009.
- DASFAA'08 : Regional Chair : Europe

8.5. Comité de pilotage

- ICWIT : International Conference on Web and Information Technologies

8.6. Mobilité

- Juillet – Août 2009: Séjour de recherche de deux mois
    - Lieu : Laboratoire de Recherche sur l'Information Multimédia (LARIM) - Université du Québec en Outaouais Canada (http://larim.uqo.ca/index.html)
    - Responsable : Prof Rokia MISSAOUI.
    - Thème : Treillis de Galois, modélisation et optimisation de traitements
- Juin – Aout 2001 : Séjour de recherche de trois mois
    - Lieu : RAID Laboratory- Computer Science Department Purdue University U.S.A (http://raidlab.cs.purdue.edu/)
    - Responsable : Prof. Bharat K. BHARGAVA
    - Thème : Data mining et sécurité
- Février 97 –Août 99 : Séjour de recherche
    - Lieu : Database Group : Department of Computer Science Hong Kong University of Science & Technology (http://db.cse.ust.hk/)
    - Responsable : Prof. Kamalakar KARLAPALEM
    - Thème : Partitionnement, conception de bases de données réparties, entreposage de données.

8.7. Editeur de numéros spéciaux de revues internationales

- Distributed and Parallel Database Journal Springer : numéro spécial sur la conception physique dans les entrepôts de données 2008
- Data & Knowledge Engineering Journal – Elsevier, Spécial issue sur la contribution des ontologies dans la conception des systèmes d'information avancés 2009.

8.8. Membre de comités de programme de conférences nationales et internationales

8.8.1. Conférences nationales

- 2009 : WWS: Workshop sur les Services Web dans les Systèmes d'Informations
- 2005, 2006, 2007, 2008, 2009 : EDA: Journées Francophones sur les Entrepôts de Données et analyse en Ligne
- 2007, 2008, 2009 : JFO : Journées Francophones sur les Ontologies
- 2006, 2008 : INFORSID : Congrès d'Informatique Des Organisations et Systèmes d'Information et de Décision
- ASD: 2006, 2007, 2008, 2009: Atelier sur les Systèmes Décisionnels.

8.8.2. Conférences internationales

- 2010 : ICA3PP : 10th International Conference on Algorithms and Architecture for Parallel Processing.
- 2009: CIKM : ACM Conference on Information and Knowledge Management

- 2007, 2008, 2009 : DASFAA: International Conference on Database Systems for Advanced Applications
- 2009: COMAD: International Conference on Management of Data
- 2010, 2009, 2008 : DEXA : International Conference on Databases and Expert Systems
- 2005, 2007, 2008, 2009: DAWAK: International Conference on Data Warehousing and Knowledge Discovery.
- 2007: IEEE-DEST: IEEE Conference on Digital Ecosystems and Technologies
- 2007: WISE: International Conference on Web Information Systems Engineering
- 2007, 2008, 2009: ICE-B International Joint Conference on e-Business
- 2005, 2006, 2007, 2008: GREP: International Workshop on Data Management in Global Data Repositories
- 2005: BPM: International Conference on Business Process Management
- 2003 : CE : ISPE International Conference On Concurrent Engineering
- 2007, 2009: ISPS: International Symposium Programming and Systems
- 2005: ICDCIT : International Conference on Distributed Computing & Internet Technology
- 2006: ICEBE: International Conference on e-Business Engineering
- 2009: WISM: International Conference on Web Information Systems and Mining
- 2009: TWOMDE : International Workshop on Transforming and Weaving Ontologies in Model Driven Engineering
- 2009: MEDWa : Workshop On Managing Evolution of Data Warehouses

8.9. Relecteur d'articles soumis pour publication dans des revues internationales

- Data & Knowledge Engineering Journal (DKE) - Elsevier 2009.
- Distributed and Parallel Database Journal, Springer 2008.
- International Journal of Business Intelligence and Data Mining, InderScience 2008.
- Integrated Computer-Aided Engineering Journal 2008.
- VLDB Journal, Springer 2005.
- ACM Transactions on Internet Technology - Special issue on Semantic Web Services: Issues, Solutions, and Applications 2006.
- International Journal of Product Lifecycle Management 2007.
- International Journal of Biomedical Engineering and Technology 2007.

8.10. Relecteur externe d'articles soumis à des conférences internationales ou à des revues nationales

- EDBT'2009: International Conference on Extending Database Technology
- DASFAA'99: International Conference on Database Systems for Advanced Applications
- VLDB'98: International Conference on Very Large Data Bases
- TSI : Technique et Science Informatiques, 2008.

8.11. Présidence de sessions de conférences

- DOLAP'09: ACM Twelfth International Workshop on Data Warehousing and OLAP
  - *Session: ETL 1.*
- *DAWAK'09:* 11th International Conference on Data Warehousing and Knowledge Discovery

  - *Session: OLAP Recommendation*
- DEXA'09: 20th International Conference on Database and Expert Systems Applications
  - *Session: Semantic Web and Ontologies.*
- DEXA'07: 18th International Conference on Database and Expert Systems Applications
  - *Session: Query Processing and Optimisation*: *Applications*
- JFO'07 : Journées Francophones sur les Ontologies (JFO'07) – Tunisie
- DAWAK'06: International Conference on Data Warehousing and Knowledge Discovery
  - *Session: Materialized views*

9. CONTRATS DE RECHERCHE

- 2005-2009
  - Participation au projet ANR RNTL Environmental Web Ontology Knowledge Hub (E-wok hub).
    - Partenaires : INRIA (Sophia Antipolis), EADS (Val de Reuil), ENSMP (Paris), IFP (Paris), BRGM (Orléans), LISI (Poitiers), CRITT Informatique (Poitiers)
    - Objectif du projet : Conception et développement d'un ensemble de serveurs Web sémantiques et portails communicants appelés Hub et proposition des interfaces d'accès à ces Hubs par des services Web.
    - Mon rôle : support du raisonnement numérique dans les bases de données à base ontologique.
    - Montant : 134 000 €.

- 2006-2010
  - Participation au Projet ANR Dafoe (Differential and Formal Ontology Editor) : http://dafoe4app.fr/
    - Partenaires: INSERM (Paris), ENST/GET (Paris), IRIT (Toulouse), LIPN (Paris), LISI (Poitiers), Mondeca (Paris), Supelec (Saclay), UTC (Compiègne)
    - Objectif : Proposition d'une méthode complète associée à une plate-forme technique pour concevoir des ontologies à partir du texte.
    - Mon rôle : conception et suivi du développement de la plateforme de conception des ontologies à partir du texte.
    - Montant : 71 000 €.

- 2005-2007
  - ACI MECANO
    - Modélisation et Exploitation de Contenu Artistique Numérique à l'aide d'Ontologies : Application à la valorisation de masses de données.
    - Laboratoires d'appui : Laboratoire Signal Image Communications de l'Université de Poitiers, LISI/ENSMA et Centre d'études Supérieures de Civilisation Médiévale (CESCM).
    - Mon Rôle : Stockage des images dans une base de données à base ontologique
    - Montant du projet : 6 000 €
    - Master 2 recherche associé à ce projet : Dalal SHOUFAN.

- 2008-2009
  - Contribution des techniques de Recherche Opérationnelle à l'Optimisation des Entrepôts de Données Relationnels.
    - Responsable de projet.

- Laboratoires d'appui : Laboratoire d'Informatique de Tours (équipe Ordonnancement et Conduite) et le laboratoire LISI.
- Mon rôle : proposition de nouveaux algorithmes de fragmentation horizontale.
- Montant du projet : 2 500 €

10. RESPONSABILITES COLLECTIVES

- Membre de la Commission d'Expertise Scientifique (C.E.S.) à l'université de Poitiers.
- Membre du comité de sélection pour le recrutement de poste d'un maître de conférences à l'université de Poitiers (MCF 27-0655).
- Membre de Commission d'audition pour un recrutement de poste de maître de conférences à l'université de Tours (MCF 27-1345).
- Membre de l'agence nationale de recherche algérienne.

11. ACTIVITES D'ENSEIGNEMENT

Depuis 1999, j'ai assuré et assure des enseignements variés en qualité d'ATER, de professeur chercheur, et de maître de conférences. Cette diversité se traduit en termes de:

1. contenu : programmation, bases de données, conception des systèmes d'informations, bases de données avancées, administration de bases de données, systèmes d'exploitation, Ingénierie Web, UML, Programmation Objet, Programmation Fonctionnelle, système d'exploitation, cryptographie, complexité des algorithmes, Architectures des ordinateurs, informatique décisionnelle, structures d'optimisation avancées, data mining, etc. ;
2. nature des enseignements : cours magistraux, travaux dirigés, travaux pratiques, encadrements de projets, bureaux d'études, suivi des étudiants en stage d'entreprise ;
3. public : premier, second et troisième cycles, élèves ingénieurs, double compétence informatique et biologie, et DEUG, licence professionnelle, étudiants à l'étranger (pays francophones et anglophones), non informaticiens.
4. volume horaire.

11.1. Enseignements dispensés

11.1.1. En Qualité de Maître de Conférences

Intitulé	Cycle	Public	Forme	VH Annuel[2]	S[3]	Années
Administration de bases de données	3	DESS Informatique (CLABD) puis Master 2	C[4], TD[5]	36h/an	P[6], T[7]	2003-Présent
Informatique décisionnelle	3	Master 2 IUP[8] Génie physiologique et	C, TD	36h/an	T	2008-Présent

[2] VH : Volume Annuel en heures ETD
[3] S : Support
[4] C : Cours
[5] TD : Travaux Dirigés
[6] P : Polycopie
[7] T : Copie de Transparents
[8] IUP : Institut universitaire professionnalisé

		Informatique				
Bases de Données Avancées	3	DESS + Master 2 Risque informatique IRIAF	C, TD	72h/an	P, T	2003-Présent
Bases de Données Avancées	2	Master 1 Informatique	C, TD	36h/an	P, T	2007-présent
Conception de Systèmes d'Informations	2	Master 1 Génie physiologique et Informatique	C, TD, TP[9]	32h/an	T	2002-Présent
Bases de Données	1	Licence 3 Informatique	C, TD, TP	54h/an	P, T	2002-Présent
Administration Système	2	Master 1 Administration	C, TP	36h/an	T	2003-2004
Merise	2	Maîtrise puis Master 1 IUP Génie physiologique Informatique	C, TD, Projet	32h/an	T	2002-Présent
Bases de Données & Web	1	Licence Professionnelle Technologies Logicielles pour le Web et les Terminaux Mobiles	C, TD,	32h/an	P, T	2007-Présent
Bases de Données	1	IUP de Chimie	C, TD, TP	36h/an	P, T	2002-2008
Conception de Bases de Données	2	Maitrise des Sciences et Techniques IRIAF	C, TD, TP	48h/an	P, T	2003-2005
Méthodes de Conception	2	Maîtrise Des Sciences et Techniques IRIAF	C, TD, Projet	48h/an	T	2003-2005
Informatique Générale	1	Licence 1 Faculté de Droit	TP	30h/an		2003-présent
Encadrement Projets	2ème & 3ème	Maîtrise & Master 2 Informatique				
Merise	2ème	Master 1 Faculté de Droit	C, Projet	22h/an	T	2008-Présent

Tableau 1 : Activités d'Enseignement en Qualité de Maître de Conférences

11.1.2. Enseignements dispensés à l'ENSMA

- **2002-2003 :** UML 3ème cycle.

[9] TP : Travaux Pratiques

- 3ème année élèves ENSMA, option 3. 15 élèves (cours) (10h/an ETD) Avec copie de transparents distribués aux élèves.

- **2004-2008 :** Suivi de Projet ADA. 2ème cycle.
  - 1ère année ingénieur ENSMA.

- **2007 :** Encadrement de Stages de 2ème cycle.
  - 2ème année ingénieur ENSMA.

11.1.3. Enseignements dispensés à l'Institut Supérieur de l'Aéronautique et de l'Espace – Toulouse

- **2007-Présent :** Informatique décisionnelle. 3ème cycle
  - Master International en Ingénierie Système. Cours (6h/an ETD). Cours en anglais.
  - Avec copie de transparents en anglais distribués aux élèves.
  - Entrepôts de données, modélisation multidimensionnelle, business intelligence.

11.1.4. Enseignements dispensés dans les Universités Algériennes

Mon laboratoire, le LISI, a porté un accord CMEP avec l'Université de Science et Technologie de Houari Boumediène (USHTB) et l'Ecole Nationale Supérieure en Informatique (ex. INI). Cette dernière a mis en place en Algérie en partenariat avec plusieurs universités une école doctorale. J'interviens depuis 2004 dans cette école.

- **2004-Présent :** Bases de données avancées et initiation à la recherche. 3ème cycle
  - Universités algériennes (écoles doctorales) : Ecole Nationale Supérieure d'Informatique – Alger (ex. INI), Université de Boumerdès, Université de Sidi Bel Abbès (mon université d'origine), Université de Sétif, Université de Saida. 60 étudiants (Cours, TD, 50 h/an ETD)
  - Avec copie de transparents distribués aux étudiants.

- **Février 2004 :** Sécurité UNIX
  - Ingénieur de l'entreprise algérienne de recherche, d'exploitation, de transport par canalisation, de transformation et de commercialisation des hydrocarbures : Sonatrach. 20 étudiants (Cours, TP 36 h/an ETD)
  - Avec copie de transparents distribués aux étudiants.

11.1.5. Enseignements dispensés en qualité de Professeur Chercheur à l'Ecole IMERIR

Intitulé	Cycle	Public	Forme	VH Annuel	S	Nombre Etudiants
Langage C	2	1ère année Ingénieur	C, TP	20h/an	P, T	64
Ingénierie Web	2	1ère année Ingénieur	C, TP	24H/an	T	64
Bases de données et Systèmes	2	1ère année Ingénieur	C, TD, TP	60h/an	P, T	64
Java & Réseau	2	2ème Année ingénieur	C, TD, TP	52h/an	P, T	60
Bases de données	3	3ème année ingénieur	C, TD, TP	36h/an	T	60

avancées						
XML & Intégration Système	3	3ème année ingénieur	C, TD, TP	16h/an	T	60
Encadrement Projet	2 & 3	1ère, 2ème & 3ème année ingénieur		48h/an		Groupe de 4 étudiants

Tableau 2 : Activités d'Enseignement en Qualité de Professeur Chercheur à l'Ecole IMERIR

11.1.6. Enseignements dispensés en qualité d'ATER à l'Université Joseph Fourier

Intitulé	Cycle	Public	Forme	VH Annuel	S	Nombre Etudiants
Introduction à la Programmation	1	1ère année DEUG : MIAS, SMB	C, TD et TP	48h/an		28
Bases de données & systèmes relationnelles	2	2ème année Ingénieur RICM[10]	TD	18h/an		25
Complexité algorithmique & Cryptographie	2	2ème an. RICM	TD	18h/an		25
Encadrement projet	3	D.E.S.S GI[11]		8 heures		10
Système & Architecture	2	2ème an. RICM & Licence ESI[12]	TD	36h/an	T	70

Tableau 3 : Activités d'Enseignement en Qualité d'ATER à l'Université Joseph Fourier

11.1.7. Enseignements dispensés en qualité de demi ATER à l'Université de Savoie

Intitulé	Cycle	Public	Forme	VH Annuel	S	Nombre Etudiants
Bureautique	1	1ère année DEUG : MIAS, MASS, SV	TP	48h/an		28
Algorithmique	1	1ère année DEUG : MIAS, MASS, SV	TD, TP	18h/an		25
Programmation & Bureautique	1	1ère année DEUG SV	TP	36h/an		25

Tableau 4 : Activités d'Enseignement en Qualité de demi ATER à l'Université Savoie

11.1.8. Enseignements dispensés en qualité de chercheur à l'Université de Hong Kong

- 1998: Distributed Data and Knowledge Bases. Ph.D. Course. 15 étudiants (15/an ETD).

[10] RICM : Ecole d'Ingénieur Polytechnique de l'université de Grenoble 1 : Réseaux Informatiques et Communication Multimédia
[11] GI : Génie Informatique Option Système d'Information
[12] ESI : Expert en Systèmes Informatiques

- Hong Kong University of Science and Technology. *Cours en anglais*.

11.2. Responsabilités administratives liées à l'enseignement

- Participation à l'élaboration de l'habilitation de Master Sécurité dans les Systèmes d'Information de Niort.
- Responsable du module de bases de données avancées Master 2 « Informatique Télécommunication Professionnel et de Recherche »
- Responsable du cours de bases de données Licence 3 Génie Physiologique et Informatique
- Responsable du cours informatique décisionnelle Master 2 Génie Physiologique et Informatique
- Elaboration des programmes d'enseignement des bases de données et systèmes d'information pour les habilitations de :
  - Master Informatique Télécommunication
  - Master 2 Management des Risques Informationnels et Industriels (MRII) Niort
  - IUP Génie Physiologique et Informatique.
- Participation aux jurys de Master
- Conférence scientifique pour les étudiants de Master 1 portant sur les travaux de recherche au laboratoire LISI
- Tuteurs d'étudiants de Master 2
- Jury de Master/jury de Baccalaureat

11.3. Supports de cours réalisés

Les supports de cours suivants on été réalisés sous forme de polycopiés et transparents et publiés à l'université de Poitiers. Ils sont distribués aux étudiants inscrits aux cours correspondants.

L. Bellatreche « Bases de données »
L. Bellatreche « Entrepôts de données »
L. Bellatreche « Merise »
L. Bellatreche « PHP & bases de données »
L. Bellatreche « XML »
L.Bellatreche « Administration systèmes »
L. Bellatreche « Informatique décisionnelle »
L. Bellatreche « Data mining »
L. Bellatreche « Structures d'optimisation de requêtes avancées »

12. RECOMMANDATIONS

Suggestion de personnalités auxquelles des avis sur mon dossier ou des lettres de recommandation peuvent être demandés.

En France	A l'étranger
Guy PIERRA Professeur ENSMA – Futuroscope Poitiers pierra@ensma.fr	Karlapalem KAMALAKAR International Institute of Information Technology Hyderabad INDIA kamal@iiit.ac.in
Yamine AIT AMEUR Professeur ENSMA – Futuroscope Poitiers yamine@ensma.fr	Mukesh MOHANIA IBM India Research Laboratory mkmukesh@in.ibm.com
Michel SCHNEIDER Professeur Université Clermont Ferrand 2 schneider@limos.fr	Qing LI City University of Hong Kong, Chine itqli@cityu.edu.hk
Michel SIMONET Chargé de Recherches CNRS – Université de Joseph Fourier Grenoble michel.simonet@imag.fr	Rokia MISSAOUI University of Québec en Outaouais, Canada Rokia.missaoui@uqo.ca
Arnaud GIACOMETTI Professeur Université de Tours Arnaud.Giacometti@univ-tours.fr	Bharat K. BHARGAVA Purdue University, U.S.A. bb@cs.purdue.edu
Dominique LAURENT Professeur, Université Cergy Pontoise dominique.laurent@dept-info.u-cergy.fr	Amit SHETH Wright State University, U.S.A. amit.sheth@wright.edu

Bibliographie

[1] R. Agrawal, A. Somani, and Y. Xu. Storage and querying of e-commerce data. In *VLDB*, pages 149–158, 2001.

[2] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, and M. Syamala. Database tuning advisor for microsoft sql server 2005. In *Proceedings of the International Conference on Very Large Databases*, pages 1110–1121, 2004.

[3] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite: Managing voluminous RDF description bases. In *SemWeb*, 2001.

[4] K. Aouiche, O. Boussaid, and F. Bentayeb. Automatic Selection of Bitmap Join Indexes in Data Warehouses. *7th International Conference on Data Warehousing and Knowledge Discovery (DAWAK 05)*, August 2005.

[5] P. M. G. Apers. Data allocation in distributed database systems. *ACM Transactions on database systems*, 13(3):263–304, 1988.

[6] Y. Arens and C. A. Knoblock. Sims: Retrieving and integrating information from multiple sources. *Proceedings of the International Conference on Management of Data (SIGMOD'1993)*, pages 562–563, May 1993.

[7] B. Bebel, J. Eder, C. Koncilia, T. Morzy, and R. Wrembel. Creation and management of versions in multiversion data warehouse. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 717–723, New York, NY, USA, 2004. ACM Press.

[8] L. Bellatreche. Logical and physical design in data warehousing environments. *in proceedings of International Conference on Extending Database Technology (EDBT) PhD Workshop*, March 2000.

[9] L. Bellatreche. Une méthodologie pour la fragmentation dans les entrepôts des données. *in 18th Congrès d'Informatique Des Organisations et Systèmes d'Information et de Décision (INFORSID'2000), Lyon*, pages 245–260, May 2000.

[10] L. Bellatreche. Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique d'un entrepôts de données. Phd. thesis, Blaise Pascal University, France, 2000.

[11] L. Bellatreche. *Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction*. Information Science Reference, 2009.

[12] L. Bellatreche. New trends in physical data warehouse design. *Distributed and Parallel Database Journal, Springer*, 25(1-2):1–3, February 2009.

[13] L. Bellatreche. Optimization and tuning in data warehouses. In *Encyclopedia of Database Systems (EDS), Edited by Ling Liu and Tamer Özsu*, pages 69–76. Springer, 2009.

[14] L. Bellatreche and S. Benkrid. A joint design approach of partitioning and allocation in parallel data warehouses. In *11th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'09)*, pages 99–110, 2009.

[15] L. Bellatreche and K. Boukhalfa. An evolutionary approach to schema partitioning selection in a data warehouse environment. *Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2005)*, pages 115–125, August 2005.

[16] L. Bellatreche and K. Boukhalfa. La fragmentation dans les entrepôts de données : une approche basée sur les algorithmes génétiques. In *1ème Journée Francophone sur les Entrepôts de données et l'Analyse en ligne (EDA'05), Revue des Nouvelles Technologies, RNTI-B-1*, pages 141–160, 2005.

[17] L. Bellatreche, K. Boukhalfa, and H. I. Ab-

dalla. Saga: A combination of genetic and simulated annealing algorithms for physical data warehouse design. In *23rd British National Conference on Databases (BNCOD'06)*, pages 212–219, July 2006.

[18] L. Bellatreche, K. Boukhalfa, and Z. Alimazighi. Simulph.d.: A physical design simulator tool. In *20th International Conference on Database and Expert Systems Applications (DEXA'09)*, pages 263–270, 2009.

[19] L. Bellatreche, K. Boukhalfa, and S. Caffiau. Paradmin : Un outil d'assistance à l'administration et tuning d'un entrepôt de données. In *4ème Journée Francophone sur les Entrepôts de données et l'Analyse en ligne (EDA'08), Revue des Nouvelles Technologies de l'Information, RNTI-B-4*, pages 99–114, 2008.

[20] L. Bellatreche, K. Boukhalfa, and M. K. Mohania. Pruning search space of physical database design. In *18th International Conference On Database and Expert Systems Applications (DEXA'07)*, pages 479–488, 2007.

[21] L. Bellatreche, K. Boukhalfa, and P. Richard. Data partitioning in data warehouses: Hardness study, heuristics and oracle validation. In *International Conference on Data Warehousing and Knowledge Discovery (DaWaK'2008)*, pages 87–96, 2008.

[22] L. Bellatreche, K. Boukhalfa, and P. Richard. Referential horizontal partitioning selection problem in data warehouses: Hardness study and selection algorithms. *International Journal of Data Warehousing and Mining*, 5(4):1–23, March 2009.

[23] L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent. A framework for combining rule-based and cost-based approaches to optimize olap queries. In *1ème Journée Francophone sur les Entrepôts de données et l'Analyse en ligne (EDA'05), Revue des Nouvelles Technologies de l'Information, RNTI-B-1*, pages 177–196, 2005.

[24] L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent. A personalization framework for olap queries. In *ACM 8th International Workshop on Data Warehousing and OLAP (DOLAP'05)*, pages 9–18, 2005.

[25] L. Bellatreche, K. Karlapalem, and G. B. Basak. Query-driven horizontal class partitioning in object-oriented databases. *in 9th International Conference on Database and Expert Systems Applications (DEXA'98), Lecture Notes in Computer Science 1460*, pages 692–701, August 1998.

[26] L. Bellatreche, K. Karlapalem, and Q. Li. Derived horizontal class partitioning in oodbss: Design strategy, analytical model and evaluation. In *17th International Conference on the Entity Relationship Approach (ER'98)*, pages 465–479, November 1998.

[27] L. Bellatreche, K. Karlapalem, and Q. Li. An iterative approach for rules and data allocation in distributed deductive database systems. In *in the 7th International Conference on Information and Knowledge Management (CIKM'98)*, pages 356–363, November 1998.

[28] L. Bellatreche, K. Karlapalem, and Q. Li. Algorithms for graph join index problem in data warehousing environments. Technical Report HKUST-CS99-07, Hong Kong University of Science & Technology, March 1999.

[29] L. Bellatreche, K. Karlapalem, and Q. Li. Evaluation of indexing materialized views in data warehousing environments. In *Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2000)*, pages 57–66, September 2000.

[30] L. Bellatreche, K. Karlapalem, and M. Mohania. What can partitioning do for your data warehouses and data marts. In *Proceedings of the International Database Engineering and Application Symposium (IDEAS'2000)*, pages 437–445, September 2000.

[31] L. Bellatreche, K. Karlapalem, and M. Mohania. Some issues in design of data warehousing systems. In *Developing Quality Complex Data Bases Systems: Practices, Techniques, and Technologies*, pages 125–172. Idea Group Publishing, 2001.

[32] L. Bellatreche, K. Karlapalem, and M. Schneider. On efficient storage space distribution among materialized views and indices in data warehousing environments. In *Proceedings of the International Conference on Information and Knowledge Management (ACM CIKM'2000)*, pages 397–404, 2000.

[33] L. Bellatreche, K. Karlapalem, and A. Simonet. Horizontal class partitioning in object-oriented databases. *in 8th International Conference on Database and Expert Systems Applications (DEXA'97), Toulouse, Lecture Notes in Computer Science 1308*, pages 58–67, September 1997.

[34] L. Bellatreche, K. Karlapalem, and A. Simonet. Query optimization using horizontal class partitioning in object-oriented databases. *in 16th Congrès d'Informatique Des Organisations et Systèmes d'Information et de Décision (INFORSID'98), Montpellier*, pages 405–422, May 1998.

[35] L. Bellatreche, K. Karlapalem, and A. Simonet. Algorithms and support for horizontal class partitioning in object-oriented databases. *in the Distributed and Parallel Databases Journal*, 8(2):155–179, April 2000.

[36] L. Bellatreche, R. Missaoui, H. Necir, and H. Drias. Selection and pruning algorithms for bitmap index selection problem using data mining. In *International Conference on Data Warehousing and Knowledge Discovery (DaWaK'2007)*, pages 221–230, September 2007.

[37] L. Bellatreche, R. Missaoui, H. Necir, and H. Drias. A data mining approach for selecting bitmap join indices. *Journal of Computing Science and Engineering*, 2(1):206–223, January 2008.

[38] L. Bellatreche, D. Nguyen Xuan, G. Pierra, and H. Dehainsala. Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry Journal, Elsevier*, 57(8-9):711–724, December 2006.

[39] L. Bellatreche, G. Pierra, D. Nguyen-Xuan, and H. Dehainsala. Intégration de sources de données autonomes par articulation a priori d'ontologies. In *Actes du XXIIème Congrès INFORSID, Biarritz, France*, pages 283–298, 2004.

[40] L. Bellatreche, G. Pierra, D. N. Xuan, D. Hondjack, and Y. A. Ameur. An a priori approach for automatic integration of heterogeneous and autonomous databases. In *15th International Conference on Database and Expert Systems Applications (DEXA'04)*, pages 475–485, 2004.

[41] L. Bellatreche, M. Schneider, H. Lorinquer, and M. Mohania. Bringing together partitioning, materialized views and indexes to optimize performance of relational data warehouses. *Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2004)*, pages 15–25, September 2004.

[42] L. Bellatreche and A. Simonet. A horizontal fragmentation algorithm in object database design. *in Third International Conference of the Austrian Center for Parallel Computation (ACPC'96), with Special Emphasis on Parallel Databases and Parallel I/O, Lecture Note in Computer Science (LNCS), 1127*, pages 223–226, September 1996.

[43] L. Bellatreche and K. Y. Woameno. Dimension table driven approach to referential partition relational data warehouses. In *to appear in ACM Twelfth International Workshop on Data Warehousing and OLAP (DOLAP09)*, 2009.

[44] R. G. Bello, K. Dias, A. Downing, F. J. J., W. D. Norcott, H. Sun, A. Witkowski, and M. Ziauddin. Materialized views in oracle. *Proceedings of the International Conference on Very Large Databases*, pages 659–664, August 1998.

[45] D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: The MOMIS project demonstration. In *The VLDB Journal*, pages 611–614, 2000.

[46] S. Benkrid and L. Bellatreche. Une démarche conjointe de fragmentation et de placement dans le cadre des entrepôts de données parallèles. In *5ème Journée Francophone sur les Entrepôts de données et l'Analyse en ligne (EDA'09), Revue des Nouvelles Technologies de l'Information, RNTI-B-5*, pages 91–106, 2009.

[47] S. Benkrid, L. Bellatreche, and H. Drias. A combined selection of fragmentation and allocation schemes in parallel data warehouses. In *4th International Workshop on Data Management in Global Data Repositories (GREP'08)*, pages 370–374, 2008.

[48] S. Bergamaschi, S. Castano, S. D. C. di Vimercati, S. Montanari, and M. Vincini. Exploiting schema knowledge for the integration of heterogeneous sorces. In *Convegno Nazionale Sistemi di Basi di Dati Evolute (SEBD98), Ancona, Italy*, pages 103–120, 1998.

[49] T. Berners-Lee, J.Hendler, and O. Lassila. The semantic web. *Scientific American*, pages 36–43, Mai 2001.

[50] P. Bernstein. Applying model management to classical meta data problems. *in Proceedings of the 2003 CIDR Conference*, 2003.

[51] P. Bernstein, A. Y. Havely, and R. A. Pottinger. A vision of managament of complex models. *in SIGMOD Record*, 29(4):55–63, 2000.

[52] B.McBride. Jena: Implementing the rdf model and syntax specification. *Proceedings of the 2nd International Workshop on the Semantic Web.*, 2001.

[53] M. Body, M. Miquel, Y. Bédard, and A. Tchounikine. A multidimensional and multiversion structure for olap applications. In *DOLAP '02: Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 1–6, New York, NY, USA, 2002. ACM Press.

[54] C. Bontempo and G. Zagelow. The ibm data warehouse architecture. *Communications of the ACM*, 41(9):38–48, 1998.

[55] L. Bouganim. Equilibrage de charges lors de l'exécution de requêtes sur des architectures multiprocesseurs hybrides. Phd. thesis, Université de Versailles Saint Quentin en Yvelines, December 1996.

[56] K. Boukhalfa. De la conception physique aux outils d'administration et de tuning des entrepôts de données. Phd. thesis, Poitiers University, France, 2009.

[57] K. Boukhalfa and L. Bellatreche. Sélection de schéma de fragmentation horizontale dans les entrepôts de données. formalisation et algorithmes. *Ingénierie des Systèmes d'Information*, 11(6):55–82, 2006.

[58] K. Boukhalfa, L. Bellatreche, and Z. Alimazighi. Hp&bji: A combined selection of data partitioning and join indexes for improving olap performance. *Annals of Information Systems, Special Issue on new trends in data warehousing and data analysis, Springer*, pages 179–2001, 2008.

[59] K. Boukhalfa, L. Bellatreche, and S. Caffiau. De l'optimisation de requêtes aux outils d'administration des entrepôts de données. *Ingénierie des Systèmes d'Information (ISI)*, 13(6):33–62, 2008.

[60] E. Bozsak, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. Staab, L. Stojanovic, N. Stojanovic, R. Studer, G. Stumme, Y. Sure, J. Tane, R. Volz, and V. Zacharias. Kaon - towards a large scale semantic web. In *Third International Conference on E-Commerce and Web Technologies (EC-Web'02)*, pages 304–313, 2002.

[61] Y. Breitbart, A. Silberschatz, and G. R. Thompson. Reliable transaction management in a multidatabase system. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 215–224, 1990.

[62] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In I. Horrocks and J. Hendler, editors, *Proceedings of the First Internation Semantic Web Conference*, number 2342 in Lecture Notes in Computer Science, pages 54–68. Springer Verlag, July 2002.

[63] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83, New York, NY, USA, 2004. ACM Press.

[64] S. Ceri, M. Negri, and G. Pelagatti. Horizontal data partitioning in database design. *Proceedings of the ACM SIGMOD International Conference on Management of Data. SIGPLAN Notices*, pages 128–136, 1982.

[65] S. Ceri and G. Pelagatti. *Distributed Databases: Principles & Systems*. McGraw-Hill International Editions, 1984.

[66] S. Chakravarthy, J. Muthuraj, R. Varadarajan, and S. B. Navathe. An objective function for vertically partitioning relations in distributed databases and its analysis. *in Distributed and Parallel Databases Journal*, 2(2):183–207, April 1994.

[67] S. Chaudhuri. Index selection for databases: A hardness study and a principled heuristic solution. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1313–1323, November 2004.

[68] S. Chaudhuri and V. Narasayya. An efficient cost-driven index selection tool for microsoft sql server. *Proceedings of the International Conference on Very Large Databases*, pages 146–155, August 1997.

[69] S. Chaudhuri and V. Narasayya. Self-tuning database systems: A decade of progress. In *Proceedings of the International Conference on Very Large Databases*, pages 3–14, September 2007.

[70] S. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The tsimmis project: Integration of heterogeneous information sources. *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, pages 7–18, Mars 1994.

[71] C. Chee-Yong. Indexing techniques in decision support systems. Ph.d. thesis, University of Wisconsin - Madison, 1999.

[72] S. Cheevers. Oracle database 10g vs. microsoft sql server 2000: Technical overview. *An Oracle White Paper*, pages 1–30, March 2004.

[73] S. Choenni, H. Blanken, and T. Chang. On the selection of secondary indices in relational databases. *Data Knowledge Engineering*, 11(3):207–238, 1993.

[74] D. Core. The dublin core metadata initiative. *Available at* `http://dublincore.org/`.

[75] O. Council. Apb-1 olap benchmark, release ii. *http://www.olapcouncil.org/research/bmarkly.htm*, 1998.

[76] C. D. The dificulty of optimum index selection. *ACM Transactions on Database Systems (TODS)*, 3(4):440–445, march 1978.

[77] B. Dageville, D. Das, K. Dias, K. Yagoub, M. Zait, and M. Ziauddin. Automatic sql tuning in oracle 10g. In *Proceedings of the International Conference on Very Large Databases*, pages 1098–1109, 2004.

[78] M. Dean and Schreiber. Web Ontology Language Reference. *W3C Recommendation (2004)*, February 2004.

[79] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In *DS-8*, pages 351–369, 1999.

[80] H. Dehaindala. *Explicitation de la sémantique dans les bases de données: base de données à base ontologique et le modèle OntoDB*. PhD thesis, Université de Poitiers, July 2007.

[81] H. Dehainsala, G. Pierra, and L. Bellatreche. Benchmarking data schemes of ontology based databases. In *Proceeding On the Move to Meaningful Internet Systems 2006 (OTM'06) : ODBASE conference*, volume 1, pages 48–49, 2006.

[82] H. Dehainsala, G. Pierra, and L. Bellatreche. Ontodb: An ontology-based database for data intensive applications. In *12th International Conference on Database Systems for Advanced Applications (DASFAA'07)*, pages 497–508, Bangkok - Thailand, April 2007.

[83] D. Dewitt, R. H. Gerber, G. Graefe, M. L. Heytens, K. B. Kumar, and M. Muralikrishna. Gamma - a high performance dataflow database machine. *VLDB*, 10:228–237, 1986.

[84] D. J. D. DeWitt, S. Madden, and M. Stonebraker. How to build a high-performance data warehouse. *http://db.lcs.mit.edu/madden/high_perf.pdf*.

[85] G. Eadon, E. I. Chong, S. Shankar, A. Raghavan, J. Srinivasan, and S. Das. Supporting table partitioning by reference in oracle. *SIGMOD'08*, 2008.

[86] C. I. Ezeife and K. Barker. A comprehensive approach to horizontal class fragmentation in distributed object based system. *International Journal of Distributed and Parallel Databases*, 3(3):247–272, 1995.

[87] G. F. and S. D.L. Editing mad* task descriptions for specifying user interfaces, at both semantic and presentation levels. In *Harrison, M.D. and Torres, J.C. (Eds.), Proceedings of DSV-IS'97, Springer-Verlag*, pages 193–208, 1997.

[88] C. Fankam, S. Jean, L. Bellatreche, and Y. A. Ameur. Extending the ansi/sparc architecture database with explicit data semantics: An ontology-based approach. In *Second European Conferenc on Software Architecture (ECSA'08)*, pages 318–321, 2008.

[89] C. Fankam, S. Jean, G. Pierra, and L. Bellatreche. Extension de l'architecture de bases de données ansi/sparc pour expliciter la sémantique des données. In *2ème Conférence Francophone sur les Architectures Logicielles (CAL'2008)*, pages 47–61, 2008.

[90] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. Patel-Schneider. Oil: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, pages 38–44, 2001.

[91] M. Frank, E. Omiecinski, and S. Navathe. Adaptive and automated index selection in rdbms. In *3rd International Conference on Extending Database Technology (EDBT'92)*, pages 277–292, 1992.

[92] F. François Goasdoué, V. Lattès, and M. C. Rousset. The use of carin language and algorithms for information integration: The picsel system. *International Journal of Cooperative Information Systems (IJCIS)*, 9(4):383–401, December 2000.

[93] C.-H. Fung, K. Karlapalem, and Q. Li. Cost-driven vertical class partitioning for methods in object oriented databases. *VLDB Journal*, 12(3):187–210, December 2003.

[94] C. W. Fung, K. Karlapalem, and Q. Li. Cost-driven evaluation of vertical class partitioning in object oriented databases. *in Fifth International Conference On Database Systems For Advanced Applications (DASFAA'97), Melbourne, Australia*, pages 11–20, April 1997.

[95] P. Furtado. Experimental evidence on partitioning in parallel data warehouses. In *DOLAP*, pages 23–30, 2004.

[96] G. Gardarin. Intégration de données et applications via xml. In *georges.gardarin.free.fr/Cours_XML/8-HDI&EAI.ppt*.

[97] G. Gardarin, J.-R. Gruser, and Z.-H. Tang. A cost-based selection of path expression processing algorithms in object-oriented databases. *Proceedings of the International Conference on Very Large Databases*, pages 390–401, 1996.

[98] K. E. Gebaly and A. Aboulnaga. Robustness in automatic physical database design. *in 11th International Conference on Extending Database Technology (EDBT'08)*, pages 145–156, 2008.

[99] M. R. Genesereth, A. M. Keller, and O. M. Duschka. Infomaster: an information integration system. In *ACM SIGMOD International Conference on Management of Data*, pages 539–542, 1997.

[100] C. Goh, S. Bressan, E. Madnick, and M. D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3):270–293, 1999.

[101] M. Golfarelli, , and E. Rizzi, S. Saltarelli. Index selection for data warehousing. *Proceedings 4th International Workshop on Design and Management of Data Warehouses (DMDW'2002), Toronto, Canada*, pages 33–42, 2002.

[102] G.-L. G. Gomez. *Construction automatisée de l'ontologie de systèmes médiateurs. Application à des systèmes intégrant des services standards accessibles via le Web.* PhD thesis, Université Paris XI Orsay, 2005.

[103] G. Graefe. Fast loads and fast queries. In *11th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'09)*, pages 111–124, 2009.

[104] O. M. Group. Meta object facility specification version 1.4. *OMG document formal*, 2003.

[105] T. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199–220, 1995.

[106] N. Guarino and C. A. Welty. A formal ontology of properties. In *Knowledge Acquisition, Modeling and Management*, pages 97–112, 2000.

[107] N. Guarino and C. A. Welty. Ontological analysis of taxonomic relationships. *in Proceedings of 19th International Conference on Conceptual Modeling (ER'00)*, pages 210–224, October 2000.

[108] H. Gupta. Selection of views to materialize in a data warehouse. *Proceedings of the 6th International Conference on Database Theory (ICDT '97)*, pages 98–112, 1997.

[109] H. Gupta. Selection and maintenance of views in a data warehouse. Ph.d. thesis, Stanford University, September 1999.

[110] H. Gupta, V. Harinarayan, A. Rajaraman, and J. Ullman. Index selection for olap. *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 208–219, April 1997.

[111] M.-S. Hacid and C. Reynaud. L'intégration de sources de données. *La revue I3: Information - Interaction - Intelligence*, Vol5 n°1, 2005.

[112] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The stanford data warehousing project. *IEEE Quarterly Bulletin on Data Engineering; Special Issue on Materialized Views and Data Warehousing*, 18(2):41–48, 1995.

[113] S. Harris and N. Gibbins. 3store: Efficient bulk rdf storage, 2003.

[114] J. Heflin. Owl web ontology language use cases and requirements. *Available at* `http://www.w3.org/TR/webont-req/`.

[115] J. Heflin. *Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment.* PhD thesis, University of Maryland, College Park., 2001.

[116] J. Heflin and J. Hendler. Dynamic ontologies on the web. *American Association for Artificial Intelligence*, pages 443–449, 2000.

[117] C. A. Hurtado, O. A. Mendelzon, and V. A. A. Maintaining data cubes under dimension updates. *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 346–355, March 1999.

[118] IEC. Iec 61360 - component data dictionary. *International Electrotechnical Commission. Available at* `http://dom2.iec.ch/iec61360?OpenFrameset`, 2001.

[119] G. iederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, Marsh 1992.

[120] W. H. Inmon. *Building the Data Warehouse*. John Wiley and Sons, Third edition, 2002.

[121] IS010303.02. Product data representation and exchange - part 2 : Express reference manual. *ISO*, (055), 1994.

[122] ISO13584-25. Industrial automation systems and integration - parts library - part 25 : Logical resource: Logical model of supplier library with aggregate values and explicit content. Technical report, International Standards Organization, Genève, 2004.

[123] H. Jagadish, L. V. S. Lakshmanan, and D. Srivastava. What can hierarchies do for your data warehouses. *Proceedings of the International Conference on Very Large Databases*, pages 530–541, September 1999.

[124] M. Jarke and Y. Vassiliou. Data warehouse quality design: A review of the dwq project. In *Invited Paper, 2nd Conference on Information Quality. Massachusetts Institute of Technology, Cambridge*, 1997.

[125] S. Jean. *OntoQL, un langage d'exploitation des bases de données à base ontologique*. PhD thesis, Université de Poitiers, Décembre 2007.

[126] S. Jean, H. Dehainsala, D. Nguyen Xuan, G. Pierra, L. Bellatreche, and Y. Aït-Ameur. Ontodb: It is time to embed your domain ontology in your database. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07) (Demo Paper)*, pages 1119–1120, April 2007.

[127] S. Jean, G. Pierra, and Y. Aït-Ameur. Domain ontologies : a database-oriented analysis. In *Web Information Systems and Technologies (WEBIST'2006)*, pages 341–351, 2006.

[128] T. Johnson. Performance measurements of compressed bitmap indices. *Proceedings of the International Conference on Very Large Databases*, pages 278–289, 1999.

[129] K. Karlapalem and Q. Li. Partitioning schemes for object oriented databases. *in Proceeding of the Fifth International Workshop on Research Issues in Data Engineering- Distributed Object Management, RIDE-DOM'95*, pages 42–49, March 1995.

[130] K. Karlapalem, Q. Li, and S. Vieweg. Method induced partitioning schemes in object-oriented databases. pages 377–384, May 1996.

[131] K. Karlapalem and N. M. Pun. Query driven data allocation algorithms for distributed database systems. *in 8th International Conference on Database and Expert Systems Applications (DEXA'97), Toulouse, Lecture Notes in Computer Science 1308*, pages 347–356, September 1997.

[132] M. Klein. *Change Management for Distributed Ontologies*. PhD thesis, Amsterdam University, 2004.

[133] Y. Kotidis and N. Roussopoulos. An alternative storage organization for rolap aggregate views based on cubetrees. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 249–258, June 1998.

[134] T. Kraft, H. Schwarz, and B. Mitschang. A statistics propagation approach to enable cost-based optimization of statement sequences. pages 267–282, October 2007.

[135] W. Labio, D. Quass, and B. Adelberg. Physical database design for data warehouses. *Proceedings of the International Conference on Data Engineering (ICDE)*, 1997.

[136] W. J. Labio, Y. Zhuge, J. L. Wiener, H. Gupta, H. García-Molina, and J. Widom. The WHIPS prototype for data warehouse creation and maintenance. In *Proceedings of SIGMOD*, pages 557–559, 1997.

[137] B. T. Le, R. Dieng-Kuntz, and F. Gandon. On ontology matching problems - for building a corporate semantic web in a multi-communities organization. In *6e Conference Internationale en Systeme d'Information d'Entreprise (ICEIS)*, pages 236–243, 2004.

[138] H. Lei and K. A. Ross. Faster joins, self-joins and multi-way joins using join indices. *Data and Knowledge Engineering*, 28(3):277–298, November 1998.

[139] A. Y. Levy, A. Rajaraman, and J. J. Ordille. The world wide web as a collection of views: Query processing in the information manifold. *Proceedings of the International Workshop on Materialized Views: Techniques and Applications (VIEW'1996)*, pages 43–55, June 1996.

[140] A. A. B. Lima, C. Furtado, P. Valduriez, and M. Mattoso. Improving parallel olap query processing in database clusters with data replication. *Distributed and Parallel Database Journal*, 25(1-2):97–123, 2009.

[141] L.Ma, Z. Su, Y. Pan, L. Zhang, and T. Liu. Rstar: an rdf storage and query system for enterprise resource management. *thirteenth ACM international conference on Information and knowledge management*, pages 484 – 491, 2004.

[142] N. Lumineau. *Organisation et Localisation de Données Hétérogènes et Réparties sur un Réseau Pair-à-Pair*. PhD thesis, Université Pierre et Marie Curie - Paris VI, 2005.

[143] N. Lumineau, A. Doucet, and S. Gançarski. Thematic schema building for mediation-based peer-to-peer architecture. *Electronic Notes in Theoretical Computer Science*, 150(2):21–36, 2006.

[144] H. Mahboubi and J. Darmont. Data mining-based fragmentation of xml data warehouses. In *ACM 11th International Workshop on Data Warehousing and OLAP (DOLAP'08)*, pages 9–16, 2008.

[145] F. Manola, E. Miller, W3C, and B. McBride. Rdf primer. *http://www.w3.org/TR/2004/REC-rdf-primer-20040210/*, Juin 2005.

[146] D. L. McGuinness and F. Harmelen. Owl web ontology language overview. *W3C Recommendation*, 10 February 2004.

[147] M. Mehta and D. J. DeWitt. Data placement in shared-nothing parallel database systems. *VLDB Journal*, 6(1):53–72, 1997.

[148] E. Mena, V. Kashyap, A. P. Sheth, and A. Illarramendi. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Conference on Cooperative Information Systems*, pages 14–25, 1996.

[149] S. Menon. Allocating fragments in distributed databases. *IEEE Transactions on Parallel and Distributed Systems*, 16(7):577–585, July 2005.

[150] P. Merle, C. Gransart, and J.-M. Geib. Corbascript and corba web: A generic object-oriented dynamic environment upon corba. In *In Proceedings of TOOLS Europe 96, Paris, France*, pages 97–112, 1996.

[151] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *VLDB '93: Proceedings of the 19th International Conference on Very Large Data Bases*, pages 120–133, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[152] M. Minsky. A framework for representing knowledge. *in Winston PE, ed. The Psychology of computer vision. New-York: McGraw-Hill*, 1975.

[153] H. MOULOUDI. *Personnalisation de requêtes et visualisations OLAP sous contraintes*. PhD thesis, Université de Tours, Décembre 2007.

[154] H. Mouloudi, L. Bellatreche, A. Giacometti, and P. Marcel. Personalization of mdx queries. In *22èmes Journées Bases de Données Avancées (BDA(06)*, 2006.

[155] musicbrainz. mmsicbrainz metadatabase. *Available at* `http://musicbrainz.org/`.

[156] A. Napoli. Une brève introduction aux logiques de descriptions. *Cours de la Logique de Description*, 2005.

[157] S. Navathe, S. Ceri, G. Wiederhold, and D. J. Vertical partitioning algorithms for database design. *ACM Transaction on Database Systems*, 9(4):681–710, December 1984.

[158] S. Navathe and M. Ra. Vertical partitioning for database design : a graphical algorithm. *ACM SIGMOD*, pages 440–450, 1989.

[159] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, pages 36–56, 1991.

[160] Nescape. Open directory project (odp). *Available zt* `http://www.aef-dmoz.org/help/geninfo.html`.

[161] A. Y. Noaman and K. Barker. A horizontal fragmentation algorithm for the fact relation in a distributed data warehouse. *in the 8th International Conference on Information and Knowledge Management (CIKM'99)*, pages 154–161, November 1999.

[162] N. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 5, 2003.

[163] N. F. Noy and M. A. Musen. Ontology versioning in an ontology management framework. *IEEE Intelligent Systems*, 19(4):6–13, 2004.

[164] F. Oliveira, K. Nagaraja, R. Bachwani, R. Bianchini, R. P. Martin, and T. D. Nguyen. Understanding and validating database system administration. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 19–19, Berkeley, CA, USA, 2006. USENIX Association.

[165] P. O'Neil and D. Quass. Improved query performance with variant indexes. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 38–49, May 1997.

[166] OntoBroker®. How to write f-logic programs. In *A Tutorial for the Language F-Logic*. Copyrighted by Ontoprise GmbH, November 2004.

[167] B. S. Ozkan, N., and P. C. Choosing the right task-modeling notation: A taxonomy. *The Handbook of Task Analysis for Human-Computer Interaction, D. Diaper and N. Stanton (Eds.), Lawrence Erlbaum Associates (LEA)*, 2004.

[168] M. T. Özsu and P. Valduriez. Distributed database systems : Where are we now? *IEEE COMPUTER*, 24(8):68–78, August 1991.

[169] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems : Second Edition*. Prentice Hall, 1999.

[170] Z. Pan and J. Heflin. Dldb: Extending relational databases to support semantic web queries. In *Workshop on Practical and Scaleable Semantic Web Systms, ISWC 2003*, pages 109–113, 2003.

[171] S. Papadomanolakis and A. Ailamaki. Autopart: Automating schema design for large scientific databases using data partitioning. *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004)*, pages 383–392, June 2004.

[172] C. Parent and S. Spaccapietra. Intégration de bases de données:panorama des problèmes et des

approches. *Ingénierie des systèmes d'information*, 4(3), 1996.

[173] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets. *ICDT*, pages 398–416, 1999.

[174] F. Paterno, G. Mori, and R. Galimberti. *CTTE: An Environment for Analysis and Development of Task Models of Cooperative Applications*. ACM Press, 2001.

[175] G. Pierra. A multiple perspective object oriented model for engineering design. *in New Advances in Computer Aided Design & Computer Graphics*, pages 368–373, 1993.

[176] G. Pierra. Un modèle formel d'ontologie pour l'ingénierie, le commerce électronique et le web sémantique: Le modèle de dictionnaire sémantique plib. *Journées Scientifique WEBSEMANTIQUE, Paris*, 2002.

[177] G. Pierra. Context-explication in conceptual ontologies : The PLIB approach. In *Proc. of Concurrent Engineering (CE'2003)*, pages 243–254, July 2003.

[178] G. Pierra. Context-explication in conceptual ontologies: The plib approach. *Proceedings of CE'2003, Special track on Data Integration in Engineering, Madeira, Portugal, edited by R. Jardim-Gonçalves and J. Cha and A. Steiger-Garçao*, pages 243–254, July 2003.

[179] G. Pierra. Context representation in domain ontologies and its use for semantic integration of data. *Journal on Data Semantics*, 10:174–211, 2008.

[180] G. Pierra, H. Dehainsala, Y. A. Ameur, L. Bellatreche, J. Chochon, and M. Mimoune. Base de Données à Base Ontologique : le modèle OntoDB. In *Proceeding of Base de Données Avancées 20èmes Journées (BDA'04)*, pages 263–286, Oct 2004.

[181] G. Pierra, J. C. Potier, G. Battier, E. Sardet, J. C. Derouet, N. Willmann, and A. Mahir. Exchange of component data: The plib (iso 13584) model, standard and tools. pages 160–176, September 98.

[182] E. Rahm and P. Bernstein. On matching schemas automatically. *VLDB Journal*, 10(4):334–350, 2001.

[183] J. Rao, C. Zhang, G. Lohman, and N. Megiddo. Automating physical database design in a parallel database. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 558–569, June 2002.

[184] R. Ravat. *od*^{3}: contribution méthodologique à la conception de bases de données orientées objet réparties. Thèse de doctorat, Université Paul Sabatier, September 1996.

[185] C. Reynaud and G. Giraldo. An application of the mediator approach to services over the web. *Special track "Data Integration in Engineering, Concurrent Engineering (CE'2003)*, pages 209–216, July 2003.

[186] J. F. Roddick. Dynamically changing schemas within database models. *Aust. Comput. J.*, 23(3):105–109, 1991.

[187] J. F. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383–393, 1995.

[188] D. C. Rogozan. Gestion de l'évolution d'une ontologie : méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions de l'ontologie. *Rapport de recherche en informatique cognitive*, 2005.

[189] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Database Processing Fundamentals, Design, and Implementation*. Prentice-Hall International Editions, 1995.

[190] E. A. Rundensteiner, A. Koeller, and X. Zhang. Maintaining data warehouses over changing information sources. *Commun. ACM*, 43(6):57–62, 2000.

[191] A. Sanjay, V. R. Narasayya, and B. Yang. Integrating vertical and horizontal partitioning into automated physical database design. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 359–370, June 2004.

[192] A. Sanjay, C. Surajit, and V. R. Narasayya. Automated selection of materialized views and indexes in microsoft sql server. *Proceedings of the International Conference on Very Large Databases*, pages 496–505, September 2000.

[193] D. Schenk and P. Wilson. *Information Modelling The EXPRESS Way*. Oxford University Press, 1994.

[194] E. Sciore. Versioning and configuration management in an object-oriented data model. *The VLDB Journal*, 3(1):77–106, 1994.

[195] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[196] E. Simon. Reality check: a case study of an eii research prototype encountering customer needs.

In *11th International Conference on Extending Database Technology (EDBT'08)*, page 1, March 2008.

[197] C. Soutou. Modeling relationships in object-relational databases. *Data and Knowledge Engineering*, 36(1):79–107, 2001.

[198] D. Srivastava, S. Dar, H. Jagadish, and A. Y. Levy. Answering queries with aggregation using views. *Proceedings of the International Conference on Very Large Databases*, pages 318–329, September 1996.

[199] K. Stoffel, M. Taylor, and J. Hendler. Efficient management of very large ontologies. In *AAAI/IAAI*, pages 442–447, 1997.

[200] T. Stöhr, H. Märtens, and E. Rahm. Multi-dimensional database allocation for parallel data warehouses. *Proceedings of the International Conference on Very Large Databases*, pages 273–284, 2000.

[201] T. Stöhr and E. Rahm. Warlock: A data allocation tool for parallel warehouses. *Proceedings of the International Conference on Very Large Databases*, pages 721–722, 2001.

[202] L. Stoimenov. Bridging objects and relations : a mediator for oo front-end to rdbms. *Information and Software Technology*, 41:57–66, Oct. 1999.

[203] L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, Karlsruhe University, 2004.

[204] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. User-driven ontology evolution management. In *EKAW*, pages 285–300, 2002.

[205] N. Stojanovic and L. Stojanovic. Usage-oriented evolution of ontology-based knowledge management systems. In *CoopIS/DOA/ODBASE*, pages 1186–1204, 2002.

[206] Z. A. Talebi, R. Chirkova, Y. Fathi, and M. Stallmann. Exact and inexact methods for selecting views and indexes for olap performance improvement. In *11th International Conference on Extending Database Technology (EDBT'08)*, pages 311–322, Mars 2008.

[207] J. Tarby and M. Barthet. Analyse et modélisation des tâches dans la conception des systèmes d'information : la méthode diane+. In *Analyse et conception de l'IHM, interaction pour les Systèmes d'Information*, 2001.

[208] G. Valentin, M. Zuliani, D. C. Zilio, G. M. Lohman, and A. Skelley. Db2 advisor: An optimizer smart enough to recommend its own indexes. *ICDE'00*, pages 101–110, 2000.

[209] G. van der Veer. Ta: Groupware task analysis - modeling complexity. *In Acta Psychologica*, pages 297–322, 1996.

[210] T. Vögele, H. S., and S. G. Buster-an information broker for the semantic web. *Knstliche Intelligenz*, 3:31–34, July 2003.

[211] P. R. S. Visser, M. Beer, T. Bench-Capon, B. M. Diaz, and M. J. R. Shave. Resolving ontological heterogeneity in the kraft project. *10th International Conference on Database and Expert Systems Applications (DEXA'99)*, pages 668–677, September 1999.

[212] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information - a survey of existing approaches. pages 108–117, August 2001.

[213] H.-C. Wei and E. Ramez. Study and comparison of schema versioning and database conversion techniques for bi-temporal databases. *Sixth International Workshop, TIME-99 Proceedings*, pages 88–98, 1999.

[214] H. Wiedmer and G. Pierra. *Methodology For Structuring Part Families*. ISO-IS 13584-42. ISO Genève, 1998.

[215] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds. Efficient rdf storage and retrieval in jena2. *HP Laboratories Technical Report HPL-2003-266*, pages 131–150, 2003.

[216] M.-C. Wu and A. P. Buchmann. Encoded bitmap indexing for data warehouses. In *Proceedings of the Fourteenth International Conference on Data Engineering (ICDE'98)*, pages 220–230, 1998.

[217] D. N. XUAN. *Intégration de bases de donnée hétérogènes par articulation a priori d'ontologies: applications aux Catalogues de Composants Industriels*. PhD thesis, Université de Poitiers, 2006.

[218] D. N. Xuan, L. Bellatreche, and G. Pierra. Ontology evolution and source autonomy in ontology-based data warehouses. In *2ème Journée sur Entrepôts de Données et Analyse en ligne (EDA'06), Revue des Nouvelles Technologies de l'Information*, pages 55–76, June 2006.

[219] D. N. Xuan, L. Bellatreche, and G. Pierra. Un modele a base ontologique pour la gestion de l'evolution asynchrone des entrepots de donnees. *MOSIM'06 :Modélisation, Optimisation et Simulation des Systèmes : Défis et Opportunités*, pages 1682–1691, April 2006.

[220] D. N. Xuan, L. Bellatreche, and G. Pierra. A versioning management model for ontology-based data warehouses. In *8th International Conference on Data Warehousing and Knowledge Discovery (DAWAK'06)*, pages 195–206, 2006.

[221] D. N. Xuan, L. Bellatreche, and G. Pierra. Ontodawa, un système d'intégration à base ontologique de sources de données autonomes et évolutives. *Ingénierie des Systèmes d'Information*, 13(2):97–125, 2008.

[222] J. Yang, K. Karlapalem, and Q. Li. Algorithms for materialized view design in data warehousing environment. *Proceedings of the International Conference on Very Large Databases*, pages 136–145, August 1997.

[223] Y. Zhang and M.-E. Orlowska. On fragmentation for distributed database design. *Information Sciences*, 1(3):117–132, 1994.

[224] Y. Zhuge, H. Garcia-Molina, and J. L. Wiener. The strobe algorithme for multi-source warehouse consistency. In *PDIS Conference, Miami Beach-Florida*, pages 146–157, 1996.

[225] D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. Storm, C. Garcia-Arellano, and S. Fadden. Db2 design advisor: Integrated automatic physical database design. *Proceedings of the International Conference on Very Large Databases*, pages 1087–1097, August 2004.

Table des figures

Liste des tableaux

Résumé

Avec le développement d'Internet et des Intranets, l'échange et le partage de l'énorme quantité d'informations provenant de diverses sources de données éparpillées sur la Toile ou au sein de différentes organisations sont devenus cruciaux. Pour répondre à ces besoins, des solutions d'intégration ont été proposées selon trois dimensions : l'intégration des données, l'intégration des applications et l'intégration des plateformes. Le travail présenté dans ce mémoire vise à proposer des solutions innovantes au niveau de la construction d'un système d'intégration de données. Une démarche compréhensive de développement d'un système d'intégration est présentée. Elle s'articule autour de trois principales phases : la construction d'un système d'intégration, son exploitation et sa personnalisation.

Pour la phase de construction, nous avons proposé une approche d'intégration sémantique automatique, tout en laissant à chacune des sources susceptibles d'être intégrer une autonomie significative tant au niveau de sa structure qu'au niveau de son évolution. Elle suppose que chaque source contienne à la fois sa propre ontologie et les relations sémantiques qui l'articulent a priori avec une ou plusieurs ontologi(s) partagé(s). Une telle source est appelée source de données à base ontologique (BDBO). Pour mettre en oeuvre notre approche d'intégration, nous avons d'abord proposé un modèle et une architecture gérant les sources de données à base ontologique. Cette architecture est constituée de quatre parties : les deux premières parties correspondent à la structure usuelle des bases de données : données reposant sur un schéma logique de données, et méta-base décrivant l'ensemble de la structure de tables. Les deux autres parties, originales, représentent respectivement les ontologies et le méta-modèle d'ontologie au sein d'un méta-modèle réflexif. Des mécanismes d'abstraction et de nomination permettent respectivement d'associer à chaque donnée le concept ontologique qui en définit le sens, et d'accéder aux données à partir des concepts, sans se préoccuper de la représentation des données.

Pour la phase d'exploitation de données, nous avons présenté des solutions pour offrir aux administrateurs des structures d'optimisation de requêtes. Etant donné que nous menons des travaux sur cette phase depuis 1996, nous avons proposé des solutions d'optimisation qui peuvent être appliquées aux systèmes d'intégration suivant une architecture matérialisée sous forme d'une base de données traditionnelle ou d'un entrepôt de données relationnels. Nous avons identifié deux types de sélection des structures d'optimisation : une sélection isolée et une sélection multiple. Dans la sélection isolée, nous avons présenté des algorithmes pour la fragmentation horizontale et les index de jointure. Pour la sélection multiple, nous avons étudié le problème de sélection des index de jointure binaire et la fragmentation dérivée en exploitant les similarités entre elles. D'autres problèmes sont également présentés comme le traitement parallèle et la répartition des ressources entre les structures redondantes (les vues matérialisées et les index). Pour faciliter les tâches d'administration, nous avons développé un outil assistant les administrateurs dans leurs tâches puvant être utilisé avant ou après la création de système d'intégration.

La personnalisation est une phase récente de nos travaux. Nous avons d'abord étudié son effet sur la sélection des structures d'optimisation. Récemment, nous avons proposé des solutions permettant la représentation des profils utilisateurs au sein d'une BDBO afin de faciliter leur partage et leur échange.