THESE

Pour l'obtention du Grade de

DOCTEUR DE L'ÉCOLE NATIONALE SUPÉRIEURE DE MÉCANIQUE ET D'AÉROTECHNIQUE

(Diplôme National — Arrêté du 25 Mai 2016)

Ecole Doctorale : Science et Ingénierie pour l'Information, Mathématiques
Secteur de Recherche : INFORMATIQUE ET APPLICATIONS

Présentée par :

Okba BARKAT

****

Utilisation conjointe des ontologies et du contexte pour la conception des systèmes de stockage de données

****

Directeurs de Thèse : **Ladjel BELLATRECHE** et **Yamine AIT AMEUR**

****

Soutenue le 24/01/2017
devant la Commission d'Examen

JURY

Président	**:**	Mirian HALFELD FERRARI	Professeur, Université Orléans
Rapporteurs	**:**	Djamal BENSLIMANE	Professeur, LIRIS, Université Lyon1
		Laurent D'ORAZIO	Professeur, IRISA, Université de Rennes 1
Membres	**:**	Dalila TAMZALIT	Maître de conférences, HDR, Université de Nantes
		Yamine AIT AMEUR	Professeur, INPT-ENSEEIHT/IRIT, Toulouse
		Ladjel BELLATRECHE	Professeur, ISAE-ENSMA, Poitiers

Remerciements

Merci à :

— **Ladjel BELLATRECHE**, mon directeur de thèse, pour m'avoir guidé tout au long de ce travail. Je le remercie pour la confiance qu'il m'a témoignée, pour sa disponibilité, pour ses précieuses orientations et pour sa franchise et sa sympathie. Je mesure la chance qui m'a été donnée d'être encadré par un directeur de thèse aussi impliqué, ouvert et compétent ;

— **Yamine AIT AMEUR**, mon co-directeur de thèse, pour son soutien et ses conseils avisés et précieux ;

— **Selma KHOURI**, pour son aide et sa contribution à la rédaction de certains articles ;

— **Emmanuel GROLLEAU**, directeur du LIAS, pour m'avoir accueilli au sein du laboratoire ;

— **Djamal BENSLIMANE** et **Laurent D'ORAZIO** de m'avoir fait l'honneur de rapporter cette thèse et à qui je souhaite exprimer ma profonde reconnaissance ;

— **Mirian HALFELD FERRARI** et **Dalila TAMZALIT** pour avoir accepté d'être membres du jury en tant qu'examinateurs. Je les remercie pour leur disponibilité et leur disposition à juger ce travail ;

— Tout le personnel du LIAS pour leur disponibilité et leur bonne humeur. Je pense particulièrement à : **Yassine Ouhamou, Allel HADJALI, Mickael Baron, Stéphane Jean, Frédéric Carreau, Claudine Rault, Brice Chardin, Laurent Guittet, et Pascal Richard** ;

— Tous mes amis et collègues: **Zouhir**, **Nadir**, **Ahcene**, **Lahcen**, **Guillaume**, **Selma**, **Nassima**, **Olga**, **Géraud**, **Thomas**, **Abdelkrim**, **Abdallah**, **Ibrahim**, **Anh Toan**, **Yves**, **Cyrille** pour tous les bons moments passés en leur compagnie ;

— Enfin et surtout à toute ma famille et particulièrement à **mes parents** et ma femme **Samira** pour leur soutien sans faille. Rien n'aurait été possible sans eux.

A mes deux anges ***Rihame*** *et* ***Adam***

A tous ceux qui me sont chers ...

Table des matières

Chapitre 1 Introduction générale **1**

1 Contexte et problématique 3

2 Objectifs de la thèse et contributions 8

3 Organisation du mémoire de la thèse 10

Partie I État de l'art

Chapitre 2 Explicitation et persistance de la donnée sémantique **15**

1 Introduction 18

2 De la donnée à la donnée sémantique 18

2.1 La donnée 18

2.2 Ontologie 19

2.2.1 Définitions et caractéristiques 20

2.2.2 Ontologies vs modèles conceptuels 21

2.2.3 Taxonomie des ontologies 22
2.2.4 Formalismes Ontologiques 23
3 Des bases de données aux bases de données sémantiques 25
3.1 Bases de données sémantiques 25
3.1.1 BDS pour le web sémantique 26
3.1.2 BDS pour l'ingénierie 26
3.2 Modèles de stockage pour les BDS 27
3.2.1 Modèle de stockage des ontologies 27
3.2.2 Modèle de stockage des instances ontologiques 28
3.3 Architectures des BDS 32
3.3.1 Architecture deux quarts 32
3.3.2 Architecture trois quarts 32
3.3.3 Architecture quatre quarts 33
3.4 Langages d'exploitation des BDS 33
4 Des entrepôts de données aux entrepôts de données sémantiques 34
4.1 Entrepôts de Données 34
4.1.1 Définition et architecture 34
4.1.2 Modélisation multidimensionnelle 35
4.1.3 Cycle de conception des ED 37
4.2 Entrepôts de Données Sémantiques 40
4.2.1 Définition 40
4.2.2 Contribution des ontologies à la construction des EDS . . 40
5 Conclusion 43

Chapitre 3 Explicitation et persistance du contexte **45**

1 Introduction 47
2 Comprendre la notion de contexte 47
2.1 Définitions du contexte 47

2.1.1 Ce que disent les dictionnaires . . . 48
2.1.2 Ce que dit la littérature . . . 48
2.2 Catégorisation du contexte . . . 49
2.2.1 Catégorisation conceptuelle . . . 50
2.2.2 Catégorisation opérationnelle . . . 50
2.3 Modélisation du contexte . . . 52
3 Couplage entre le contexte et l'ontologie . . . 55
3.1 Ontologies représentant la dépendance au contexte . . . 55
3.2 Ontologies contextualisées . . . 56
3.3 Ontologies de multi-représentation . . . 57
3.4 Synthèse . . . 58
4 Contexte et systèmes de stockage . . . 59
4.1 Contexte et Bases de Données classiques . . . 59
4.1.1 Approches existantes . . . 59
4.1.2 Étude comparative et synthèse . . . 60
4.2 Contexte et Entrepôts de Données . . . 62
4.2.1 Approches existantes . . . 62
4.2.2 Étude comparative et synthèse . . . 63
5 Conclusion . . . 64

Partie II Contributions

Chapitre 4 Modélisation et explicitation du contexte : une approche d'externalisation 69

1 Introduction . . . 71

2 Fondements théoriques 72
2.1 Définition des ressources de modélisation 72
2.1.1 Standards de l'OMG 72
2.1.2 Ressources de modélisation pour notre modèle de contexte 75
2.2 Langage de modélisation EXPRESS 75
2.2.1 Le langage 75
2.2.2 Représentation des instances : fichier physique 77
2.2.3 Représentation graphique : EXPRESS-G 77
2.2.4 Choix d'EXPRESS comme langage de modélisation . . . 78
3 Notion de contexte 79
3.1 Notre vision du contexte 79
3.2 Notre catégorisation du contexte 80
3.2.1 Cas d'étude 80
3.2.2 Catégories du contexte 81
4 Notre approche de modélisation de contexte 84
4.1 Notre modèle de Contexte 84
4.1.1 L'entité 'Definition Context' 86
4.1.2 L'entité 'Evaluation Context' 87
4.2 Lien du modèle de contexte avec l'ontologie 94
5 Conclusion 95

Chapitre 5 Vers une contextualisation des Bases de Données Sémantiques 97

1 Introduction 100
2 Fondements théoriques 101
2.1 La BDS *OntoDB* 101
2.2 Langage OntoQL 101
2.2.1 Exploitation des ontologies 103
2.2.2 Exploitation des instances 104

2.2.3 Exploitation du méta-modèle 104
2.3 Choix du système OntoDB/OntoQL comme infrastructure de développement 105
3 Persistance du contexte dans les BDS : l'exemple d'OntoDB 105
3.1 Extension du modèle d'OntoDB avec le modèle de contexte proposé 105
3.1.1 Création des constructeurs correspondant à la catégorie "*Contexte de définition*" 106
3.1.2 Création des constructeurs correspondant à la catégorie "*Contexte d'évaluation*" 106
3.2 Mise en œuvre du lien entre le modèle de contexte et le modèle d'ontologie dans OntoDB 109
4 Prise en charge du contexte dans les requêtes : extension du langage OntoQL 110
4.1 Extension de la grammaire d'OntoQL : l'opérateur 'Using Context' 110
4.2 Extension de la sémantique d'OntoQL : Interprétation des requêtes 111
5 Validation : étude de cas 114
5.1 Persistance du fragment de l'ontologie *Healthcare* 115
5.2 Persistance et interrogation des informations contextuelles 115
6 Conclusion 123

Chapitre 6 Projection du contexte sur les phases conceptuelles de l'Entrepôt de Données Sémantique 125

1 Introduction 128
2 Nos hypothèses 128
3 Formalisation des entrées de notre approche 129
3.1 Ontologies contextuelles 129
3.2 Besoins des utilisateurs 131
3.3 Canevas d'intégration de sources de données sémantiques contextuelles 132
3.3.1 Schéma global $\mathcal{G}$ 132

3.3.2 Sources $\mathcal{S}$ . . . 133
3.3.3 Mappings $\mathcal{M}$ . . . 134
4 Notre approche de conception des EDS Contextuels . . . 135
4.1 Modélisation Conceptuelle . . . 135
4.1.1 Définition de l'OED . . . 135
4.1.2 L'annotation multidimensionnelle et contextuelle de l'OED 136
4.2 Modélisation Logique . . . 138
4.3 Processus ETL . . . 138
4.3.1 Opérateurs ETL . . . 138
4.3.2 Détails de l'algorithme ETL . . . 144
4.4 Modélisation physique . . . 147
5 Validation : expérimentation et prototype . . . 149
5.1 Présentation de notre prototype : CONTIC-DW Design Tool . . . 149
5.1.1 Environnement de développement . . . 149
5.1.2 Architecture fonctionnelle de CONTIC-DW Design Tool 149
5.2 Présentation de notre expérimentation . . . 150
5.2.1 Scénario d'expérimentation . . . 150
6 Conclusion . . . 157

Chapitre 7 Conclusion **159**

1 Conclusion . . . 161
1.1 Approche générique pour la modélisation du contexte . . . 161
1.2 Contextualisation des BDS . . . 162
1.3 Contextualisation des EDS . . . 162
2 Perspectives . . . 163
2.1 Enrichissement de notre modèle de contexte . . . 163
2.2 Extension du langage OntoQL . . . 163
2.3 Proposition d'un langage OLAP contextuel . . . 164

2.4 Développement d'un outil CASE contextuel 164
2.5 Les besoins et le contexte 164
2.6 Expérimentation et validation de l'approche 164
2.7 Évolution de l'ontologie et du contexte 165
2.8 La considération d'autres types de données 165

Bibliographie **167**

Annexes **183**

Table des figures **191**

Liste des tableaux **193**

Glossaire **195**

Chapitre 1

Introduction générale

Sommaire

1 Contexte et problématique 3
2 Objectifs de la thèse et contributions 8
3 Organisation du mémoire de la thèse 10

1 Contexte et problématique

Avec la mondialisation, les entreprises, qu'elles soient de petite, moyenne ou grande taille, et les organismes ne cessent d'acquérir et de stocker des données générées localement ou issues de sources externes à des fins décisionnelles. Leur objectif est en effet de rester compétitifs ce qui passe souvent par un processus d'intégration de ces données. Ce dernier est une tâche difficile et coûteuse. Selon Gartner [1], le marché des solutions d'intégration a atteint 2.8 millions de dollars à la fin de 2015 et ne cesse d'augmenter. Une fois ces données intégrées, des outils décisionnels permettent de les traduire en informations utiles aux activités des entreprises et organismes.

Les données, pour lesquelles les entreprises dépensent des sommes d'argent importantes afin de les acquérir, peuvent être classées en deux catégories principales : **(a)** les *données traditionnelles* et **(b)** les *données techniques*. Cette classification est motivée par le fait que ces deux types ont suscité beaucoup d'intérêt de la part de deux communautés (le Web Sémantique et l'Ingénierie) pour expliciter la sémantique de leurs données afin de les échanger et de les intégrer, comme nous allons le voir dans la suite de ce manuscrit.

— **Les données traditionnelles.** Elles sont souvent structurées et utilisées dans le cadre des applications de gestion quotidienne (day-2-day applications) comme la gestion bancaire, les agences de voyages, etc. Le sens (la sémantique) de ces données est implicite, ce qui augmente leur ambiguïté, surtout dans le cas où elles sont partagées et/ou échangées par plusieurs acteurs ou applications. Les créateurs de ces données possèdent la sémantique, mais celle-ci n'est pas explicitée. Depuis plusieurs décennies, les systèmes de gestion de bases de données (SGBD) ont largement contribué au stockage et à l'exploitation de ce type de données via des langages de requêtes comme SQL.

Pour expliciter la sémantique de ces données, plusieurs travaux ont fait appel aux ontologies [80] qui ont explosé grâce aux travaux sur le Web Sémantique de Tim Berners-Lee et le développement du langage OWL [132]. Ce langage a été fractionné en trois langages distincts : OWL LITE, OWL DL et DL FULL. Récemment, plusieurs ontologies et bases de connaissances ont vu le jour comme YAGO (Yet Another Great Ontology) [165] et YAGO2 qui possède 10 millions d'entités avec plus de 120 millions d'instances ontologiques[89].

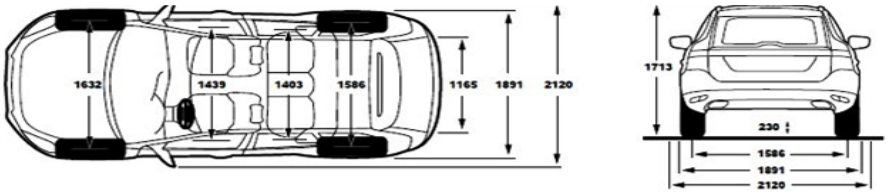
Les ontologies ont contribué à affiner la classification des données traditionnelles en deux sous-types : **(i)** les données traditionnelles annotées par une ou plusieurs ontologies et **(ii)** les données traditionnelles sémantiques de naissance. Dans le premier cas, les données traditionnelles ont été créées sans référencement à une ontologie. Ensuite, pour des besoins d'explicitation de leur sémantique, elles sont annotées par une ou plusieurs ontologies existantes [54]. Dans le deuxième cas, les ontologies sont porteuses de données qui

1. http://solutionsreview.com/data-integration/whats-changed-2016-gartner-magic-quadrant-for-data-integration-tools/

sont alors sémantiques. Plus précisément, les données sont nées avec leurs ontologies.

Le volume des données sémantiques, quelque soit leur type, a explosé après leur adoption par un nombre important de communautés : médecine [62], réseaux sociaux [133], ingénierie [141, 108], commerce électronique [3], etc. Pour assurer leur stockage et exploitation efficaces, un nombre important de systèmes de stockage a été proposé. Nous pouvons citer ainsi Oracle [53, 136], IBM Sor [121], Rdfsuite [4], Jena [130], etc. Cette exploitation est assurée par des langages de requêtes comme SPARQL [162].

— **Les données techniques.** Dans les années 80, les industriels, motivés par la généralisation des processus de conception, de fabrication, de maintenance et de démantèlement des produits industriels, ont émis le besoin de passer de documents à des *données techniques*. Cela peut être réalisé via des efforts de modélisation, d'échange, d'intégration et de publication des catalogues de composants. Une *donnée technique* est toute information qui permet de décrire le produit pendant son cycle de vie et ceci sur la totalité des métiers rencontrés (cf. figure 1.1). Elle peut être structurée ou non structurée. Les systèmes de gestion de données techniques (SGDT) ont largement contribué à gérer cette donnée. Pour faciliter l'intégration et le partage des données techniques, le projet STEP [120] a été lancé au niveau de l'ISO (International Organization for Standardization) pour définir une représentation non ambiguë des données du produit industriel, interprétable par tout système informatique, et couvrant tout le cycle de vie des produits.



	D5 AWD	D4 AWD
Motorisation :	2.4L, 5 cylindres en ligne, double turbo (haute et basse pression), injection directe diesel à rampe commune	2.4L, 5 cylindres en ligne, double turbo (haute et basse pression), injection directe diesel à rampe commune
Transmission :	Manuelle 6 vitesses ou Geartronic 6 vitesses	Manuelle 6 vitesses ou Geartronic 6 vitesses
Consommation Euromix (en l/100km) (MAN. / AUTO.) :	5.3/6.4	5.3/6.4
CO_2 rejeté (g/km) :	139/169	139/169
Puissance max., ECE (kW/ch/tr/min) :	158/215/4000	133/181/4000
Couple max., ECE (Nm/tr/min) (MAN. // AUTO.) :	420/1500 – 3250 //440/1500 – 3000	420/1500 – 2500
Accélération 0 à 100 km/h (secondes) :	8.1/8.3	9.8/10.2
Vitesse maximale sur circuit (km/h) (MAN. / AUTO.) :	210/205	200/195
Volume du réservoir (litres) :	70	70
Norme environnementale :	Euro 5	Euro 5

FIGURE 1.1 – Données techniques : exemple d'une voiture

Dans le cadre des efforts de normalisation de STEP, le projet PLIB (Part Library sous la norme ISO 13584) a été lancé en 1987 pour *modéliser*, *échanger* et *référencer* des catalogues informatisés de composants ou objets techniques préexistants. PLIB permet aussi bien la représentation de composants abstraits, tels que ceux utilisés par exemple

dans un processus de conception fonctionnelle, que celle de composants fournisseurs ou normalisés. Cette norme permet également d'associer à un objet du catalogue un nombre quelconque de représentations, propres à chacune des disciplines qui utilisent l'objet. Elle permet enfin d'intégrer dans un environnement homogène et cohérent des bibliothèques fournies par différentes sources [61].

Dans les premières versions de PLIB, les données de composants étaient *implicites*. Pour expliciter les données de composants conçus à l'aide de PLIB, plusieurs initiatives ont été menées pour définir des *ontologies* ou des *dictionnaires sémantiques*. Ces ontologies permettent d'identifier et de représenter les différents concepts présents dans un catalogue sous une forme échangeable (pour que l'utilisateur puisse comprendre le contenu s'il ne le connaît pas) et référençable. Plusieurs ontologies dans le milieu industriel ont été créées. Nous pouvons citer l'exemple de l'IEC pour les composants électroniques, ISO 13584-511 pour les fastners, etc.

Le volume des données techniques a également explosé. Par conséquent, des solutions de persistance ont été proposées. Nous pouvons citer l'exemple du projet OntoDB, développé au sein du laboratoire LIAS dans les années 2000 pour manipuler initialement les données techniques décrites par le modèle d'ontologie PLIB. OntoDB a été donc lancé pour ranger à la fois ces objets et les ontologies qui en définissent le sens au sein de bases de données. Ces dernières sont appelées des bases de données sémantiques (BDS) ou base de données à base ontologique (BDBO).

Bilan sur le besoin d'explicitation de la sémantique dans les systèmes de stockage : Cette discussion, nous a permis d'identifier trois initiatives principales de la part de la communauté scientifique, à savoir :

— le *besoin continuel* d'expliciter le sens des données, quelque soit leur type, à l'aide d'ontologies ou de dictionnaires de données,

— les efforts entrepris par les mondes industriel et académique pour proposer des solutions de stockage et d'exploitation des données volumineuses, et

— l'exploitation de toutes les fonctionnalités offertes par des ontologies en termes de description d'un domaine, de résolution des conflits sémantiques et syntaxiques, et de raisonnement. La particularité des ontologies est qu'elles ont montré leur contribution dans la conception des bases de données [195] et les entrepôts de données [105]. En ce qui concerne les bases de données, le recours aux ontologies a été motivé par la similarité de ces dernières avec les modèles conceptuels [43]. En effet, comme les modèles conceptuels, les ontologies conceptualisent l'univers du discours au moyen de classes hiérarchisées et de propriétés caractéristiques. Par contre, ils diffèrent dans leur objectif de modélisation. Les modèles conceptuels prescrivent les informations devant être représentées dans un système informatique particulier afin de répondre à un cahier des charges applicatif donné. A l'opposé, les ontologies visent à décrire, en se basant sur un consensus,

l'ensemble des informations permettant la conceptualisation des domaines d'applications indépendamment de toute application et de tout système dans lequel elles peuvent être utilisées. Pour les entrepôts de données, les ontologies ont été utilisées pour répondre à plusieurs besoins tels que : (i) la résolution des problèmes d'hétérogénéités syntaxique et sémantique qui pourraient exister lors de l'intégration et de l'échange de données entre plusieurs acteurs [141] et (ii) la construction et l'exploitation des entrepôts de données (Figure 1.2). Récemment, Alberto Abelló, Oscar Romero, Torben Bach Pedersen, Rafael Berlanga Llavori, Victoria Nebot, María José Aramburu Cabo et Alkis Simitsis ont publié un état de l'art dans la revue IEEE TKDE portant sur le rôle des ontologies dans la conception et l'exploitation des applications décisionnelles [1]. Plus précisément, dans le cadre de la thèse de Selma Khouri [105] effectuée au laboratoire LIAS, la présence des ontologies a été exploitée pour redéfinir les opérateurs d'ETL (Extract, Transform, Load) au niveau ontologique afin de les rendre plus génériques. Dans la thèse d'Aïcha BenSalem [178], effectuée au sein de l'entreprise TALEND [2] (leader mondial sur les outils ETL Open source), les ontologies ont contribué à augmenter la qualité des données de l'entrepôt de données.

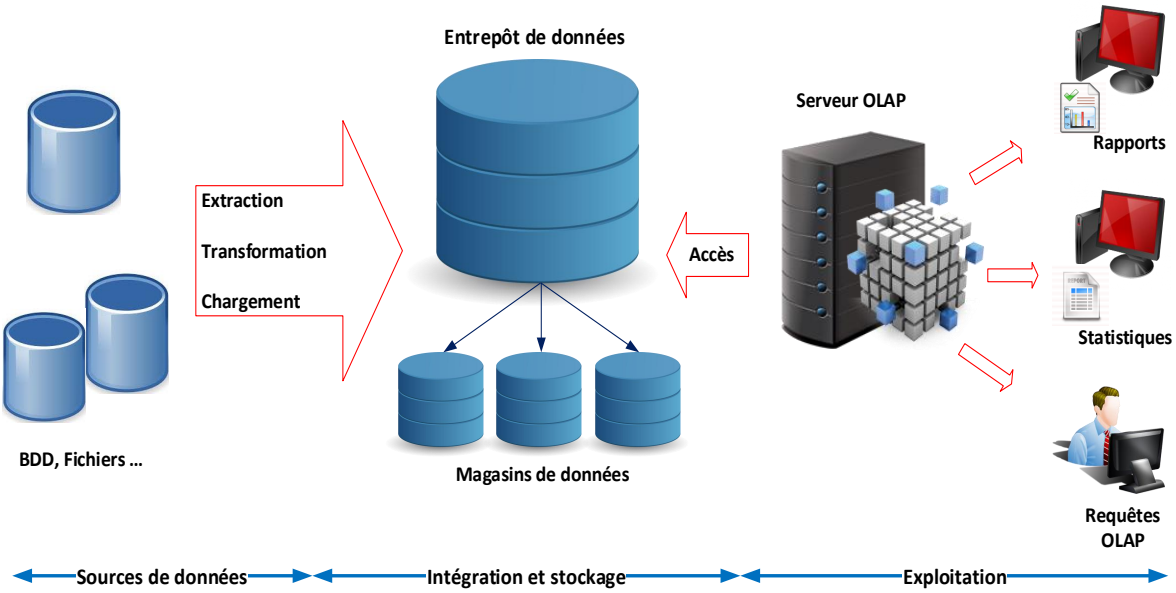


FIGURE 1.2 – Cycle de vie de construction d'un entrepôt de données

A ce stade, nous avons mis l'accent sur l'évolution des données traditionnelles vers les données sémantiques et sur le rôle des ontologies dans le processus de construction des bases/entrepôts de données sémantiques (cf. figure 1.3). Parallèlement, les ontologies ont pris en compte une autre dimension, en l'occurrence le *contexte*. Ce dernier a contribué au développement des applications *sensibles au contexte*. D'une manière générale, le contexte est défini comme un ensemble d'informations qui définissent l'environnement entourant une activité [2]. Nous tenons

2. https://fr.talend.com/

à souligner l'appropriation du contexte par plusieurs communautés techniques (informatique) et sociales (la philosophie, la psychologie et le traitement automatique du langage naturel, etc.). Cette *diversité* a fait naître plusieurs dimensions caractérisant le contexte. Ces dimensions se déclinent en deux catégories principales [199] : catégorisation conceptuelle et catégorisation opérationnelle. Chaque catégorie a ses propres dimensions (que nous détaillons dans le Chapitre 3). Par exemple, les travaux sur la recherche contextuelle d'informations prennent en compte cinq dimensions spécifiques du contexte [24] : (1) le dispositif, (2) la tâche/problème, (3) le document, (4) les éléments spatio-temporels et (5) l'utilisateur.

Vu l'intérêt du contexte, quelques études ont été menées pour l'intégrer dans le processus de construction des ontologies. Après une analyse des ces travaux, nous avons identifié deux familles principales de travaux s'articulant autour de la considération *jointe* des ontologies et du contexte : (i) des ontologies contextuelles pour le Web Sémantique et (ii) des ontologies contextuelles pour l'ingénierie. Pour la première famille, nous nous intéressons à deux travaux importants. Le premier est lié à la construction d'ontologies contextuelles [15] afin d'offrir la multi-representation des éléments ontologiques. Cela permet aux utilisateurs de naviguer à travers des contextes différents et de personnaliser les ontologies. Dans le cadre de ce travail, des constructeurs des logiques modales de description pour décrire la relation structurelle entre les contextes ont été définis. Le second concerne l'enrichissement du formalisme OWL par des concepteurs contextuels (C-OWL) [26].

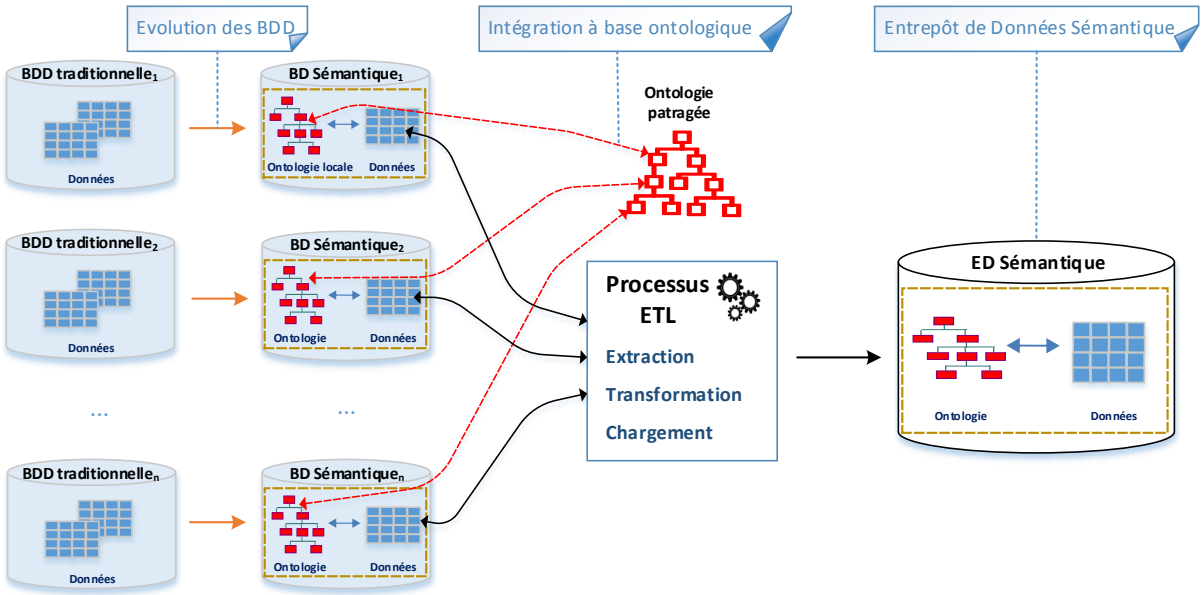


FIGURE 1.3 – Rôle des ontologies dans la construction et l'exploitation des systèmes de stockage de données

Pour la deuxième famille, nous avons identifié que le modèle d'ontologie PLIB a pris en compte le contexte dans la définition des *propriétés* pour le domaine de *l'ingénierie*.

Notons que ces deux familles sont complémentaires en terme de prise en compte du contexte.

La première famille s'intéresse particulièrement aux concepts (les classes de l'ontologie) contextuels, tandis que la deuxième se focalise sur les propriétés de l'ontologie. De plus dans les deux cas, le contexte a toujours été embarqué au sein des ontologies. Plus précisément, lors de la construction de l'ontologie, les concepteurs doivent prendre en compte le contexte afin de définir les concepts et les propriétés de l'ontologie. Cette prise en considération complexifie encore plus ce processus.

Bilan sur les ontologie contextuelles : Les travaux menés sur les ontologies contextuelles se sont focalisés sur l'embarcation du contexte au sein des ontologies et la définition de langages permettant de manipuler les concepts contextuels et le raisonnement. Ces ontologies contextuelles n'ont pas été utilisées dans la construction des applications de bases de données ou d'entrepôts de données. Pour bénéficier d'une manière continuelle des avantages des ontologies dans la conception des applications de bases de données avancées, il est indispensable de considérer d'une manière jointe *et flexible* les trois dimensions qui sont : les ontologies, le contexte, et les bases/entrepôts de données (cf. figure 1.4).

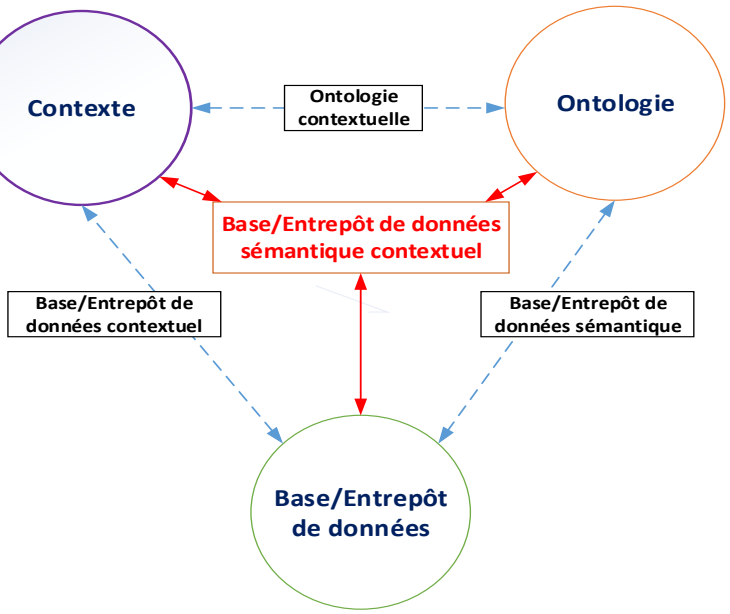


FIGURE 1.4 – Différentes liaisons entre ontologie, contexte et base/entrepôt de données

2 Objectifs de la thèse et contributions

Nos travaux tentent de répondre à l'enjeu que nous avons identifié, à savoir la prise en compte de l'intégralité des trois dimensions discutées précédemment (c'est-à-dire les ontologies, le contexte et les bases/entrepôts de données). Les travaux existants dans le domaine de la

prise en charge du contexte dans les systèmes de gestion et d'analyse de données souffrent de plusieurs problèmes :

— l'absence d'une description détaillée de toutes les dimensions du contexte. Cette description peut être réalisée avec le recours de la modélisation, en mettant la lumière sur les entités ainsi que leurs propriétés décrivant le contexte et les relations entre elles. Cette description doit être flexible, en acceptant d'autres dimensions futures ;

— la considération du contexte d'une manière embarquée dans les ontologies complexifie leur construction et leur exploitation. Une piste intéressante consiste alors à faire sortir le contexte de l'ontologie et assurer un lien avec elle ;

— l'absence de la valorisation du contexte dans le processus de construction et l'exploitation de bases de données avancées, telles que les entrepôts de données. Le fait que les ontologies ont largement contribué à ce processus et la forte relation entre les ontologies et le contexte nous motive à lier le contexte aux ontologies pour mieux réaliser ce processus.

Contrairement à ces travaux, nous proposons dans cette thèse une approche complète pour la contextualisation des bases de données sémantiques (BDS) et des entrepôts de données sémantiques (EDS) qui est constituée des trois contributions illustrées par la figure 1.5 :

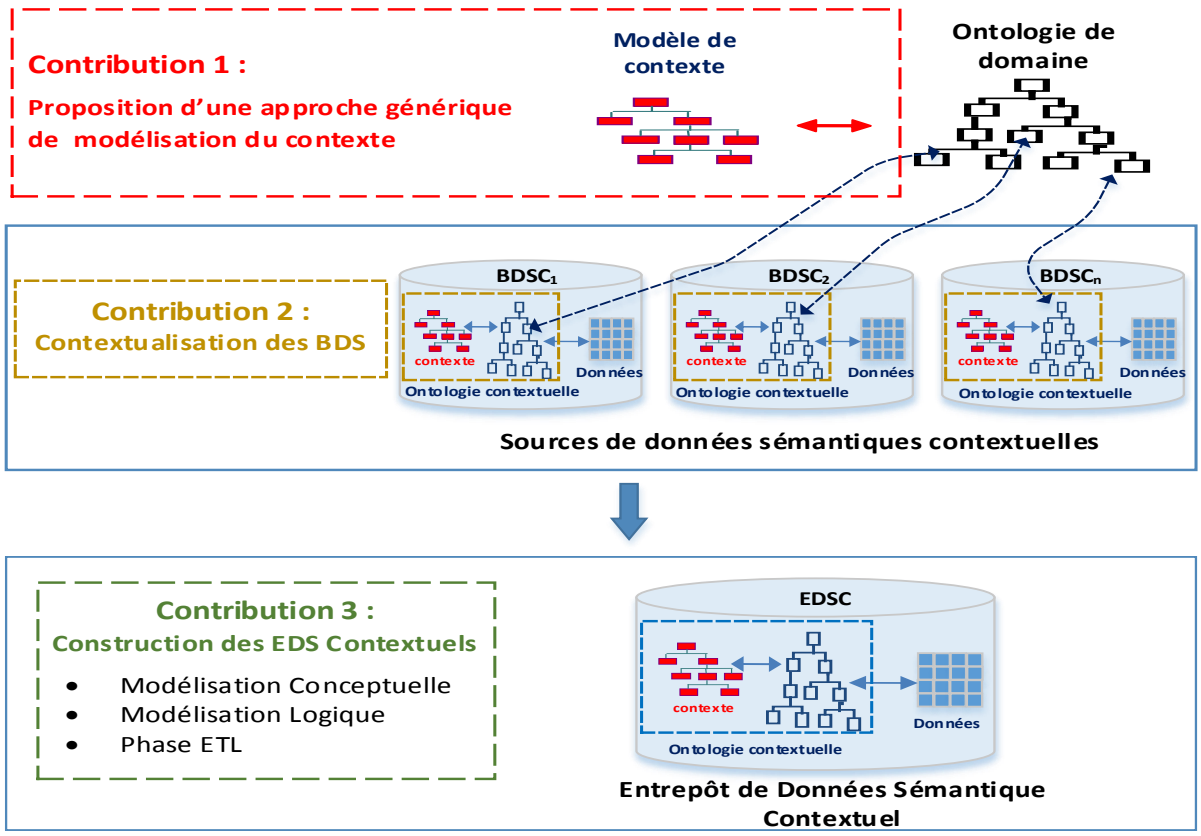


FIGURE 1.5 – Vue globale de la thèse : aperçu des contributions

— **La proposition d'une approche de modélisation générique et formelle du contexte** qui prend en compte à la fois les propriétés et les concepts, contrairement aux approches existantes. Cette contribution repose sur cinq étapes principales. En premier lieu, nous décortiquons la notion de contexte à travers la précision de notre vision propre au domaine de l'ingénierie des données et l'identification des différentes dimensions du contexte. En second lieu, nous proposons une classification des informations contextuelles [3] en plusieurs catégories. En troisième lieu, nous définissons et discutons quelques exigences de modélisation. En quatrième lieu, nous proposons un modèle générique et partageable du contexte que nous décrivons avec le langage de modélisation EXPRESS (ISO 10303-11:1994) [4]. En dernier lieu, nous définissons le lien entre ce modèle de contexte et le modèle d'ontologies.

— **La contextualisation des BDS**. Dans le but d'illustrer l'utilité de notre approche de modélisation du contexte, nous proposons, dans cette deuxième contribution, une approche complète pour la prise en charge totale du Contexte dans les systèmes de BDS. Cela consiste en l'augmentation des BDS par le modèle de Contexte afin de permettre la persistance des informations contextuelles et à la proposition d'un moyen dédié pour les interroger. Pour la réalisation de cette contribution, nous prenons l'exemple du système *OntoDB/OntoQL* développé au sein de notre laboratoire qui représente la BDS *OntoDB* et son langage d'exploitation *OntoQL*. Nous enrichissons le modèle d'OntoDB [58, 210] avec les constructeurs de notre modèle de contexte afin de permettre la persistance des informations contextuelles. Ensuite, nous étendons le langage *OntoQL* afin de proposer un nouvel opérateur spécifique à l'interrogation du contexte.

— **La contextualisation des EDS**. Cette dernière contribution consiste à proposer une méthodologie complète pour la conception des EDS Contextuels en revisitant le cycle de construction des EDS et en projetant notre modèle de contexte sur ses phases les plus importantes à savoir : la modélisation conceptuelle, la modélisation logique et le processus ETL.

3 Organisation du mémoire de la thèse

Ce mémoire de thèse est structuré en deux parties. Après cette introduction générale, la première partie présente notre état de l'art qui se décline en deux chapitres.

Le chapitre 2 introduit, dans un premier temps, la notion de donnée sémantique. Ensuite, il détaille la notion d'ontologie, en présentant sa définition, ses caractéristiques, sa comparaison avec les modèles conceptuels et ses différents formalismes. Enfin, il introduit les concepts fondamentaux relatifs aux bases/entrepôts de données sémantiques et discute le rôle des ontologies

3. La notion du Contexte inclut implicitement la notion d'information et, par conséquent, nous utilisons les deux termes *Contexte* et *Information Contextuelle* indifféremment.

4. http://www.iso.org/iso/catalogue_detail.htm?csnumber=18348

dans les processus de leur construction et de leur exploitation.

Le chapitre 3 présente la notion de contexte, d'une manière générale, puis expose un état des lieux portant sur son utilisation par la communauté informatique avec une focalisation sur les bases/entrepôts de données, l'informatique ubiquitaire et l'Intelligence Artificielle. Ensuite, il détaille les efforts scientifiques qui ont consisté à lier/connecter le contexte aux ontologies. Enfin, il se termine par la présentation de deux analyses qui comparent les principaux travaux proposés dans la littérature consistant à intégrer le contexte dans les systèmes de bases/entrepôts de données.

La deuxième partie de cette thèse comporte trois chapitres correspondant à nos contributions.

Le chapitre 4 présente notre approche de modélisation générique du contexte, et ce indépendamment de l'application à concevoir. Il commence par introduire notre vision de la notion de contexte. Ensuite, il détaille les différentes catégories du contexte que nous avons prises en considération. Enfin, il propose notre approche d'externalisation du contexte de l'ontologie en assurant un lien entre les deux. Ce lien est matérialisé par l'établissement des correspondances entre les concepts du modèle d'ontologie et le modèle de contexte.

Le chapitre 5 détaille notre proposition relative à la contextualisation des bases de données sémantiques. Celle-ci consiste en l'augmentation des bases de données sémantiques avec notre modèle de contexte et à la proposition d'un mécanisme dédié aux interrogations contextuelles. Afin d'expliquer cette proposition, l'exemple du système OntoDB/OntoQL, qui représente la base de données sémantique OntoDB et son langage de requêtes OntoQL, est choisi comme infrastructure de développement. En effet, ce chapitre détaille l'extension du modèle d'OntoDB avec notre modèle de contexte ainsi que l'extension de son langage de requêtes OntoQL afin de prendre en charge le contexte. Il se termine par la présentation des implémentations réalisées en considérant un cas d'étude issu du domaine médical.

Le chapitre 6 présente notre dernière contribution relative à l'intégration du contexte dans le cycle de conception des entrepôts de données sémantiques. D'abord, ce chapitre introduit la formalisation des entrées principales sur lesquelles repose notre approche, à savoir : les ontologies contextuelles, les besoins des utilisateurs et le canevas d'intégration de sources de données sémantiques contextuelles. Ensuite, il détaille les différentes étapes de notre approche de conception en projetant le contexte sur les phases suivantes du cycle de conception d'entrepôt de données : les modélisations conceptuelle et logique, le processus ETL et la modélisation physique. Enfin, un prototype d'outil d'aide à la conception des entrepôts de données sémantiques contextuels, implémentant les différentes étapes de notre approche, est présenté.

Le dernier chapitre conclut ce manuscrit en établissant un bilan récapitulatif de nos contributions et en esquissant un certain nombre de perspectives ouvertes par nos travaux.

Ce mémoire comporte trois annexes. L'Annexe A illustre la représentation en UML de notre modèle de contexte. Les Annexes B et C fournissent les différentes requêtes OntoQL utilisées

dans nos expérimentations.

Liste des publications :

La liste suivante présente les articles publiés dans le cadre de cette thèse :

1. **Barkat, O.**, Extending Semantic Databases to handle Context - An ontology modeling approach. In Proceedings of the 5th International Conference on Model and Data Engineering (MEDI), pages 119-137, Rhodes Island, Greece, September 26-28, 2015. LNCS, Springer.
2. **Barkat, O.**, Bellatreche, L., Linking Context to Ontologies. In Proceedings of the 11th International Conference on Semantics, Knowledge and Grids (SKG), pages 57-64, Beijing, China, Aug 19-21, 2015. IEEE.
3. **Barkat, O.**, Khouri, S., Bellatreche, L., Boustia, N., Bridging Context and Data Warehouses through Ontologies. In Proceedings of the 32nd ACM SIGAPP Symposium On Applied Computing (ACM SAC), Marrakech, Morocco, April 03-07, 2017.

Première partie

État de l’art

Chapitre 2

Explicitation et persistance de la donnée sémantique

Sommaire

1 Introduction 18

2 De la donnée à la donnée sémantique 18

2.1 La donnée 18

2.2 Ontologie 19

2.2.1 Définitions et caractéristiques 20

2.2.2 Ontologies vs modèles conceptuels 21

2.2.3 Taxonomie des ontologies 22

2.2.4 Formalismes Ontologiques 23

3 Des bases de données aux bases de données sémantiques 25

3.1 Bases de données sémantiques 25

3.1.1 BDS pour le web sémantique 26

3.1.2 BDS pour l'ingénierie 26

3.2 Modèles de stockage pour les BDS 27

3.2.1 Modèle de stockage des ontologies 27

3.2.2 Modèle de stockage des instances ontologiques 28

3.3 Architectures des BDS 32

3.3.1 Architecture deux quarts 32

3.3.2 Architecture trois quarts 32

3.3.3 Architecture quatre quarts 33

3.4 Langages d'exploitation des BDS 33

4 Des entrepôts de données aux entrepôts de données sémantiques . . . 34

4.1 Entrepôts de Données 34

4.1.1 Définition et architecture 34

4.1.2 Modélisation multidimensionnelle 35
4.1.3 Cycle de conception des ED 37
4.2 Entrepôts de Données Sémantiques 40
4.2.1 Définition 40
4.2.2 Contribution des ontologies à la construction des EDS 40
5 Conclusion 43

1 Introduction

Les technologies des bases de données joue un rôle crucial au sein des entreprises car elle est au cœur de leurs systèmes d'information. Ceci est principalement dû à leurs capacités de stocker et de gérer un capital important qui est la donnée. Cette technologie a toujours su prendre en charge de nouveaux types de données (traditionnelles, techniques, sémantiques, graphes, etc). A chaque évolution de ces types, des méthodes de conception, d'optimisation et de déploiement ont vu le jour. Cette technologie a eu sa success story. Cette technologie a su répondre aux besoins des applications de type OLTP (OnLine Transaction Processing) et de type OLAP (OnLine Analytical Processing) qui nécessitent des efforts d'intégration de données issues d'un nombre important de sources hétérogènes, réparties et évolutives [114].

Parallèlement au succès des bases de données, le phénomène du Web a changé le monde avec le développement d'Internet et l'explosion du nombre d'utilisateurs de ce dernier. Au début de l'année 2009, ce nombre a atteint le quart de la population mondiale [155]. Cela a poussé la communauté scientifique à structurer le Web en lui associant de la sémantique, ce qui a fait naître le *Web sémantique*. Cette situation a fait émerger le développement des ontologies pour le Web et pour d'autres domaines (comme l'ingénierie [141], la médecine [62], etc). La conséquence directe de ces changements est la naissance des sources de données sémantiques avec des volumes très importants. La technologie de bases de données a su s'adapter afin de satisfaire les besoins du stockage et de la gestion de ces données.

Dans ce chapitre, nous présentons un état de l'art portant sur les efforts déployés par la technologie de base de données pour prendre en compte les Bases de Données Sémantiques (BDS) et des Entrepôts de Données Sémantiques (EDS). Il est organisé en six sections. Dans la section 2, nous introduisons d'abord la notion de la donnée sémantique. Ensuite nous présentons la notion d'ontologie en donnant sa définition, ses caractéristiques, sa comparaison avec les modèles conceptuels et ses différents formalismes. Dans la section 3, nous présentons les BDS en détaillant trois points essentiels : (i) les modèles de stockages, (ii) les différentes architectures et (iii) les langages d'exploitation. Dans la section 4, nous introduisons d'abord les principales notions liées à l'entreposage de données. Ensuite, nous présentons la notion d'entrepôt de données sémantique en analysant la connexion des ontologies avec les différentes phases du cycle de vie d'un entrepôt de données. La section 5 conclut ce chapitre.

2 De la donnée à la donnée sémantique

2.1 La donnée

La donnée est définie comme un symbole qui représente, sur la base de règles d'interprétation externe, une connaissance sur : un fait, un procédé, une notion, un chiffre, une personne, etc [83]. Dans le domaine de la gestion de données, nous classons la donnée, suivant sa nature,

en deux familles principales: (i) la donnée traditionnelle et (ii) la donnée technique.

En effet, afin de répondre aux différents besoins des entreprises, la nature de la donnée a évolué dans le temps. D'abord, les *données traditionnelles*, qui sont utilisées dans le cadre des applications de gestion quotidienne, ont été dépourvues de toute sémantique explicitement spécifiée. Cela a posé certains problèmes, notamment dans le cas où ces données ont été amenées à être partagées et/ou intégrées. Afin de stocker et de gérer ce types de données, plusieurs systèmes de gestion de base de données (SGBD) ont été proposés. Ensuite, dans les années 80, un autre type de données, à savoir la *donnée technique*, a vu le jour. Cette donnée, qui peut être structurée ou non structurée, correspond à toute information permettant de décrire un produit pendant son cycle de vie. La gestion des données techniques est assurée par des systèmes de gestion de données techniques (SGDT). Ceux-ci sont « *des outils logiciels intégrés permettant de consolider et redistribuer l'ensemble des informations de définition du produit, d'en organiser, gérer et contrôler les accès, les modifications, le partage, le groupement, la sécurité, l'approbation des données techniques, créées sous différents formats et d'assurer l'archivage des données techniques dans un environnement hétérogène et distribué* » [95]. L'hétérogénéité des données techniques et la diversité des applications qui les utilisent ont rendu très difficile l'intégration des ces données ainsi que leur échange entre les différents systèmes. Là aussi, comme c'est le cas pour les données traditionnelles, l'explicitation de la sémantique des données est apparue comme un besoin urgent.

Afin de répondre à ce besoin, plusieurs initiatives menées dans le domaine de la représentation de connaissances ont permis de proposer des modèles capables d'expliciter la sémantique des données. Ces modèles, qui permettent de modéliser explicitement et consensuellement les aspects structurels et descriptifs des différents concepts d'un domaine donné, ne sont autres que les *ontologies*. Grâce à la possibilité de référencer une ontologie, les données, que ce soient traditionnelles ou techniques, ont évoluées vers des *données sémantiques* dont la signification est définie explicitement par ce référencement.

Nous distinguons entre deux types de données sémantiques :

— *Les données sémantiques annotées* : ce sont les données dont la signification est définie explicitement par référence à une ontologie ;

— *Les données sémantiques de naissance* : ce sont les instances directes d'une ontologie.

Comme nous nous intéressons dans cette thèse à ce type de données sémantique, nous présentons dans la section suivante les principales notions relatives aux ontologies.

2.2 Ontologie

Nous détaillons dans cette section la notion d'ontologie. Nous commençons par définir cette notion et dégager les caractéristiques qui la distinguent des autres modèles informatiques tels que les modèles conceptuels et les modèles de connaissances. Nous exposons ensuite une caté-

gorisation des ontologies du domaine. Enfin, nous présentons les différents formalismes ontologiques existants.

2.2.1 Définitions et caractéristiques

Plusieurs définitions de la notion d'ontologie ont été proposées dans la littérature. La définition la plus référencée est celle proposée par Gruber [80] qui définit une ontologie comme une : "*spécification explicite d'une conceptualisation* ". Nous retenons aussi la définition proposée par *Pierra* [151] qui, à notre avis, est plus complète : "*une ontologie est une représentation formelle, explicite, référençable et consensuelle de l'ensemble des concepts partagés d'un domaine sous forme de classes, de propriétés et de relations qui les lient*". Cette définition met l'accent sur quatre caractéristiques importantes :

- *Formelle* : l'ontologie est basée sur une sémantique formelle et définie par des langages traitables par machine (vérification automatique de consistance, raisonnement) ;
- *Explicite* : chaque concept de l'ontologie est spécifié explicitement indépendamment de tout point de vue particulier ou tout contexte implicite ;
- *Référençable* : tout concept de l'ontologie est associé à un identifiant permettant de le référencer de manière unique à partir de n'importe quel environnement ;
- *Consensuelle* : l'ontologie est une conceptualisation acceptée et validée par l'ensemble des membres d'une communauté.

Malgré la diversité des modèles ontologiques, ceux-ci reposent sur les notions principales suivantes :

- *Les concepts* (classes ou entités) : ils représentent des ensembles d'individus partageant des caractéristiques communes. Les concepts ontologiques peuvent être classés en deux catégories [80] :
  - *Les concepts primitifs* : ce sont les concepts pour lesquels l'ontologie ne fournit pas de définitions complètes. Ils sont définis grâce au savoir communautaire préexistant ou bien par des documentations contextuelles. Ces concepts primitifs sont utilisés comme une base pour la définition des autres concepts ;
  - *Les concepts définis* : ce sont les concepts dont les définitions sont exprimées par des conditions nécessaires et suffisantes en terme d'autres concepts primitifs ou définis.
- *Les instances* (individus ou objets) : elles représentent l'ensemble des individus du domaine d'ontologie.
- *Les attributs* (propriétés ou rôles) : Ils décrivent et caractérisent les instances des concepts de l'ontologie avec des valeurs caractéristiques ou des associations avec d'autres concepts.
- *Les relations* : ce sont les associations définies entre les concepts de l'ontologie, comme, par exemple, la relation de subsomption "*is-a"* qui sert à organiser les concepts en une hiérarchie.

— *Les axiomes* : ils désignes les affirmations et assertions acceptées comme vraies dans le domaine de l'ontologie. Ils sont utilisés pour contrôler la cohérence de l'ontologie et inférer de nouvelles connaissances.

2.2.2 Ontologies vs modèles conceptuels

Si les ontologies et les modèles conceptuels usuels s'accordent sur *le principe de la modélisation* en conceptualisant tous les deux l'univers du discours sous forme d'un ensemble de concepts (classes) caractérisés par des attributs (propriétés) [67], ils divergent cependant sur *l'objectif de modélisation*. En effet, les modèles conceptuels sont construits selon des *approches prescriptives* qui dépendent des besoins applicatifs. Jean [96] stipule que cette approche comporte plusieurs implications :

— seules les données pertinentes pour l'application cible sont décrites ;

— les données doivent respecter les définitions et les contraintes définies dans le modèle conceptuel ;

— aucun fait n'est inconnu : c'est l'hypothèse du monde fermé [166] ;

— la conceptualisation est faite selon le point de vue des concepteurs et avec leurs conventions ;

— le modèle conceptuel est optimisé pour l'application cible.

Les ontologies, quant à elles, se veulent indépendantes de tout contexte particulier. Elles sont développées suivant des *approches descriptives* qui visent à définir l'ensemble des concepts appartenant à un domaine donné et non pour un objectif applicatif particulier. Contrairement à l'approche prescriptive, (i) les données référençant une ontologie ne doivent pas forcément respecter les contraintes définies dans celle-ci, (ii) les données manquantes sont considérées inconnues (l'hypothèse du monde ouvert [96]) et enfin, (iii) la conception d'une ontologie doit être consensuelle.

Nous constatons que, malgré ces différences, les ontologies peuvent être considérées comme des modèles conceptuels consensuels ou, du moins, comme un premier niveau de conception qui nécessite des réajustements selon les besoins applicatifs. Beaucoup de chercheurs ont, en effet, proposé d'utiliser les ontologies de domaine comme des modèles conceptuels des bases de données. Nous pouvons citer, à titre d'exemples, les travaux de *Roldan Garcia et al.* [171], *Fankam et al.* [67] et *Bellatreche et al.* [11] qui proposent des méthodes de conception de bases de données s'appuyant sur la définition des modèles conceptuels en suivant la même procédure : (i) *extraire* un sous ensemble d'une ontologie de domaine puis (ii) *l'enrichir* avec de nouveaux concepts suivant les besoins applicatifs.

2.2.3 Taxonomie des ontologies

Différents types d'ontologies ont été utilisés par les communautés de recherche. Suivant la manière dont ces ontologies conceptualisent le domaine, deux types sont à distinguer : les Ontologies Linguistiques (OL) et les Ontologies Conceptuelles (OC) [52, 151].

Ontologies Linguistiques (OL)

Ce sont des ontologies qui visent à définir l'ensemble des termes (les mots) utilisés dans un domaine particulier. Elles permettent de fournir une représentation des concepts d'un domaine donné en langage naturel. Les OL décrivent et manipulent aussi les relations entre ces mots telles que la synonymie et l'antonymie ; et comme ces relations sont souvent contextuelles, la supervision d'un expert humain est, en général, nécessaire pour les inférences basées sur ces ontologies.

Les OL ont été largement utilisées dans le domaine d'intégration de sources de données. Nous pouvons citer à titre d'exemple le projet MOMIS [14] qui, à partir de différentes sources de données, utilise une OL pour la construction semi-automatique du schéma global intégrateur. Cependant, ce processus doit être entièrement validé par un expert humain.

Ontologies Conceptuelles (OC)

Ce sont les ontologies qui manipulent des concepts et non des mots. En effet, elle se focalisent sur la définition des concepts d'un domaine particulier. Les OC se divisent en deux catégories suivant le type des concepts manipulés :

— Les Ontologies Conceptuelles Canoniques (OCC) : ce sont les ontologies dont les définitions ne présentent aucune redondance et ne contiennent donc que des concepts primitifs. Les OCC sont très importantes dans le domaine de la conception des bases de données, où la redondance est à exclure, mais aussi dans la création des formats d'échange où chaque information échangée n'est représentée que par une seule et unique définition. *Dublin core* [205] est un exemple normalisé d'ontologies conceptuelles canoniques.

— Les Ontologies Conceptuelles Non Canoniques (OCNC) : ce sont les ontologies qui acceptent dans leur définition les relations d'équivalence entre concepts permettant de représenter les concepts définis. Ces relations d'équivalence sont basées sur deux types d'opérateurs : (i) les *opérateurs de classes*, comme les opérateurs ensemblistes (union, intersection, différence, etc) et les opérateurs de restrictions sur les valeurs des propriétés, et (ii) les *opérateurs de propriétés*, comme les relations algébriques et logiques. Grâce à leur capacité à faire des déductions, les OCNC sont beaucoup utilisées dans le domaine de l'intelligence artificielle.

2.2.4 Formalismes Ontologiques

Les formalismes ontologiques sont des langages qui permettent de définir et de représenter des ontologies. Suivant l'objectif visé, plusieurs formalismes ont été proposés dans la littérature. Dans le domaine de la gestion et l'échange de données, certains d'entre-eux ont été proposés pour décrire des ontologies qui ne supportent que la définition des concepts primitifs, comme le modèle PLIB [151]. D'autres formalismes, tels que les langages du web sémantique (RDF/RDFS [30], DAML+OIL [50], OWL [132], etc) ont été proposés pour la définition des ontologies supportant la définition des concepts définis et offrant ainsi des possibilités de déductions et d'inférences.

Pour le web sémantique

Le formalisme RDF/RDFS : RDF (Resource Description Framework) est un langage de description des informations relatives aux ressources sur le Web. Les informations encodées en RDF sont traitables par machine. Pour identifier les ressources, RDF utilise des URI (Uniform Resource Identifiers). Les déclarations RDF prennent la forme de triplets : (i) un *sujet* désignant la ressource à décrire, (ii) un *prédicat* représentant la propriété appliquée sur la ressource, et (iii) un *objet* représentant la valeur de la propriété. Ne proposant pas de constructeurs pour la définition d'ontologies, RDF a été étendu par un ensemble de constructeurs donnant lieu au modèle RDF-Schéma (ou RDFS).

RDFS permet de construire des concepts, éventuellement définis à partir d'autres concepts, partagés sur le web. Les expressions RDFS sont écrites sous forme de triplets RDF conformément à la sémantique définie en RDFS.

Le formalisme DAML+OIL : DARPA [5] Agent Markup Language (DAML) + Ontology Inference Layer (OIL) est le fruit de la fusion des deux travaux de recherche : (i) *DAML* du ministère de la défense américain qui est un langage de marquage sémantique pour les ressources du Web et (ii) *OIL* développé par la communauté de recherche européenne qui est un langage enrichissant RDFS afin d'offrir de nouvelles primitives pour la définition des classes comme l'union, l'intersection et le complémentaire d'une classe.

DAML + OIL a été utilisé par le World Wide Web Consortium (W3C) [6] comme une base pour le développement du langage OWL.

Le formalisme OWL : OWL est un langage reconnu par le W3C comme le standard de description d'ontologies pour le web sémantique. Inspiré de DAML+OIL, OWL permet une

5. Defense Advanced Research Projects Agency
6. http://www.w3.org/

description complexe des ontologies en offrant un vocabulaire riche et une sémantique formelle pour la définition des classes et des propriétés ainsi que les relations entre les classes. En effet, il dispose d'une panoplie de constructeurs comme l'identité, l'équivalence, l'union, le complément d'une classe, l'intersection, les cardinalités, la symétrie, la transitivité, la disjonction, etc.

OWL se décline en trois versions, OWL Lite, OWL-DL et OWL Full, allant respectivement du moins au plus expressif :

— *OWL-Lite* : syntaxiquement, c'est le langage le plus simple. Il décrit une hiérarchie de classification et offre des mécanismes de contraintes très simples ;

— *OWL-DL* : il est plus expressif qu'OWL-Lite et garantit la complétude des calculs et la décidabilité des raisonnements [51] ;

— *OWL-Full*: en permettant aux utilisateurs de disposer de toute la liberté syntaxique de RDF, OWL-Full est le langage le plus expressif. Cependant, il ne garantit ni la complétude des raisonnements, ni leur décidabilité.

Pour l'e-ingénierie

Le formalisme PLIB : à l'origine, le modèle Parts LIBrary (PLIB) a été conçu pour modéliser, de façon consensuelle, les catalogues de composants industriels afin de permettre leurs échanges entre les différents utilisateurs et fournisseurs. L'objectif principal d'une ontologie PLIB est de définir précisément et de façon concise l'ensemble des catégories et des propriétés primitives caractérisant les objets d'un domaine donné du monde réel [151].

Ce modèle permet de décrire des classes, des propriétés, des domaines de valeurs et des instances tout en définissant la hiérarchisation des classes et des propriétés. C'est un modèle privilégié pour la création des OCC car il ne permet la création d'une nouvelle classe que quand celle-ci ne peut pas être définie par d'autres classes. Chaque concept est identifié de manière unique par un GUI (Globally Unique Identifier). Le modèle PLIB offre un opérateur de modularité (la relation *case-of*) permettant, par exemple, aux concepteurs de définir leurs propres ontologies locales à partir d'une ontologie partagée.

Noyau commun aux différents modèles

La table 2.1, extraite de celle de [99], illustre une étude comparative entre les différents modèles d'ontologies présentés ci-dessus par rapport aux différents constructeurs qu'ils offrent. Nous utilisons les notations + (respectivement −) pour indiquer qu'un modèle d'ontologie supporte totalement (respectivement partiellement) un constructeur donné.

Cette revue des différents formalismes ontologiques nous a permis de constater qu'hormis le fait qu'ils diffèrent par leurs objectifs (orientés gestion et échange de données ou orientés

Constructeurs	**RDFS**	**OWL**	**PLIB**
Ontologie : namespace	+	+	+
Classes	+	+	+
Héritage multiple	+	+	−
Propriétés	+	+	+
Propriétés dépendantes du contexte			+
Types de données :			
• Primitifs	+	+	+
• Classe	+	+	+
• Collections	+	+	+
• Unités de mesure			+
Instances :			
• Identification universelle	+	+	
• Multi-instanciation	+	+	−
• Méta-modélisation	+	+	

TABLE 2.1 – Constructeurs fournis par les modèles d'ontologies

inférence), ils présentent certaines caractéristiques communes. En effet, tous ces formalismes conceptualisent le domaine visé en utilisant les concepts de classes et de propriétés. Ils définissent la hiérarchisation des classes en utilisant la relation de subsomption. Ils permettent d'associer à chaque concept un identifiant qui assure son référencement à partir de n'importe quel environnement. Ils utilisent tous des types de données simples et des collections. En résumé, ces formalismes fournissent des constructeurs communs pour la définition des classes, des propriétés, des types de données et d'instances.

3 Des bases de données aux bases de données sémantiques

La croissance spectaculaire de l'utilisation des ontologies par différentes sortes d'applications dans plusieurs domaines a entraîné une explosion de la quantité de données manipulées. Par conséquent, afin de stocker et gérer efficacement cette forte volumétrie de données, un nouveau type de bases de données, appelé Bases de Données Sémantiques, a été créé.

3.1 Bases de données sémantiques

On appelle Base de Données Sémantique (BDS) ou bien Base de Données à Base Ontologique (BDBO), une source de données qui contient à la fois des ontologies, un ensemble de données et des liens entre ces données et les éléments ontologiques qui en définissent le sens

[153, 55].

En effet, les BDS sont des bases de données qui permettent de spécifier explicitement la sémantique de leurs données. Elles ont les caractéristiques suivantes :

— les ontologies et les données sont stockées au sein de la même base de données et peuvent faire l'objet des mêmes traitements (insertion, mise à jour, interrogation, suppression) ;

— toute donnée est associée à un élément ontologique qui en définit le sens (son référent ontologique) et, de façon réciproque, chaque concept ontologique est associé aux données qui lui correspondent ;

— les ontologies locales aux BDS peuvent référencer des ontologies externes.

Plusieurs BDS académiques et industrielles ont été proposées dans la littérature, que ce soit pour le web sémantique, ou bien pour l'ingénierie.

3.1.1 BDS pour le web sémantique

Parmi les BDS qui ont été proposées pour le web sémantique nous pouvons citer : RDF-SuitePARKA [193, 4], Jena [130, 206], Sesame [33], KAON [27], DLDB [145], RStar [122], OntoMS [147], IBM Sor [121], 3Store [85] et Oracle Semantic [53, 136].

3.1.2 BDS pour l'ingénierie

Contrairement au domaine du web sémantique, très peu de systèmes de BDS ont été proposés pour la gestion des données techniques sémantiques. La solution la plus connue est OntoDB [55, 98] qui a initialement été implantée sur le système *PostgreSQL* avec le modèle d'ontologie PLIB, avant qu'elle soit étendue pour prendre en charge différents modèles ontologiques.

Cette diversité des BDS résulte principalement des trois points suivants :

— *la diversité des formalismes ontologiques :* chaque BDS utilise un modèle d'ontologie particulier. Comme nous l'avons vu dans la sous-section 2.2.4, plusieurs formalismes ont été proposés pour répondre à différents objectifs : PLIB, RDF, DAML, OIL, DAML+OIL, OWL ;

— *la diversité des modèles de stockage :* dans les BDS, plusieurs modèles de stockage sont utilisés pour stocker les ontologies et les données à base ontologique (les instances) ;

— *la diversité des architectures :* les BDSs peuvent utiliser un seul schéma commun ou différents schémas pour stocker l'ensemble des données.

Nous détaillons dans ce qui suit les modèles de stockage et les architectures des BDS.

3.2 Modèles de stockage pour les BDS

A l'inverse des bases de données traditionnelles, qui stockent habituellement leurs modèles logiques selon une représentation relationnelle, les BDS, qui comportent deux parties principales à savoir l'ontologie et les données, utilisent une variété de modèles de stockage pour stocker ces deux parties. Le modèle de l'ontologie et les instances ontologiques peuvent être stockés en utilisant soit le même modèle de stockage soit des modèles de stockage différents.

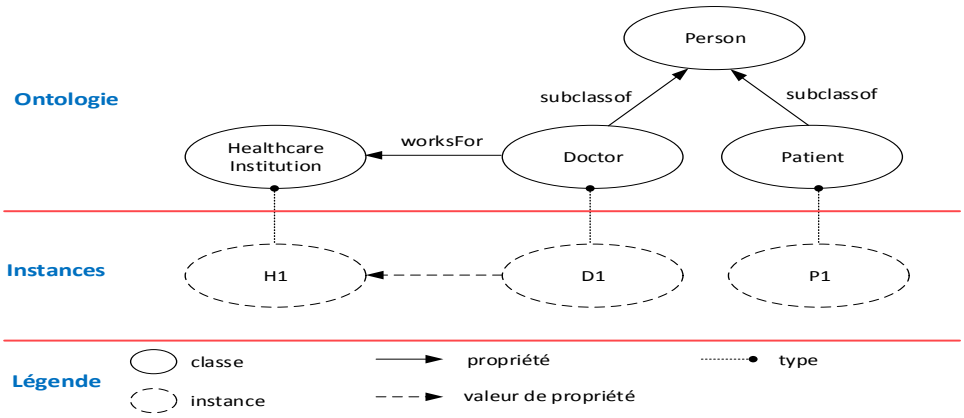


FIGURE 2.1 – Extrait de l'ontologie *Healthcare*

3.2.1 Modèle de stockage des ontologies

Dans la littérature, deux approches principales ont été proposées pour le stockage des ontologies : la représentation générique et la représentation spécifique.

1. **Représentation générique :** également appelée *représentation verticale*, cette représentation consiste en une unique table de triplets composée de trois colonnes (Sujet, Prédicat, Objet). Ces trois colonnes représentent respectivement un élément de l'ontologie, un prédicat et la valeur du prédicat. Ces triplets définissent les éléments d'une ontologie en utilisant les deux formes suivantes :

   — la forme *(E, rdf:type, entité)* : cette forme est utilisée pour indiquer le type de l'élément E. Ce type est une entité définie en RDF-Schéma, par exemple rdfs:Class ou rdfs:Property ;

   — la forme *(E, attribut, valeur)* : cette forme est utilisée pour caractériser l'élément E avec des valeurs d'attributs RDF-Schéma comme par exemple rdf:label ou rdf:comment.

   Les figures 2.1 et 2.2 représentent respectivement un fragment de l'ontologie *Healthcare* et sa représentation générique.

Triplet		
Sujet	Prédicat	objet
http://lias.fr/HealthCareOnto#HealthCareInstitution	rdf:type	rdfs:Class
http://lias.fr/HealthCareOnto#Person	rdf:type	rdfs:Class
http://lias.fr/HealthCareOnto#Patient	rdf:type	rdfs:Class
http://lias.fr/HealthCareOnto#Doctor	rdf:type	rdfs:Class
http://lias.fr/HealthCareOnto#Patient	rdfs:subclassOf	http://lias.fr/HealthCareOnto#Person
http://lias.fr/HealthCareOnto#Doctor	rdfs:subclassOf	http://lias.fr/HealthCareOnto#Person
http://lias.fr/HealthCareOnto#worksFor	rdf:type	rdf:Property
http://lias.fr/HealthCareOnto#worksFor	rdfs:domain	http://lias.fr/HealthCareOnto#Doctor
http://lias.fr/HealthCareOnto#worksFor	rdfs:range	http://lias.fr/HealthCareOnto#HealthCareInstitution
...	...	...

FIGURE 2.2 – Représentation générique du fragment de l'ontologie *Healthcare*

2. **Représentation spécifique :** cette représentation consiste à utiliser le modèle relationnel ou relationnel-objet supporté par le SGBD sous-jacent à la BDS pour représenter son modèle d'ontologies. Chaque BDS définit son propre schéma de tables, de façon ad hoc, en fonction des concepts du modèle d'ontologies qu'elle supporte. Par exemple, le schéma de tables suivant est utilisé pour une ontologie RDFS : class, subclass, domain, range, property, subproperty. Plusieurs systèmes de BDS utilisent cette représentation, tels que Sesame, RSTAR, OntoDB, OntoMS, DLDB, PARKA et KAON.
   La figure 2.3 illustre la représentation spécifique de notre fragment de l'ontologie *Healthcare* en utilisant les trois tables suivantes :
   — *Class* qui permet de stocker les identifiants internes (*id*), les identifiants externes (*URI*), les noms (*label*) et les commentaires (*comment*) des classes qui définissent l'ontologie ;
   — *Property* qui permet de stocker les identifiants internes et externes des propriétés (*id* et *URI*), les noms (*label*) et les commentaires (*comment*) des propriétés. Elle stocke également les domaines et les co-domaines des propriétés (*domain* et *range*) par des références aux classes ;
   — *Subclass* qui permet de stocker la hiérarchie des classes en indiquant pour chaque classe ses super-classes.

3.2.2 Modèle de stockage des instances ontologiques

Dans la littérature, trois approches principales ont été proposées pour la représentation des données à base ontologiques au sein d'une BDS : approche verticale, approche binaire et ap-

Class				SubClassOf	
Id	URI	label	...	sub	sup
1	http://lias.fr/HealthCareOnto#HealthCareInstitution	HealthCareInstitution		3	2
2	http://lias.fr/HealthCareOnto#Person	Person		4	2
3	http://lias.fr/HealthCareOnto#Patient	Patient		...	...
4	http://lias.fr/HealthCareOnto#Doctor	Doctor		...	...

Property				
Id	URI	comment	domain	range
1	http://lias.fr/HealthCareOnto#worksFor	...	4	1
...	...	...	...	...

FIGURE 2.3 – Représentation spécifique du fragment de l'ontologie *Healthcare*

proche horizontale [66]. Nous présentons dans ce qui suit ces trois approches :

1. **Approche verticale** : cette approche représente les instances ontologiques par une unique table de triplets (Sujet, Prédicat, Objet). Ces triplets représentent respectivement l'identifiant d'une instance ontologique, un prédicat et la valeur du prédicat. Chaque instance ontologique *i* est définie par les triplets en utilisant les deux formes suivantes :

   — la forme (*i, rdf:type, Class)* : cette forme est utilisée pour indiquer les classes auxquelles l'instance *i* appartient ;

   — la forme *(i, prédicat, valeur)* : cette forme est utilisée pour caractériser l'instance *i* en lui assignant des valeurs de propriétés.

   Cette représentation facilite l'insertion de nouveaux triplets. Cependant, l'interrogation de la base de données est souvent coûteuse car elle peut nécessiter un nombre élevé d'opérations d'auto-jointures. De plus, dans la plupart du temps, la mise à jour d'une information nécessite la modification de plusieurs triplets.
   Il existe une variante de cette représentation appelée *approche verticale avec ID*. Dans le but de faciliter l'accès aux données, cette variante utilise des tables spécifiques (tables dictionnaires) pour stocker les identifiants (URI) des ressources et toutes les valeurs littérales. La tables des triplets stocke les identifiants des ressources et des littéraux référencés dans les tables dictionnaires. Les figures 2.4 et 2.5 illustrent respectivement la représentation verticale et sa variante avec *ID* des instances de notre fragment d'ontologie *Healthcare*.

2. **Approche binaire :** dans cette approche, les classes et les propriétés auxquelles appartiennent les instances ontologiques sont stockées dans des tables de structures différentes. Suivant l'approche utilisée pour représenter l'héritage, elle se décline en trois variantes : table unique, ISA et NOISA.

   — La variante *table unique* : cette variante utilise une table unique pour stocker toutes les classes de l'ontologie et une table pour chaque propriété. La table des classes

Triplet		
sujet	prédicat	objet
http://lias.fr/HealthCareOnto#P1	rdf:type	http://lias.fr/HealthCareOnto#Patient
http://lias.fr/HealthCareOnto#D1	rdf:type	http://lias.fr/HealthCareOnto#Doctor
http://lias.fr/HealthCareOnto#H1	rdf:type	http://lias.fr/HealthCareOnto#HealthcareInstitution
http://lias.fr/HealthCareOnto#H1	rdfs:comment	CHU de Poitiers
http://lias.fr/HealthCareOnto#D1	http://lias.fr/HealthCareOnto#worksFor	http://lias.fr/HealthCareOnto#H1
...	...	...

FIGURE 2.4 – Représentation verticale des instances du fragment de l'ontologie *Healthcare*

Ressource	
ID	URI
C1	http://lias.fr/HealthCareOnto#Person
C2	http://lias.fr/HealthCareOnto#Patient
C3	http://lias.fr/HealthCareOnto#Doctor
C4	http://lias.fr/HealthCareOnto#HealthcareInstitution
P1	rdf:type
P2	rdfs:comment
P3	http://lias.fr/HealthCareOnto#worksFor
I1	http://lias.fr/HealthCareOnto#P1
I2	http://lias.fr/HealthCareOnto#D1
I3	http://lias.fr/HealthCareOnto#H1
...	...

Litteral	
ID	URI
L1	CHU de Poitiers
...	...

Triplet		
sujet	prédicat	objet
I1	P1	C2
I2	P1	C3
I3	P1	C4
I2	P3	I3
I3	P2	L1
...	...	...

FIGURE 2.5 – Représentation verticale avec ID des instances du fragment de l'ontologie *Healthcare*

est composée de deux colonnes : *URI* pour stocker les identifiants des instances et *ClassID* pour stocker les identifiants des classes.

— La variante *ISA*[196] : cette variante utilise les caractéristiques des SGBD relationnels objets pour représenter l'héritage des classes et des propriétés. Elle consiste à associer à chaque classe et propriété une table spécifique pour représenter leurs instances respectives.

— La variante *NOISA* [196] : cette variante qui n'utilise pas l'héritage des tables, définit séparément les tables des propriétés et des classes sans mises en relation.

Dans les trois variantes, les tables des propriétés sont composées de deux colonnes : *id* pour stocker les identifiants des instances, et *value* pour stocker les valeurs des propriétés correspondantes.

RDFSuite, *Genea*, *IBM SOR* et *DLDB-OWL* sont des exemples de BDS qui utilisent cette représentation.
Cette approche est efficace en terme de temps de réponse pour des requêtes simples. Cependant, elle nécessite plusieurs opérations de jointures pour les requêtes complexes. La figure 2.6 illustre les trois variantes de l'approche binaire.

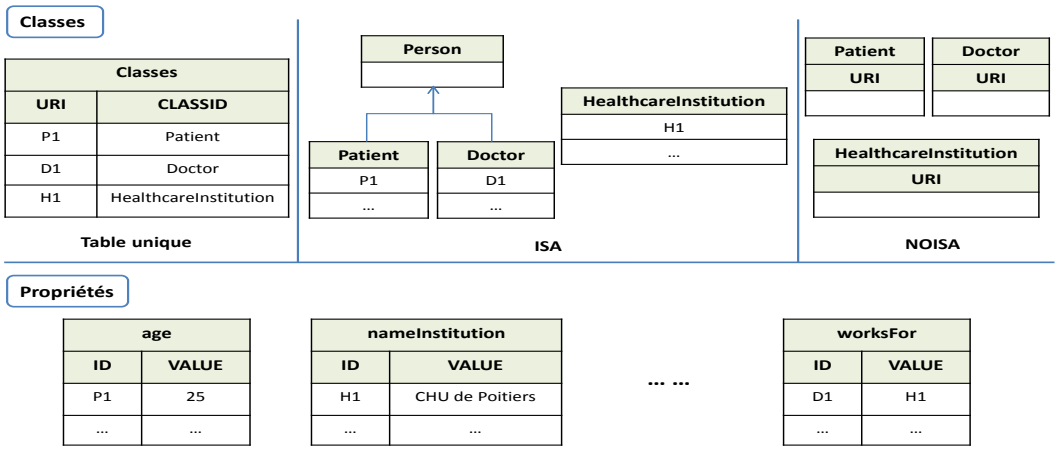


FIGURE 2.6 – Représentation binaire des instances du fragment de l'ontologie *Healthcare*

3. **Approche horizontale** : cette approche est similaire à la représentation traditionnelle utilisée dans les BD conventionnelles. Elle consiste, comme le montre la figure 2.7, à utiliser, pour chaque classe, une seule table relationnelle pour stocker ses instances. Les propriétés de chaque classe représentent les colonnes de la table qui lui est associée. Les BDS *OntoDB* et *OntoMS* utilisent cette représentation.
   Une variante de cette approche consiste à utiliser une seule table relationnelle pour stocker toutes les instances des classes ainsi que leurs valeurs de propriétés.

Patient			
ID	name	age	smoker
P1	Jean	25	Yes
...	...	...	...

Doctor			
ID	name	age	worksFor
D1	Patrick	50	H1
...	...	...	...

HealthcareInstitution	
ID	nameInstitution
H1	CHU de Poitiers
...	...

FIGURE 2.7 – Représentation horizontale des instances du fragment de l'ontologie *Healthcare*

Quelques BDS combinent ces approches, dans ce cas on parle d'*approche hybride*. Nous pouvons citer *Jena2* [206] qui combine les représentations verticale et binaire, ou bien *DLDB* qui combine l'approche binaire et horizontale.

3.3 Architectures des BDS

L'arrivée des BDS a permis de remettre en cause l'architecture traditionnelle d'un SGBD qui consiste en deux parties principales : le contenu et la méta-base. En se basant sur les différents niveaux d'abstraction que les BDS peuvent supporter, trois principales architectures ont été identifiées : (i) architecture de *type I* ou architecture *deux quarts*, (ii) architecture de *type II* ou architecture *trois quarts*, (iii) architecture de *type III* ou architecture *quatre quarts*.

3.3.1 Architecture deux quarts

Cette architecture est similaire aux architectures de bases de données classiques. Elle est composée de deux parties :

— la partie *méta-base* ou *catalogue du système* qui contient l'ensemble des tables système permettant la gestion et le bon fonctionnement ;

— la partie *donnée* qui représente les instances ontologiques et également le schéma de l'ontologie.

Généralement, cette architecture utilise un seul schéma composé d'une unique table de triplets ou une table verticale pour stocker toutes les informations de l'ontologie (la représentation verticale). Elle présente l'avantage d'être simple à mettre en oeuvre et d'effectuer facilement les mises à jour des instances et des propriétés. Cependant, son inconvénient majeur réside dans le fait de tout stocker dans une seule table (modèle et instances). Ainsi, l'exécution des requêtes demande de nombreuses opérations d'auto-jointure sur cette table volumineuse. *Jena2* est une BDS qui utilise ce type d'architecture.

3.3.2 Architecture trois quarts

Cette architecture consiste à stocker les ontologies et leurs instances dans deux schémas différents. Elle est composée de trois parties :

— la partie *ontologie* qui est dédiée au niveau ontologique. Elle est spécifique au formalisme d'ontologie utilisé (OWL, PLIB, RDFS, etc.) et comporte des tables figées pour le stockage des différents concepts ontologiques tels que les classes et les propriétés ;

— la partie *données* et la partie *méta-base* qui jouent le même rôle que dans l'architecture précédente

Cette architecture offre de meilleures performances par rapport à l'architecture *deux quarts*. Cependant, elle souffre de son manque de flexibilité dans la mesure où son schéma de l'ontologie est figé et ainsi ne permet pas l'introduction de concepts issus d'autres modèles d'ontologies. Les systèmes RDFSuite et IBM Sor utilisent cette architecture.

3.3.3 Architecture quatre quarts

Cette architecture a été proposée afin de répondre à l'insuffisance dont souffre l'architecture précédente. En effet, elle l'étend en définissant un schéma supplémentaire appelé *méta-schéma* qui permet de stocker le modèle d'ontologies dans un méta-modèle réflexif. Cette partie *méta-schéma* offre une flexibilité pour l'ontologie en permettant un accès générique au modèle ontologique, l'évolution du modèle d'ontologies et l'introduction de nouveaux concepts issus de différents formalismes ontologiques. Le système *OntoDB* utilise cette architecture.
La figure 2.8 illustre les trois architectures des BDS.

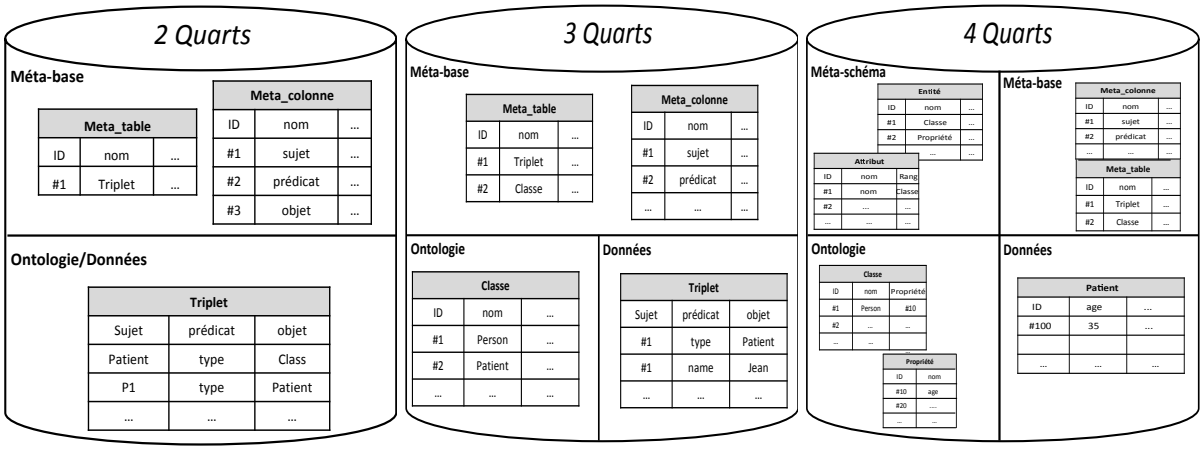


FIGURE 2.8 – Architectures des Bases de Données Sémantiques

3.4 Langages d'exploitation des BDS

Afin de permettre l'exploitation des données à base ontologique stockées dans les BDS sans se préoccuper du modèle de stockage et des détails de mise en oeuvre, une panoplie de langages d'exploitations a été proposée dans la littérature. *Bailey et al.* [8] recensent plusieurs langages de requêtes et les classifient en sept catégories :

1. La famille *SPARQL* : cette catégorie regroupe les langages qui permettent d'interroger des données *RDF* en traitant les triplets. Parmi les langages qui appartiennent à cette famille, nous trouvons par exemple : *SPARQL* [162], *RDQL* [181], *SquishQL* [134] et *TriQL* [40]. Le Consortium W3C [7] recommande l'utilisation du langage *SPARQL* pour les requêtes sémantiques. Ce dernier est utilisé dans plusieurs BDS.
2. La famille *RQL* : cette catégorie concerne les langages d'interrogation qui prennent en compte les données et le schéma d'ontologie lors des interrogations. Ils supportent le modèle de données *RDF* qui impose deux contraintes essentielles : (1) l'interdiction des

7. https://www.w3.org

cycles dans les relations de subsomption, et (2) l'obligation de la définition du domaine et du co-domaine pour chaque propriété. Cette famille regroupe les langages RQL [103], eRQL [197], SeRQL [32] et OntoQL [97].

3. La famille des *langages inspirés de XPath, XSLT ou XQuery* : cette famille regroupe les langages qui s'inspirent ou étendent un langage de requête XML, comme par exemple : XQuery for RDF [169], XSLT for RDF [191], Versa [143], RPath [128], etc.
4. La famille *Metalog (Anglais contrôlé)* : Cette catégorie fait référence à Metalog [125] qui est un système pour l'interrogation et le raisonnement dans le Web Sémantique. Ce système diffère des autres langages RDF par sa syntaxe qui est le langage naturel contrôlé (Anglais) et par le fait qu'il combine l'interrogation et le raisonnement.
5. La famille des *langages avec règles réactives* : cette famille regroupe les langages qui offrent la possibilité de définir des règles de vérification dans leurs requêtes comme : Algae [161], iTQL [207] et WQL [119].
6. La famille des *langages déductifs* : cette famille regroupe des langages basés sur un langage de règles de déduction et de transformation pour RDF tels que : N3QL [18], R-DEVICE [9], TRIPLE [187], etc.
7. La famille *autres langages RDF* : cette catégorie regroupe les langages de requêtes RDF qui ne font partie d'aucune des familles énumérées ci-dessus, comme RDF-QBE [167] et RDFQL [93].

4 Des entrepôts de données aux entrepôts de données sémantiques

Dans cette section, nous introduisons d'abord les principales notions liées à l'entreposage de données. Ensuite, nous présentons la notion d'entrepôt de données sémantique.

4.1 Entrepôts de Données

Les Entrepôts de Données ont été proposés pour répondre au besoin des entreprises en terme d'analyse des quantités croissantes des données qu'elles gèrent. Dans cette section, nous définissons les concepts importants inhérents à l'entreposage de données.

4.1.1 Définition et architecture

Un ED est défini, selon W.H Inmon [92], comme : « *une collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour supporter un processus d'aide à la décision* ». Cette définition spécifie six caractéristiques d'un ED :

— *Orientées sujet :* à l'inverse des systèmes d'information classiques qui organisent leurs données suivant les besoins applicatifs des entreprises, les ED organisent leurs données par sujets (production, clients, diagnostics, etc) ;

— *Intégrées :* les données de l'ED proviennent de différentes sources présentant souvent des problèmes d'hétérogénéité syntaxique et/ou sémantique. Le but de l'intégration est de résoudre ces problèmes afin d'uniformiser les données peuplant l'ED ;

— *Non volatiles :* sauf le cas du rafraichissement, les données de l'ED ne peuvent pas être ni modifiées ni supprimées. Elles ne sont accessibles qu'en mode consultation ;

— *Historisées :* l'évolution des données de l'ED doit être prise en compte en les associant à un référentiel temporel ;

— *Résumées :* les données doivent être agrégées et réorganisées afin de faciliter leur analyse et ainsi favoriser le processus de prise de décision ;

— *Disponibles pour l'interrogation et l'analyse :* la manipulation et l'analyse des données de l'ED peuvent être réalisées en utilisant différents outils : requêtes, outils OLAP [8], outils de fouille de données, etc.

L'architecture d'un système d'ED se compose de trois composantes principales :

1. *Les sources de données* : ce sont les différentes sources, souvent hétérogènes et distribuées, qui contiennent les informations à intégrer dans l'ED. Elles peuvent être des BD, des fichiers de données, des fichiers XML, des sources externes à l'entreprise, etc.
2. *L'intégration et le stockage :* c'est le rôle du *processus ETL* [9] qui sert à extraire les données des sources, les nettoyer et les charger dans l'ED. Afin de répondre aux besoins spécifiques des utilisateurs, un ED peut comporter plusieurs magasins de données (data-marts) contenant chacun des extraits de l'ED destinés à une application distincte.
3. *L'exploitation :* l'exploitation de l'ED est possible grâce au (i) *serveur OLAP* qui permet d'accéder à ses données et de traduire les requêtes des utilisateurs en requêtes ED afin de fournir les résultats à des outils d'aide à la prise de décision ; et à des (ii) *outils de front end* qui, suivant les besoins des décideurs, structurent les données selon différentes formes : tableaux, graphiques, statistiques, etc.

4.1.2 Modélisation multidimensionnelle

La modélisation multidimensionnelle est une approche de modélisation qui a été proposée par *Kimball et al.* [111] pour remplacer la modélisation traditionnelle relationnelle qui s'avérait inappropriée pour le support de la prise de décision et les analyses en ligne de type OLAP [49].

8. OnLine Analytical Processing : traitement analytique en ligne

9. Extract, Tansform and Load : processus regroupant des opérations d'extraction de données des sources, de transformation des données suivant les formats cibles et du chargement de ces données dans l'entrepôt

Définition

La modélisation multidimensionnelle consiste à offrir une vision multidimensionnelle des données en mettant en évidence les sujets analysés et les différentes perspectives de l'analyse. Cette technique de modélisation considère les données analysées comme des points dans un espace à différentes dimensions. Elle renferme deux éléments fondamentaux :

— *les faits* : ils représentent les sujets de l'analyse (par exemple les séjours des patients). Ils sont décrits par plusieurs *mesures* qui leur fournissent des descriptions quantitatives. Ces mesures peuvent être atomiques ou dérivées (pour les séjours : durée moyenne de séjours, taux d'utilisation des lits, etc).

— *les dimensions* : elles représentent, sous forme de listes hiérarchisées, les perspectives de l'analyse (par exemple, les séjours des patients peuvent être analysés selon les dimensions : temps, services, pathologies, etc).

La figure 2.9 illustre un exemple de *cube de données* permettant d'analyser les durées moyennes de séjours des patients (représentées par les cellules du cube) selon les dimensions suivantes : temps, service et pathologie.

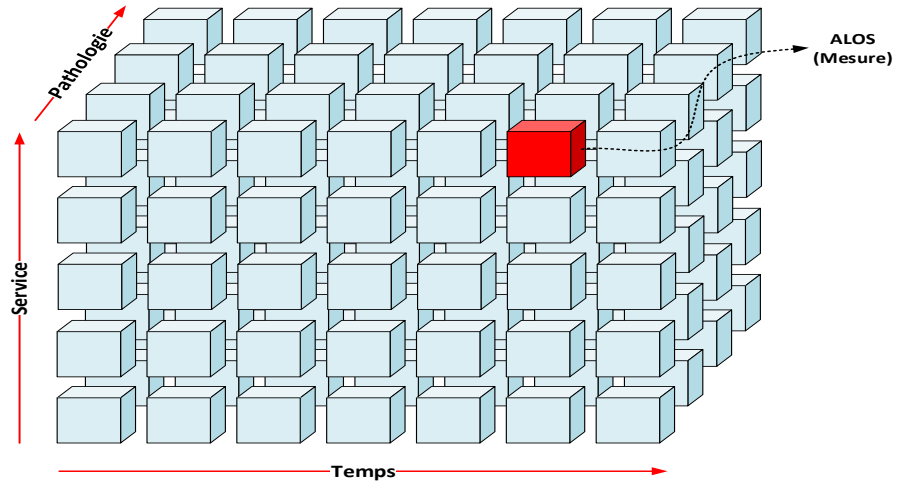


FIGURE 2.9 – Cube de données à trois dimensions

Modèles en étoile, en flocon de neige et en constellation

Il existe trois types de modèles multidimensionnels :

— Schéma en étoile (Star schema) : cette structure, qui ressemble visuellement à une étoile, est constituée d'un fait central et de ses dimensions.

— Schéma en flocon de neige (Snowflake schema) : ce schéma est une variante du schéma en étoile. Il consiste à éclater les dimensions de ce dernier en sous-hiérarchies.

— Schéma en constellation de faits (Multi-star schema) : ce schéma consiste à représenter plusieurs faits qui se partagent les mêmes dimensions.

La figure 2.10 illustre ces trois structures de modélisation multidimensionnelle.

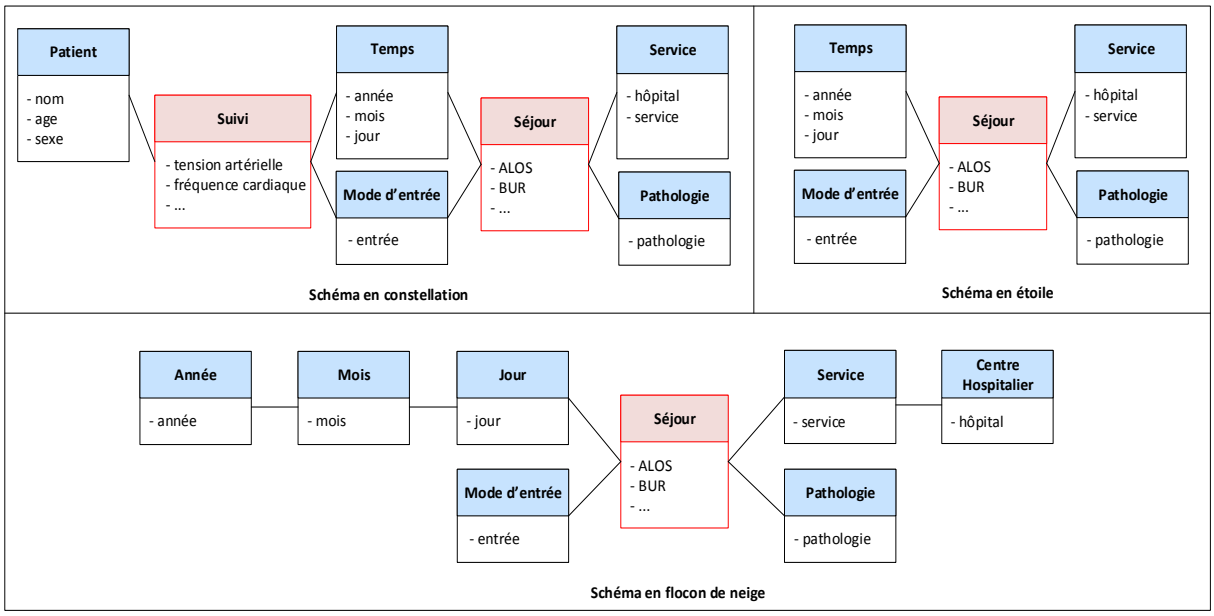


FIGURE 2.10 – Schémas en étoile, en flocon de neige et en constellation

4.1.3 Cycle de conception des ED

Le cycle de vie d'un ED inclut trois phases principales [111] :

— *Planification* : comme son nom l'indique, cette étape consiste en une phase de préparation de terrain pour la construction de l'ED. Elle se compose de trois taches :
  — fixer les objectifs et les finalités de l'ED à construire ;
  — évaluer techniquement et économiquement la faisabilité et la rentabilité de l'ED ;
  — identifier les utilisateurs de l'ED ainsi que leurs rôles.

— *Conception et implémentation* : cette phase consiste à concevoir le schéma de l'ED et à préparer les ressources nécessaires à son implémentation et à son déploiement ;

— *Maintenance et évolution* : cette phase concerne l'optimisation et la mise à jour périodique de l'ED en fonction des changements pouvant se produire au niveau des sources de données ou des besoins des utilisateurs.

Au vu de son importance, la phase de *conception* est divisée en cinq autres principales phases [79] : la définition des besoins, la modélisation conceptuelle, la modélisation logique, le processus ETL et la modélisation physique. Ces dernières constituent le *cycle de conception* de l'ED.

Définition des besoins

Comme dans tout projet informatique, la *définition des besoins* est une étape clé dans les projets de construction des ED. Elle permet d'identifier les différentes fonctions de l'entrepôt ainsi que l'ensemble des informations qui doivent être disponibles [36]. Il existe deux types de besoins : les besoins *fonctionnels* qui concernent les services et les informations que l'entrepôt doit fournir et les besoins *non fonctionnels* [77] qui concernent les contraintes du système et les exigences implicites auxquelles il doit répondre comme la sécurité, la flexibilité, etc.

La définition des besoins comporte quatre activités principales :

— *La collecte des besoins* : elle consiste à spécifier les besoins selon l'une des trois approches suivantes : (i) la collecte axée sur les *données* qui consiste à identifier les besoins à partir de l'ensemble des données disponibles au niveau des sources, (ii) la collecte axée sur les *utilisateurs* qui met l'accent sur les différents utilisateurs qui expriment leurs exigences et (iii) la collecte axée sur les *objectifs* qui se concentre sur la stratégie de l'entreprise en impliquant les cadres directeurs de cette dernière dans le processus de conception.

— *L'analyse des besoins* : cette étape, comme son nom l'indique, consiste à analyser les besoins recueillis afin de détecter les différents conflits possibles : besoins contradictoires, besoins complémentaires, etc.

— *La validation des besoins* : cette étape consiste à garantir, en présence des différents utilisateurs concernés, que les besoins collectés répondent aux besoins réels de l'entreprise.

— *Le suivi de l'évolution des besoins* : elle consiste à gérer l'évolution des besoins des utilisateurs et étudier son impact sur l'ED.

Modélisation conceptuelle

La modélisation conceptuelle consiste à définir le schéma conceptuel multidimensionnel de l'ED. Ce dernier est soit élaboré à partir des besoins des utilisateurs, soit défini comme une vue sur les données des sources.

Modélisation logique

La modélisation logique consiste à traduire le modèle conceptuel de l'ED en un modèle logique qui tient compte des spécificités du modèle d'implémentation choisi. Il existe trois approches de représentations logiques :

— L'approche *Multidimensional On-Line Analytical Processing (MOLAP)* : cette approche consiste en une représentation des données sous la forme d'un tableau multidimensionnel (un cube). L'intérêt principal de cette approche réside dans sa performance en terme de temps d'accès. Cependant, elle présente les limitations suivantes : (1) la non disponibilité

des langages d'interrogation nécessitant la redéfinition des opérations pour manipuler les structures multidimensionnelles et (2) la complexité des mises à jours.

— L'approche *Relational On-Line Analytical Processing (ROLAP)* : cette approche consiste en une représentation relationnelle du modèle de l'ED. Chaque fait et chaque dimension sont représentés respectivement par une table de fait contenant des attributs représentant les mesures et les clés étrangères de chacune des tables de dimension et une table de dimension se constituant d'attributs représentant les paramètres de dimension. Les avantages majeurs de cette approche sont : (1) l'utilisation d'un standard bien maîtrisé (le modèle relationnel) et (2) la grande capacité de stockage. La modélisation ROLAP utilise les trois schémas présentés précédemment : le schéma en étoile, le schéma en flocon de neige et le schéma en constellation.

— L'approche *Hybrid On-Line Analytical Processing (HOLAP)* : cherchant à bénéficier des avantages des deux approches citées précédemment, cette approche consiste à représenter, d'une manière transparente à l'utilisateur, les données qui sont fréquemment utilisées dans un tableau multidimensionnel et représenter les autres données dans un schéma relationnel.

Processus Extract-Transform-Load (ETL)

Le processus ETL représente la phase la plus importante du cycle de vie de l'ED. En effet, il est très coûteux en termes de temps [200] et de ressources [184]. Le processus ETL consiste en trois phases :

— **Extraire** les données à partir des différentes sources hétérogènes et les propager dans une zone temporaire de préparation appelée *Data Staging Area (DSA)* où seront effectuées des manipulations de nettoyage.

— **Transformer** les données récupérées afin qu'elles soient exploitables. Cela consiste à les nettoyer et les homogénéiser.

— **Charger** les données transformées dans l'ED.

Plusieurs outils ETL ont été proposés, que ce soit dans le monde industriel comme Microsoft Data Transformation Services (DTS), Oracle Warehouse Builder (OWB), IBM Data Warehouse Center, Informatica PowerCenter, Talend open studio et Pentaho Data Integration (PDI) ou bien dans le monde académique comme AJAX [73], ARKTOS [201], HumMer [138] et Potter's Wheel [164].

Modélisation physique

Cette dernière phase correspond à l'implémentation physique du modèle logique de l'ED. Durant cette phase, l'administrateur choisit les techniques et structures d'optimisation adéquates

qui doivent assurer les performances optimales en termes d'accès à l'ED. Quatre tâches principales doivent être réalisées :

1. *Choisir les structures d'optimisation* parmi plusieurs existantes : la fragmentation, les index, les vues matérialisées, la compression, etc. Cette tâche est un problème complexe qui dépend de la charge des requêtes à optimiser et exige une certaine expertise ;
2. *Choisir le mode de sélection* : l'administrateur peut utiliser une seule structure d'optimisation ou bien en combiner plusieurs [22] ;
3. *Développer des algorithmes de sélection* : ces algorithmes peuvent être *simples* ou *lourds*. Les algorithmes du premier type sont faciles à mettre en oeuvre mais il sont d'une faible efficacité (eg. la gestion du buffer [48]). Les autres sont plus efficaces mais leur temps d'optimisation est élevé (eg. la fragmentation horizontale [42] ;
4. *Valider les solutions d'optimisation* : le déploiement sur un environnement réel est nécessaire afin d'évaluer les performances effectives des solutions choisies. Il existe plusieurs outils qui offrent des conseils et des recommandations de solutions d'optimisation, comme : SQL Access Advisor [113], DB2 Design Advisor [211] et Parinda [123].

4.2 Entrepôts de Données Sémantiques

Nous consacrons cette sous-section à la présentation des entrepôts de données sémantiques.

4.2.1 Définition

Les ontologies, de part leur caractère consensuel, ont permis à la communauté de recherche sur les systèmes d'intégration de données (SID) d'en bénéficier pour connecter les différentes conceptualisations correspondant aux différentes sources à une ontologie de domaine. Cette caractéristique commune a été ensuite exploitée par les académiques ainsi que les industriels afin de définir des SID où les ontologies contribuent à la résolution des différents conflits syntaxiques et sémantiques identifiés dans les sources de données. Deux architecture principales de SID à base ontologique ont été alors proposées [12] : (i) dans la première architecture, les ontologies de domaine jouent le rôle du schéma global du SID [6], (ii) dans la seconde architecture, chaque source est associée à une ontologie locale faisant référence à une ontologie globale et des mappings sont définis entre les ontologies globale et locales [202]. Ces travaux ont ouvert la voie à l'utilisation des ontologies de domaine dans les ED.

4.2.2 Contribution des ontologies à la construction des EDS

En examinant la littérature dans le domaine de la conception et la construction des systèmes d'entreposage de données, nous avons constaté que les chercheurs se sont activement intéressés à la connexion des ontologies avec les différentes phases du cycle de vie de l'ED. *Pardillo*

et al. [146], sans avoir proposé une approche de conception concrète, décrivent comment la conception des ED peut tirer profit des ontologies.

Afin de présenter les autres travaux existants, nous les avons classifiés, comme le montre la figure 2.11, selon les trois niveaux du cycle de vie des ED suivants : les sources de données, la conception de l'ED et l'exploitation de l'ED.

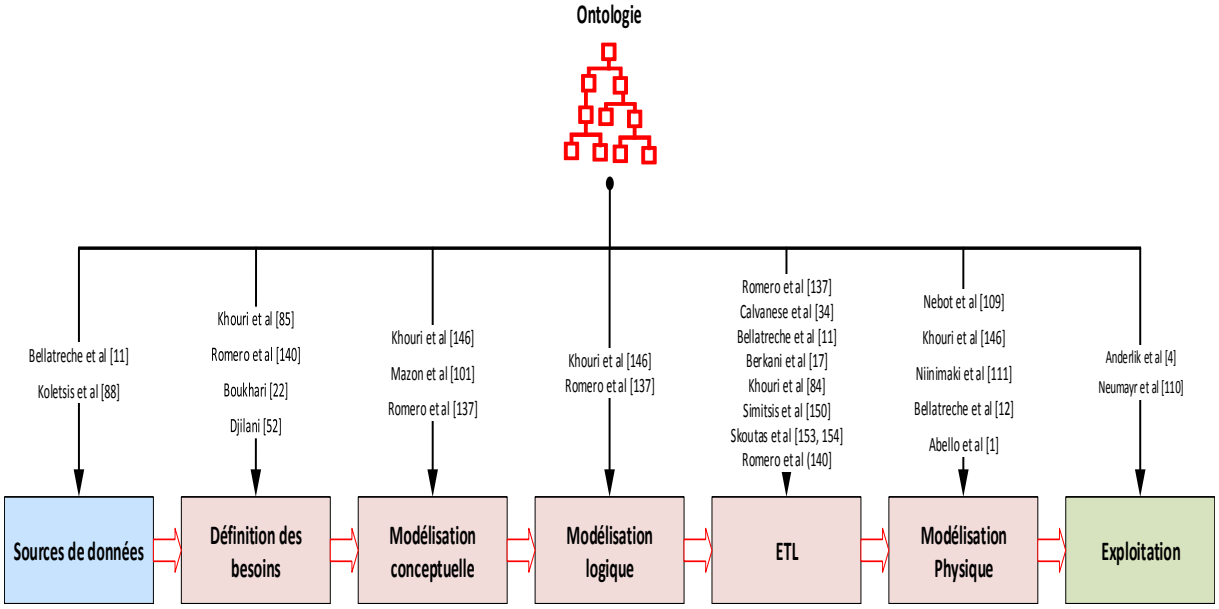


FIGURE 2.11 – Projection de l'ontologie sur les phases du cycle de vie d'un entrepôt de données

Au niveau des sources de données

Les problèmes liés à l'hétérogénéité des sources de données représentent un obstacle qui rend le processus de leur intégration complexe. Afin de pallier ces problèmes, quelques travaux de recherche ont proposé d'utiliser les ontologies au niveau des sources de données afin d'expliciter leur sémantique et de les préparer ainsi pour un processus d'intégration moins compliqué.

Parmi ces travaux, quelques-uns proposent d'utiliser l'architecture des BDS qui intègre les ontologies dans les SGBD [12]. D'autres utilisent des techniques d'annotation sémantique de données qui consistent à enrichir les données des sources en les associant avec les éléments sémantiques d'une ontologie [112].

Au niveau du cycle de conception de l'ED

Beaucoup de travaux se sont concentrés sur les différentes phases du cycle de conception des ED:

— **Définition des besoins :** d'une manière générale, les ontologies ont été utilisées dans cette phase comme une référence commune qui sert à unifier les besoins dans le contexte des environnements distribués (comme le cas des entreprises étendues) où la diversité des acteurs favorise l'hétérogénéité des vocabulaires et des langages de modélisation utilisés pour définir ces besoins. Quelque travaux ont proposé d'utiliser les ontologies pour la définition et la spécification des besoins [107, 175]. D'autres les ont utilisées pour l'analyse et le raisonnement [23, 59].

— **Modélisation conceptuelle :** le formalisme d'ontologies est considéré comme un modèle conceptuel qui décrit un domaine d'une manière abstraite indépendamment des détails d'implantation. En outre, l'ontologie est une conceptualisation basée sur une théorie formelle qui permet d'effectuer un certain niveau de raisonnement automatique sur les concepts et les individus [100]. Ces deux caractéristiques ont encouragé différents travaux à considérer les ontologies dans la phase de modélisation conceptuelle des ED. Certains les ont utilisées comme un premier niveau de conception [182]. D'autres les ont utilisées pour annoter les concepts du schéma conceptuel de l'ED avec leurs rôles multidimensionnels [182, 129, 172].

— **Modélisation logique :** cette phase définit la traduction du schéma conceptuel obtenu en un schéma logique. De nombreux travaux proposent la traduction d'une ontologie (représentant le schéma conceptuel) en des schémas relationnels suivant différentes approches [182, 172].

— **Phase ETL :** la phase ETL a été largement étudiée en raison de sa complexité et de sa consommation du temps. L'utilisation initiale des ontologies dans la conception des systèmes d'ED a été empruntée aux SID à base ontologique. Ceux-ci utilisent en effet les ontologies pour identifier et comprendre la sémantique des sources de données et du magasin de données cible. Les premiers travaux exploitant les ontologies pour la conception des systèmes d'ED ont utilisé ces dernières pour décrire le schéma global de l'ED [172] ou bien pour décrire à la fois le schéma global et les schémas des sources [37].

  Divers travaux [12, 17, 106] ont considéré les BDS comme sources de données candidates au processus d'entreposage de données ou comme des architectures implémentant le système ED cible. Certaines études ont prouvé que le processus d'intégration utilisant les BDS selon certains scénarios spécifiques d'intégration peut être entièrement automatisé [12].

  Différents autres travaux ont étendu l'utilisation des ontologies pour : (i) la proposition d'une description textuelle d'une modélisation ETL conceptuelle [186], (ii) l'automatisation de la conception du processus ETL [189, 190] par l'annotation des schémas sources et du schéma cible par des expressions complexes construites à partir de concepts d'une ontologie de domaine et (iii) la proposition d'une approche de modélisation conceptuelle du processus ETL [175].

— **Modélisation physique :** quelques travaux ont décrit la modélisation physique des EDS suivant l'architecture du SGBD utilisé : SGBD classiques [139] ou SGBD sémantiques

[182]. Certaines études [142] ont défini un processus ETL utilisant une ontologie générique comme une ontologie de haut niveau afin d'automatiser la construction des systèmes OLAP. D'autres travaux ont utilisé les ontologies pour la définition des structures d'optimisation qui représente une tâche importante de la conception physique [13]. D'autres études ont proposé de concevoir des systèmes d'ED, non pas dans le monde fermé classique, mais en utilisant des espaces sémantiques ouverts où les données sont externes, souvent non structurées, dynamiques et éventuellement en streaming [1].

Au niveau de l'exploitation de l'ED

Quelques travaux ont proposé d'utiliser les ontologies dans la phase d'exploitation des ED. Anderlik et al [5] ont proposé d'exploiter les niveaux de subsomption entre les concepts ontologiques pour redéfinir des opérations OLAP. *Neumayr et al.* [140] ont introduit une approche pour le raisonnement sur des ontologies multidimensionnelles en utilisant *Datalog*.

5 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art portant sur deux notions : les BDS et les EDS.

Dans la première partie de ce chapitre, après avoir introduit la notion d'ontologie, nous avons fait un tour d'horizon des BDS qui ont été proposées principalement pour (i) gérer des grandes quantités de données et de (ii) répondre au besoin de représenter consensuellement et explicitement la sémantique de ces données. Nous avons commencé par définir les BDS. Ensuite, nous avons détaillé les trois axes principaux qui font la diversité des BDS : (1) les modèles de stockage, (2) les architectures des BDS et (3) les langages d'exploitation. Cette revue des ontologies et des BDS nous a permis de constater que cette nouvelle génération de BDS s'appuie sur une vision classique de l'ontologie où les concepts sont représentés d'une manière unique et mono-contextuelle. Nous estimons que les modèles des BDS doivent être revus afin d'inclure la notion de contexte.

La deuxième partie de ce chapitre a été consacrée à l'étude des différentes approches proposées dans la littérature afin d'exploiter les ontologies dans le processus de conception des systèmes d'ED. Cette étude nous a permis de constater que les chercheurs se sont intéressés à la projection des ontologies sur l'ensemble des phases du cycle de vie des ED mais seulement de manière isolée et sélective. Le deuxième constat que nous avons fait rejoint notre discussion précédente concernant l'aspect classique mono-contextuel des ontologies sur lequel est basée l'intégralité des approches étudiées. Nous estimons que l'hétérogénéité et la répartition des sources de données ainsi que la divergence des points de vue et des profils des utilisateurs des systèmes d'ED nécessitent la prise en charge de la notion du contexte dans la conception de tels systèmes.

Nous détaillerons dans le chapitre suivant la notion de contexte et nous présenterons une revue de littérature des différentes propositions connectant le contexte avec les systèmes des bases/entrepôts de données.

Chapitre 3

Explicitation et persistance du contexte

Sommaire

1 Introduction 47

2 Comprendre la notion de contexte 47

2.1 Définitions du contexte 47

2.1.1 Ce que disent les dictionnaires 48

2.1.2 Ce que dit la littérature 48

2.2 Catégorisation du contexte 49

2.2.1 Catégorisation conceptuelle 50

2.2.2 Catégorisation opérationnelle 50

2.3 Modélisation du contexte 52

3 Couplage entre le contexte et l'ontologie 55

3.1 Ontologies représentant la dépendance au contexte 55

3.2 Ontologies contextualisées 56

3.3 Ontologies de multi-représentation 57

3.4 Synthèse 58

4 Contexte et systèmes de stockage 59

4.1 Contexte et Bases de Données classiques 59

4.1.1 Approches existantes 59

4.1.2 Étude comparative et synthèse 60

4.2 Contexte et Entrepôts de Données 62

4.2.1 Approches existantes 62

4.2.2 Étude comparative et synthèse 63

5 Conclusion 64

1 Introduction

La notion de contexte a initialement émergé dans divers domaines de recherche comme la psychologie et la philosophie [44], et c'est à partir des années 90, que les chercheurs dans le domaine de l'informatique ont commencé à s'y intéresser. Plusieurs définitions et interprétations de cette notion ont été proposées suivant les différentes perspectives et particularités des communautés de recherche.

Dans le domaine des systèmes d'information, l'ère du *Big Data*, qui caractérise notre monde actuel, a submergé les différents acteurs des SI par de grandes quantités de données provenant de nombreuses sources diverses et hétérogènes. C'est là que la notion de contexte a trouvé toute son utilité. En effet, afin que ces données, qui représentent un atout considérable pour les entreprises, puissent être correctement exploitées, elles nécessitent d'être reliées à leurs contextes, tels que les connaissances expertes du domaine, les profils des utilisateurs, les dimensions spatio-temporelles, les unités de mesures, etc. Ce besoin a poussé les chercheurs à s'intéresser à la prise en charge du contexte dans les modèles conceptuels et les modèles de connaissances ainsi que dans les différents systèmes de stockage de données.

Nous proposons dans ce chapitre un état de l'art sur la notion de contexte. Nous y présentions les différents approches qui ont été proposées dans la littérature pour la modélisation du contexte et pour sa prise en charge dans les ontologies ainsi que dans les différents systèmes de stockage de données.

Ce chapitre est organisé en cinq sections. Dans la section 2, nous nous concentrons sur la notion de contexte en présentant et en discutant les différentes définitions, catégorisations et approches de modélisation proposées dans la littérature. Dans la section 3, nous présentons les travaux qui ont permis de connecter les deux notions : contexte et ontologie. Dans les sections 4, nous analysons les approches existantes qui ont proposé de prendre en charge le contexte dans les systèmes de base de données et des entrepôts de données. La section 5 conclut ce chapitre.

2 Comprendre la notion de contexte

2.1 Définitions du contexte

Le *contexte* a été utilisé dans plusieurs disciplines telles que la psychologie, la philosophie et les langages naturels [10, 115]. Dans le domaine de l'informatique, de nombreuses définitions du contexte ont été proposées par plusieurs communautés : Intelligence Artificielle [28], Informatique Diffuse [179], Recherche d'Informations [69], Bases de Données [19], etc. Dans cette sous-section, nous commençons par donner quelques définitions figurant dans des dictionnaires de référence pour cette notion de contexte. Ensuite, nous présentons en détail celles proposées par les différentes communautés de recherche.

2.1.1 Ce que disent les dictionnaires

— Le dictionnaire Larousse [10] définit le contexte comme « *un ensemble de circonstances dans lesquelles se produit un évènement, se situe une action* ».

— Le dictionnaire en ligne (*le-dictionnaire.com*) [11] le définit comme « *un ensemble des circonstances entourant un événement* ».

— L'encyclopédie Wikipédia [12] le définit comme suit : « *Le contexte d'un événement inclut les circonstances et conditions qui l'entourent* ».

— Le Centre National de Ressources Textuelles et Lexicales (CNRTL) [13] définit le contexte comme suit : « *Ensemble de circonstances liées, situation où un phénomène apparaît, un événement se produit* ».

Nous remarquons que ces définitions partagent toutes la même idée, à savoir que le contexte est un ensemble de circonstances associées à un évènement.

Nous abordons dans la sous-section suivante la notion de contexte dans le domaine de l'informatique en présentant les différentes visions proposées par les communautés de recherche.

2.1.2 Ce que dit la littérature

Les chercheurs ont commencé à s'intéresser au contexte dans les années 90. En effet, dans le domaine de l'Intelligence Artificielle, *McCarthy* [131] l'a introduit comme une entité mathématique abstraite avec des propriétés. *Brézillon* [29] l'a défini comme un ensemble de conditions pertinentes et d'influences qui font qu'une situation est unique et compréhensible.

Dans le domaine de l'Informatique Ubiquitaire, beaucoup de définitions ont été proposées pour cette notion. *Abowd et al.* [2] ont recensé ces différentes définitions, les ont analysées et les ont classées en trois catégories :

— *Définitions par énumération d'exemples* ([180], [35], [177], [57]) : dans cette catégorie, les chercheurs définissent le contexte en listant des exemples : la localisation, le temps, l'identité de l'utilisateur, etc.

— *Définitions par synonymes* ([34], [70], [204], [170], [91]) : dans ce cas, les chercheurs utilisent des synonymes pour définir le contexte : environnement, situation de l'utilisateur, paramètres de l'application, etc.

— *Définitions spécifiques* ([179], [56], [148]) : ici, chacun définit le contexte d'une manière spécifique à son domaine d'application.

10. http://www.larousse.fr/dictionnaires/francais/contexte/18593?q=contexte#18491
11. http://www.le-dictionnaire.com/definition.php?mot=contexte
12. https://fr.wikipedia.org/wiki/Contexte
13. http://www.cnrtl.fr/definition/contexte

Abowd et al. ont conclu : (i) que les définitions basées sur des exemples sont difficile à appliquer dans la pratique et qu'elles ne permettent pas d'identifier un nouveau contexte ; (ii) que bien que plus générales que les énumérations, les définitions par synonymes ne donnent pas d'indications pour analyser les éléments pouvant constituer un contexte et ne permettent pas de les identifier et (iii) que les définitions de la troisième catégorie sont trop spécifiques et ne peuvent pas être utilisées pour identifier le contexte.

Finalement, ils ont proposé une définition plus générique qui encapsule toutes les autres définitions : « *toute information pouvant être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet considéré comme pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application eux-mêmes* ».

Cette définition, de par son caractère générique, est jusqu'à ce jour la définition du contexte la plus répandue et la plus utilisée par les différentes communautés de recherche. Chacun, selon son domaine d'application, peut soit l'adopter telle qu'elle est soit la compléter et/ou la modifier suivant ses spécificités.

Dans le domaine des bases de données, bien que quelques chercheurs comme *Pitoura et al.* [159], *Bolchini et al.* [20] et *Elmongui et al.* [63] ont approuvé la définition de *Abowd et al.* et l'ont utilisée en l'état, d'autres travaux ont proposé de nouvelles définitions; comme par exemple, *Stefanidis et al.* [192] qui, en s'appuyant sur la définition de *Abowd et al.*, ont proposé leur propre définition : « *le contexte est toute information externe à la BDD et peut être utilisée pour caractériser la situation d'un utilisateur ou information interne pouvant être utilisée pour caractériser les données* » ou bien *Feng et al.* [68] qui l'ont re-défini comme « *la situation dans laquelle l'accès de l'utilisateur à la base de données s'effectue* ».

Au final, nous remarquons que le champ des définitions de la notion de contexte s'étend de définitions trop larges jusqu'à des définitions trop spécifiques. Ainsi, et même si la définition proposée par *Abowd et al.* est celle qui est la plus répandue dans le monde de la recherche, nous concluons que la notion de contexte varie selon les spécificités du domaine considéré et ne peut pas avoir une définition unique et consensuelle acceptable par toutes les communautés de recherche.

Nous détaillons dans le chapitre suivant notre propre vision correspondant à notre travail présenté dans cette thèse.

2.2 Catégorisation du contexte

Afin de faciliter la compréhension et l'utilisation du contexte, plusieurs travaux de recherche ont proposé de classifier les informations qui le composent en plusieurs catégories et/ou paramètres (dimensions) selon différentes perspectives. Nous présentons ici un état de l'art recensant les différentes catégorisations proposées dans la littérature (cf. Table 3.1). Pour cela, nous nous basons sur le travail de *Van Bunningen et al.* [199] qui classe les différentes catégorisations du contexte en deux grandes familles : catégorisation conceptuelle et catégorisation opérationnelle.

2.2.1 Catégorisation conceptuelle

Cette famille regroupe les catégorisations qui sont basées sur le sens du contexte et qui distinguent les différents types de contexte au niveau conceptuel. Les travaux qui appartiennent à cette famille ont d'abord restreint la notion de contexte aux deux paramètres *localisation* et *temps*. Ensuite, cette notion a été enrichie par la considération d'un ensemble de nouvelles dimensions :

- *Schilit et al.* [179] ont catégorisé le contexte en trois catégories sous forme de questions : (i) Où êtes vous : la localisation de l'utilisateur, (ii) Qui est avec vous : les informations concernant les personnes se trouvant autour de l'utilisateur et (iii) Quelles ressources se trouvent dans l'environnement de l'utilisateur.
- *Ryan et al.* [177] ont considéré les dimensions du contexte suivantes : l'identité, la localisation, le temps et l'environnement.
- *Abowd et al.* [2] ont considéré les quatre dimensions suivantes : l'identité, la localisation, le temps et l'activité.
- *Chen et al.* [46] ont proposé la catégorisation suivante :
  - *le contexte d'utilisateur* : le profil de l'utilisateur, sa localisation, les personnes se trouvant à proximité de lui, etc ;
  - *le contexte physique* : il se réfère à l'environnement de l'utilisateur, comme par exemple la température, le niveau de bruit, l'éclairage, etc ;
  - *le contexte de calcul* : il inclut la connectivité des réseaux, la bande passante des communications et les ressources à proximité telles que les imprimantes, les écrans et les postes de travail ;
  - *le contexte temps* : journée, semaine, mois, saison, année, etc.
- *Chong et al.* [47] ont considéré les dimensions suivantes : le contexte de calcul, le contexte physique, le contexte historique, l'identité et le temps.
- *Zhong* [81] a proposé les catégories suivantes : contexte d'utilisateur, contexte de calcul (système), contexte physique (environnement), contexte temps et contexte social.

Cette famille de catégorisation permet de fournir des méthodes assez claires et structurées pour organiser le contexte. L'avantage principal de cette famille de classifications est sa flexibilité et son extensibilité. Cependant, elle ne permet pas de fournir des informations concernant les sources de contexte et la manière d'acquisition de données.

2.2.2 Catégorisation opérationnelle

Cette famille concerne les catégorisations basées sur la façon dont le contexte a été acquis, modélisé et traité. Parmi les travaux qui appartiennent à cette famille, nous pouvons citer :

- *Henricksen* [87] : il a proposé quatre catégories de contexte.

	Catégorisation conceptuelle	**Catégorisation opérationnelle**
Schilit et al. (1994) [179]	Où, Qui, Quel(les)	-
Ryan et al. (1999) [177]	Identité, Localisation, Temps, Environnement	-
Abowd et al. (1999) [2]	Identité, Localisation, Temps, Activité	-
Chen et al. (2000) [46]	Utilisateur, Physique, Calcul, Temps	-
Wang et al. (2004) [203]	-	Bas niveau, Haut niveau
Henricksen (2003) [87]	-	Capturé, Statique, Dynamique, Dérivé
Prekop et al. [160] et Hofer et al [88] (2003)	-	Physique (externe), Logique (interne)
Chang Xu (2005) [208]	-	Physique, Logique
Guan et al. (2007) [82]	-	Bas niveau, Haut niveau
Chong et al. (2007) [47]	Calcul, Physique, Historique, Identité, Temps	-
Zhong (2009) [81]	Utilisateur, Système, Environnement, Social, Temps	-
Rizou et al. (2010) [168]	-	Bas niveau, Haut niveau

TABLE 3.1 – Catégorisation du contexte

1. **Capturé :** les données capturées directement par des capteurs, comme par exemple la température mesurée avec un thermomètre.
2. **Statique :** les informations qui ne changent pas avec le temps, comme par exemple le fabricant du capteur.
3. **Dynamique :** les informations qui changent au fil du temps, comme par exemple l'emplacement du capteur.
4. **Dérivé :** les informations calculées en utilisant d'autre contextes primaires, comme la distance entre deux capteurs GPS.

— *Prekop et al.* [160], *Chang Xu* [208] et *Hofer et al.* [88] : ils ont distingué entre deux catégories de contexte :

1. **Physique (externe) :** les informations de l'environnement extérieur qui sont captu-

rées directement, comme par exemple la localisation GPS.

2. **Logique (interne) :** les informations abstraites sur l'environnement utilisées pour enrichir la sémantique des contextes physiques, comme par exemple les noms des rues.

— *Wang et al.* [203], *Guan et al.* [82] et *Rizou et al.* [168] : ils ont considéré deux catégories de contexte :

1. **Bas niveau :** les informations simples qui peuvent être directement obtenues à partir de capteurs ou d'autres sources.
2. **Haut niveau :** les informations abstraites qui doivent être inférées à partir des contextes de bas niveau.

Nous remarquons que cette famille (la catégorisation opérationnelle) permet de fournir des informations concernant les sources des différents contextes et la manière d'accéder aux données (raisonnement ou non). Cependant, elle reste ambiguë quant à la classification des informations contextuelles. Les mêmes informations peuvent appartenir à différentes catégories, comme par exemple la localisation qui peut être capturée ou dérivée.

2.3 Modélisation du contexte

Plusieurs techniques ont été proposées dans la littérature pour la modélisation des informations contextuelles. Nous présentons dans cette sous-section ces différentes techniques et nous les comparons entre elles en faisant apparaître leurs avantages et leurs limitations.

Chen et al. [46] et *Strang et al.* [194] ont recensé les six techniques de modélisation de contexte les plus populaires : les modèles clé-valeur, les modèles à balises, les modèles graphiques, les modèles orientés objets, les modèles logiques et les modèles ontologiques.

1. **Modèles clé-valeur :** il s'agit de la structure de données la plus simple pour modéliser le contexte. Cette approche utilise des paires de type 'clé-valeur' pour représenter l'information contextuelle (par exemple 'temps-10h00'). Cette approche est simple d'utilisation mais très limitée en termes d'expressivité et d'évolution du modèle.
2. **Modèles à balises :** ces modèles se basent sur une structure de données hiérarchique constituée de tags de balisages associés à des valeurs d'attributs. La plupart de ces modèles sont dérivés du modèle Composite Capabilities/Preference Profiles [14] (CC/PP) qui a été proposé par le W3C pour répondre aux problèmes d'accès à des pages ou des services Web par des dispositifs de plus en plus évolutifs (smartphone, tablette, TVs connectées, etc). L'avantage de ces modèles à balises est l'utilisation des schémas XML réutilisables pour la définition des propriétés contextuelles. Cependant, ces modèles sont limités en terme d'expressivité, en particulier : l'expression des contraintes sur les valeurs des propriétés et l'expression des relations et liens entre ces propriétés. Parmi ces modèles, nous

14. https://www.w3.org/TR/CCPP-struct-vocab/

pouvons citer à titre d'exemples : Centaurus Capability Markup Language (CCML) [102], Comprehensive Structured Context Profiles (CSCP) [86] et Context Description Framework (CDF) [110].

3. **Modèles graphiques :** ces modèles se basent sur les langages formels graphiques offrant une visualisation immédiate des modèles. Unified Modelling Language (UML) [15] et Object Role Modelling (ORM) [16] en sont des exemples. En terme d'expressivité, les modèles graphiques sont meilleurs que les modèles clé-valeur et les modèles à balises car ils permettent de représenter les relations. Cependant, le nombre d'implémentations différentes peut rendre difficile leur utilisation surtout en ce qui concerne l'interopérabilité (entre BDD par exemple). De plus, le mécanisme d'interrogation du contexte (par exemple SQL) peut s'avérer difficile en exigeant des requêtes complexes. Toutefois, des solutions s'articulant autour du mouvement noSQL [84] peuvent résoudre ces limitations.
4. **Modèles orientés objets :** ces modèles utilisent l'approche orientée objet pour la modélisation des informations contextuelles. Ils bénéficient de ce fait de ses avantages comme l'encapsulation et la réutilisabilité. Cependant, ces modèles souffrent de deux inconvénients majeurs : le manque de capacités de raisonnement ainsi que la difficulté de validation des conceptions orientées objet en raison de l'absence de normes et de standards dédiés.
5. **Modèles logiques :** ces modèles utilisent les faits, les expressions et les règles pour modéliser le contexte. L'utilisation de règles permettant l'expression des différentes politiques, contraintes et préférences offre une riche expressivité par rapport aux autres modèles. Les modèles logiques prennent en charge le raisonnement logique et permettent d'extraire de nouvelles informations contextuelles à partir d'autres. Cependant, la réutilisation de ces modèles est réduite à cause de l'absence de standards.
6. **Modèles ontologiques :** ces modèles utilisent des ontologies pour modéliser le contexte. Cette approche présente beaucoup d'avantages. En effet, il existe de nombreux standards (RDF, RDFS, OWL) et une panoplie d'outils et de moteurs de raisonnement qui peuvent être utilisés selon les besoins de chacun. L'inconvénient majeur de cette approche basée sur les ontologies est la complexité de la recherche et l'interrogation du contexte qui peut nécessiter beaucoup de temps lorsque la quantité des données augmente considérablement.

La table 3.2 illustre, de manière synthétique, la comparaison entre ces différentes approches de modélisation.

15. http://www.uml.org/
16. https://www.ormfoundation.org/

Approche	Avantages	Inconvénients	Synthèse
Clé-valeur	• Simple • Flexible	• Non structurée • Recherche d'information difficile • Pas de représentation des relations • Pas de validation et standards • Forte dépendance aux applications	Cette approche peut être utilisée pour une quantité limitée de données comme, par exemple, les préférences des utilisateurs.
Balisage	• Plus structurée • Validation possible par schémas • Disponibilité des outils	• Dépendance vis-à-vis des applications • Représentation compliquée des informations à plusieurs niveaux • Recherche d'information difficile	Cette approche peut être utilisée comme un format de transfert de données.
Graphique	• Représentation des relations • Recherche d'information facile • Disponibilité de standards et outils de modélisation • Validation par contraintes	• Interrogation complexe • Interopérabilité difficile entre différentes implémentations	Cette approche est facile à utiliser. Les BDD peuvent contenir de grandes quantités de données et fournissent des moyens efficaces et rapides pour la recherche d'informations relativement simples.
Orientée objets	• Représentation des relations • Intégration facile en utilisant les langage de programmation • Disponibilité des outils	• Recherche d'information difficile • Manque de validation	Cette approche peut être utilisée pour représenter le contexte, en interne, au niveau du code de programmation.
Logique	• Simple • Génération de contexte de haut niveau • Raisonnement logique • Disponibilité des outils	• Pas de standards • Manque de validation • Dépendance vis-à-vis des applications	Cette approche permet de générer des contextes de haut niveau à partir de contextes de bas niveau. Elle peut être utilisée, comme un supplément, pour améliorer d'autres approches.
Ontologique	• Raisonnement sémantique • Expressivité riche • Forte validation • Partage et indépendance par rapport aux applications • Outils et standards	• Complexité de la recherche et de l'interrogation d'information	Cette approche peut être utilisée pour modéliser la connaissance d'un domaine ainsi que les informations contextuelles en utilisant les avantages offerts par les ontologies.

TABLE 3.2 – Comparaison entre les techniques de modélisation du contexte

3 Couplage entre le contexte et l'ontologie

En raison des nombreux avantages de l'approche basée sur les ontologies, plusieurs chercheurs estiment dans leurs travaux que c'est le moyen le plus approprié pour représenter et gérer le contexte [194, 149]. Nous présentons dans cette section une brève discussion des différents travaux qui ont tenté de coupler les deux notions *Ontologie* et *contexte*. Notre but est de mettre en évidence l'applicabilité de cette approche pour le domaine considéré dans nos travaux, à savoir l'ingénierie des données.

Nous classons ces travaux en trois familles selon l'approche suivie pour représenter le contexte : ontologies représentant la dépendance au contexte, ontologies contextualisées et ontologies de multi-representation.

3.1 Ontologies représentant la dépendance au contexte

Afin de représenter et de minimiser la dépendance au contexte dans une ontologie, *Pierra* [151] a proposé le modèle d'ontologie PLIB qui définit et prend en charge deux types de contextes :

— **Le contexte de modélisation** : le modèle PLIB permet de définir précisément les classes et les propriétés en imposant un couplage fort entre elles : (i) une propriété ne peut être définie sans l'indication de son domaine d'application par l'intermédiaire d'une classe et (ii) une classe ne peut être définie sans l'indication des propriétés essentielles pour définir ses instances. Il distingue entre trois catégories de classes :

  — les classes de définition (*item class*) qui contiennent les propriétés essentielles d'une classe ;

  — les classes de représentation (*functional model class*) qui contiennent les propriétés qui n'ont de sens que par rapport à un point de vue ;

  — les classes de vue (*functional view class*) qui définissent la perspective selon laquelle les propriétés des classes de représentation sont définies.

  Par exemple, le concept '*disque dur*' peut être représenté, en utilisant le modèle PLIB, par trois classes :

  1. une classe de définition qui sera le domaine de la propriété essentielle *capacité* ;
  2. une classe de représentation définie comme une vue de la première classe et qui représente le domaine de la propriété *prix* ;
  3. une classe de vue liée à celle de représentation qui sera, par exemple, le domaine des propriétés *date de vente* et *type d'acheteur*.

— **Le contexte d'évaluation** : il concerne l'évaluation des propriétés qui peuvent dépendre soit d'unités ou de monnaies à expliciter, soit d'autres propriétés. Dans le premier cas,

le modèle PLIB associe à toute propriété représentant une quantité mesurable une unité définie par un schéma EXPRESS standard (ISO 10303-41 :2000). Dans le deuxième cas, PLIB permet de distinguer entre trois types de propriétés :

— Propriétés caractéristiques non-dépendantes du contexte : ce sont les propriétés essentielles des classes qui possèdent une valeur pour chaque instance. Elle sont indépendantes des autres propriétés. Par exemple, la date de naissance d'une personne.

— Propriétés dépendantes du contexte : ce sont les propriétés des classes dont les valeurs sont représentées en fonction d'autres paramètres caractérisant des contextes particuliers d'utilisation ou d'évaluation. Par exemple, le poids d'une personne dépend de la date à laquelle la mesure a été faite.

— Paramètres de contexte : ce sont les propriétés qui représentent les contextes possibles d'utilisation ou d'évaluation. Par exemple, la date à laquelle le poids de la personne a été mesuré.

3.2 Ontologies contextualisées

En considérant qu'une ontologie est construite pour être partagée et qu'un contexte est construit pour être conservé localement, *Bouquet et al.* [26, 25] ont proposé la notion d'*ontologie contextualisée* qui combine ces deux notions dans la même structure afin de mettre en commun leurs avantages. Une *ontologie contextualisée* désigne une ontologie dont le contenu est défini dans un contexte particulier et mis en relation avec le contenu d'autres ontologies en utilisant des mappings.

Afin de permettre la représentation des *ontologies contextualisées*, *Bouquet et al.* ont proposé une extension de OWL nommée Context OWL (C-OWL). Ce formalisme, basé sur les principes de la logique de description distribuée (LDD) [21], s'appuie sur deux principes essentiels :

— une sémantique à ontologies locales : chaque contexte correspond à une ontologie locale ayant ses propres langage et interprétation ;

— des relations sémantiques entre ontologies locales : ce sont les mappings qui correspondent à des relations sémantiques entre les ontologies locales des différents contextes. Ils établissent des correspondances entre les éléments de ces ontologies.

En résumé, un contexte en C-OWL est une ontologie locale représentée en OWL qui possède son propre langage et sa propre interprétation. Les éléments de chaque contexte (les classes, les propriétés et les individus,) sont reliés entre eux par des passerelles associées à une sémantique qui permet de réaliser des raisonnements globaux [183] en réutilisant les connaissances d'un contexte. La figure 3.1 illustre un exemple d'appariement entre deux contextes avec C-OWL (production de voiture ($O_{production}$) et vente de voiture (O_{vente})) qui utilise les cinq formes de passerelles possibles :

— $i : x \xrightarrow{\equiv} j : y$ (***equivalent***) indique que l'élément $(i : x)$ du contexte O_i est considéré selon O_j comme équivalent à l'élément $(j : y)$.
Exemple : production:Automobile $\xrightarrow{\equiv}$ vente:Voiture

— $i : x \xrightarrow{\sqsubseteq} j : y$ (***dedans***) indique que l'élément $(i : x)$ du contexte O_i est considéré selon O_j comme plus spécifique que l'élément $(j : y)$.
Exemple : vente:BonMarché $\xrightarrow{\sqsubseteq}$ production:faibleCoût.

— $i : x \xrightarrow{\sqsupseteq} j : y$ (***dessus***) indique que l'élément $(i : x)$ du contexte O_i est considéré selon O_j comme plus général que l'élément $(j : y)$.
Exemple : production:Utilisateur $\xrightarrow{\sqsupseteq}$ vente:Client.

— $i : x \xrightarrow{*} j : y$ (***compatible***) indique que l'élément $(i : x)$ du contexte O_i est considéré selon O_j comme compatible avec l'élément $(j : y)$.
Exemple : production:Conducteur $\xrightarrow{*}$ vente:Client.

— $i : x \xrightarrow{\perp} j : y$ (***incompatible***) indique que l'élément $(i : x)$ du contexte O_i est considéré selon O_j comme incompatible avec l'élément $(j : y)$.
Exemple : vente:BonMarché $\xrightarrow{\perp}$ production:MatériauNoble.

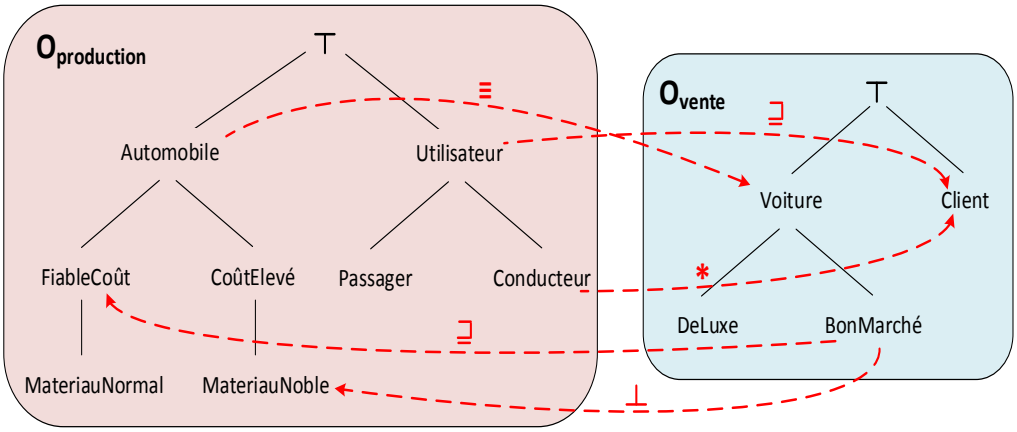


FIGURE 3.1 – Exemple d'appariement C-OWL

3.3 Ontologies de multi-représentation

Dans [15], *Benslimane et al.* considèrent que le contexte est étroitement lié aux notions de niveau d'abstraction, de granularité, d'intérêt des communautés d'utilisateurs et de perception des développeurs d'ontologies. Ils estiment qu'un même domaine peut être modélisé par plusieurs ontologies *mono-contextuelles* où chacune d'elles est décrite dans un contexte particulier. Par exemple, le concept *Employee* peut être défini dans un contexte s_1 comme quelqu'un qui

possède un numéro d'employé et dans un contexte s_2 comme quelqu'un qui travaille dans une société.

Afin de pouvoir décrire simultanément une même ontologie selon plusieurs contextes, *Benslimane et al.* [16] ont défini la notion d'*ontologies multi-contextuelles* ou *ontologies de multi-représentation* (OMR). Celles-ci caractérisent les concepts par un ensemble variable de propriétés ou d'attributs selon plusieurs contextes et granularités. Ils proposent une formalisation de ces ontologies basée sur les logiques de description.

Ils ont introduit un nouveau constructeur pour la notion de contexte dont la syntaxe et la sémantique sont les suivantes :

— **Syntaxe :** si C est un concept et $s_1, ..., s_m$ est un ensemble de *noms de contexte* :
C $\longrightarrow$ (C)[S] (restrictions contextuelles)
S $\longrightarrow$ liste de *noms de contexte*.

— **Sémantique :** la partie sémantique est définie par des interprétations contextuelles. Une interprétation contextuelle $\mathcal{I} = (\mathcal{I}_0, ..., \mathcal{I}_j, ..., \mathcal{I}_t)$ est un t-uplets indexé par les contextes 1,...,t. Chaque élément $\mathcal{I}_j$ est une interprétation non contextuelle $(\Delta^{\mathcal{I}}, .^{\mathcal{I}^j})$ qui consiste en un domaine d'interprétation $\Delta^{\mathcal{I}}$ et une fonction d'interprétation $.^{\mathcal{I}^j}$. Cette fonction d'interprétation relie chaque concept atomique $A \in C$ au sous-ensemble $A^{\mathcal{I}^j} \subseteq \Delta^{\mathcal{I}}$ et chaque rôle $R \in \mathcal{R}$ au sous-ensemble $R^{\mathcal{I}^j} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$:

$$\begin{gathered}
\bot^{\mathcal{I}^j} = \emptyset \\
\top^{\mathcal{I}^j} = \Delta^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}^j} = C^{\mathcal{I}^j} \sqcap D^{\mathcal{I}^j} \\
(C \sqcup D)^{\mathcal{I}^j} = C^{\mathcal{I}^j} \sqcup D^{\mathcal{I}^j} \\
(\exists R.C)^{\mathcal{I}^j} = \{\ x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in R^{\mathcal{I}^j} \wedge y \in C^{\mathcal{I}^j}\} \\
(\forall R.C)^{\mathcal{I}^j} = \{\ x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in R^{\mathcal{I}^j} \rightarrow y \in C^{\mathcal{I}^j}\} \\
(\leq nR)^{\mathcal{I}^j} = \{x \in \Delta^{\mathcal{I}} \mid \ \| \{y \mid (x,y) \in R^{\mathcal{I}^j}\} \| \leq n\} \\
(\geq nR)^{\mathcal{I}^j} = \{x \in \Delta^{\mathcal{I}} \mid \ \| \{y \mid (x,y) \in R^{\mathcal{I}^j}\} \| \geq n\} \\
((C)[S])^{\mathcal{I}^j} = \left\{ \begin{array}{cc} C^{\mathcal{I}^j} & if\ j \in S \\ \emptyset & otherwise \end{array} \right\}
\end{gathered}$$

Le concept *employee* de l'exemple précédent est défini comme suit :
Employee = ($\exists$*EmployeeNumber.Number*)[s_1] $\sqcup$ ($\exists$ *WorksFor.Company*)[s_2].

3.4 Synthèse

La revue des différentes approches proposées dans la littérature pour coupler la notion de contexte à celle d'ontologie nous a permis d'identifier certaines lacunes. Le modèle PLIB permet de modéliser le contexte au sein d'une ontologie en imposant un couplage fort entre les propriétés et les classes (contexte de modélisation) et surtout en représentant la dépendance des

concepts au contexte (contexte d'évaluation). Cependant, cette modélisation est minimale car elle ne permet pas d'expliciter la relation existant entre les différents concepts pour représenter l'intégralité de l'information contextuelle. Les deux autres approches (ontologies contextualisées et ontologies de multi-représentations) ne permettent pas de représenter le contexte d'évaluation. Elles se contentent de représenter les différentes interprétations dans une ontologies. De plus, l'inconvénient de ces deux approches est que le développement des ontologies est une tâche très gourmande en temps et le fait d'y incorporer le contexte conduit à son augmentation.

4 Contexte et systèmes de stockage

Dans cette sections, nous analysons les approches existante qui ont proposé de prendre en charge le contexte dans les systèmes de base/entrepôt de données.

4.1 Contexte et Bases de Données classiques

Plusieurs approches ont été proposées dans la littérature pour la modélisation du contexte et sa prise en charge au sein des systèmes des bases de données. Dans cette section, nous commençons d'abord par donner un aperçu de chacune d'elles. Ensuite, nous présentons les exigences qu'une approche de modélisation de contexte doit, selon notre vision, vérifier. Enfin, nous présentons une analyse comparative de ces approches.

4.1.1 Approches existantes

Nous résumons ici les différents travaux de recherche proposés dans la littérature.

— *Roussos et al.* [176, 209] ont proposé un modèle appelé *Context Relational model (CR model)* ainsi qu'un ensemble d'opérations de base qui étendent respectivement le modèle relationnel et l'algèbre relationnelle pour prendre en charge le contexte. Dans leurs travaux, ils considèrent le contexte comme une entité de première classe au niveau des modèles des bases de données et des langages de requête.

— *Pierra* [152] a proposé une approche pour représenter et minimiser la dépendance au contexte dans les ontologies de domaine. Il s'agit du modèle d'ontologies PLIB qui a servi comme base pour la proposition de la base de données sémantique *OntoDB*[55]. Nous rappelons que deux types de contextes sont pris en compte par le modèle PLIB : (i) *le contexte de modélisation* qui consiste à définir précisément les classes et les propriétés en imposant un couplage fort entre elles, et (ii) *le contexte d'évaluation* qui concerne l'évaluation des propriétés qui peuvent dépendre soit d'unités de mesure ou de monnaies à expliciter, soit d'autres paramètres de contexte.

— *Martinenghi et al.* [126, 127] ont proposé une extension du modèle relationnel dans laquelle ils considèrent le contexte comme une entité de première classe. Ils ont également proposé une extension de l'algèbre relationnelle appelée *Algèbre Relationnelle Contextuelle (ARC)* qui permet de réaliser des interrogations contextuelles.

— *Levandoski et al.* [118, 117] ont proposé *CareDB*, un système de BDD relationnelle qui permet de prendre en charge le contexte et les préférences dans le traitement des requêtes. Trois types de contexte sont pris en considération : le contexte de l'utilisateur, le contexte de la base de données et le contexte de l'environnement.

— *Pitoura et al.* [158] ont proposé d'annoter les préférences avec des informations contextuelles. Le contexte est modélisé sous forme d'un ensemble de paramètres hiérarchisés. Beaucoup de travaux ont été proposés dans le domaine des préférences contextuelles. Une revue de ces travaux est disponible dans [192].

— *Freitas et al.* [71] ont proposé *CODI4In*, une approche pour la personnalisation des requêtes dans les applications orientées données en tenant compte du contexte de l'utilisateur. Le modèle de contexte proposé dans leur travail est une ontologie nommée *CODI-User*.

— *Bolchini et al.* [20] et *Quintarelli et al.* [163] ont proposé dans un premier temps une méthodologie pour la définition des vues contextuelles dans les BDD relationnelles à travers l'introduction d'un modèle nommé *Context Dimension Tree (CDT)*. Celui-ci représente les contextes d'utilisateur tels que son emplacement, son rôle et ses intérêts sous forme d'un arbre. Certains opérateurs relationnels ont aussi été redéfinis afin de permettre la formulation des vues contextuelles. Dans un second temps, ils ont introduit un ensemble d'opérateurs permettant de modifier le schéma de contexte en cas d'évolution.

4.1.2 Étude comparative et synthèse

Nous considérons qu'il est important que l'approche de modélisation du contexte réponde aux exigences suivantes :

— E1 : *Identification des dimensions du contexte* : le modèle de contexte doit définir clairement les dimensions pertinentes de l'information contextuelle ;

— E2 : *Généricité du modèle* : le modèle de contexte doit être suffisamment générique pour pouvoir être utilisé dans différentes applications ;

— E3 : *Aspect formel du modèle* : la définition du modèle doit être la plus précise possible ;

— E4 : *Modélisation au niveau conceptuel* : les informations contextuelles doivent être définies au niveau conceptuel afin de garantir l'indépendance de toute implémentation spécifique ;

— E5 et E6 : *Considération des paramètres internes* et *Considération des paramètres externes* : le modèle doit prendre en considération les deux types de paramètres du contexte : (i) les paramètres internes stockés dans la BDD et (ii) les paramètres externes à la BDD ;

— E7 : *Explicitation sémantique* : l'approche doit fournir une solution pour rendre explicite la sémantique des données manipulées ;

— E8 : *Mécanisme de persistance du modèle* : l'approche doit offrir un mécanisme de persistance du modèle ;

— E9 : *Technique d'exploitation dédiée* : l'approche doit être dotée d'une technique dédiée pour exploiter le modèle de contexte.

La table 3.3 présente la comparaison entre les différentes approches citées dans la sous-section précédente par rapport à ces exigences. Dans cette table, nous utilisons le symbole ✓ pour indiquer qu'une exigence est prise en charge par une approche. Si une exigence n'est pas prise en compte par une approche donnée, le symbole ✗ est utilisé.

Référence	Description	E1	E2	E3	E4	E5	E6	E7	E8	E9
Roussos et al. [176, 209]	Extension du modèle relationnel et de l'algèbre relationnelle pour la prise en charge du contexte	✗	✓	✓	✓	✓	✓	✗	✓	✓
Pierra [152]	Représentation du contexte dans les ontologies de domaine (le modèle PLIB)	✗	✓	✓	✓	✓	✗	✓	✓	✗
Martinenghi et al. [126, 127]	Extension du modèle relationnel et de l'algèbre relationnelle pour permettre de réaliser des interrogations sensibles au contexte	✗	✓	✓	✓	✓	✓	✗	✓	✓
Levandoski et al. [118, 117]	Système de BDD relationnel permettant la prise en charge du contexte et des préférences dans les requêtes	✓	✗	✗	✗	✗	✓	✗	✓	✓
Pitoura et al. [158]	Annotation des préférences avec les informations contextuelles	✗	✗	✓	✓	✓	✓	✗	✗	✗
Freitas et al. [71]	Personnalisation des requêtes avec le contexte de l'utilisateur	✓	✗	✓	✓	✓	✓	✓	✓	✓
Bolchini et al. [20] et *Quintarelli et al.* [163]	Méthodologie pour la définition des vues contextuelles dans les BDD relationnelles	✓	✗	✓	✓	✓	✗	✓	✓	✓

TABLE 3.3 – Analyse des travaux proposant la prise en charge du contexte dans les systèmes de Bases de Données

L'analyse de cette table montre qu'aucune de ces approches ne satisfait l'ensemble de nos exigences. Par conséquent, nous jugeons qu'il est nécessaire de proposer une nouvelle approche

de modélisation du contexte tout en s'appuyant sur les avantages et les fonctionnalités des modèles existants, et ce afin d'avoir une approche la plus générique possible.

4.2 Contexte et Entrepôts de Données

Dans cette section, nous présentons un état de l'art des approches proposées dans la littérature pour l'intégration du contexte dans les systèmes d'ED.

4.2.1 Approches existantes

Les principaux travaux proposés dans la littérature et relatifs aux ED contextuels sont les suivants :

— *Calvanese et al.* [38] ont été les premiers à proposer une approche d'intégration et de réconciliation de données basée sur la représentation du contexte d'évaluation. Dans cette approche, le modèle conceptuel du domaine et le modèle conceptuel des sources sont formalisés à l'aide d'une logique de description spécifique appelée *DLR*. Ils ont proposé un algorithme de ré-écriture pour fournir les requêtes permettant de charger les différentes relations d'ED en prenant en charge le contexte.

— *Perez et al.* [150] ont proposé une approche pour la construction d'un ED contextuel en combinant un ED classique avec un entrepôt de documents. Le but de cette approche est d'offrir un système d'aide à la décision qui combine les données structurées et les documents non structurés des différentes sources et qui permet aux utilisateurs d'analyser, à l'aide d'un nouveau type de cubes OLAP appelé *R-OLAP*, les données intégrées dans différents contextes. Ce cube est caractérisé par deux dimensions : *Pertinence* et *Contexte*. La première dimension permet de mesurer la pertinence de chaque fait dans le contexte d'analyse. La deuxième dimension relie chaque fait avec les documents qui expliquent le contexte.

— *Jerbi et al.* [101] ont proposé une approche pour la personnalisation des analyses OLAP en se basant sur un modèle multidimensionnel qui représente les préférences de l'utilisateur et ses contextes d'analyse.

— *Garrigos et al.* [76] ont proposé une approche de modélisation conceptuelle pour personnaliser le système OLAP. Cette personnalisation consiste à utiliser : (i) un modèle d'utilisateur qui représente toutes les informations relatives à l'utilisateur nécessaires à la personnalisation (les différents contextes), et (ii) un ensemble de règles de personnalisation qui spécifient les actions de personnalisation nécessaires. Ces modèles permettent d'obtenir un ensemble de schémas OLAP contextualisés spécifiques à l'utilisateur.

— *Pitarch et al.* [156, 157] ont proposé une approche permettant de prendre en compte les hiérarchies contextuelles au sein d'un ED du point de vue des mesures. Ils ont proposé un algorithme de réécriture de requêtes qui permet d'exploiter de telles hiérarchies.

— *Oukid et al.* [144] ont proposé un modèle de cube textuel contextuel nommé *CXT-Cube* dans lequel chaque dimension est liée à un facteur contextuel (contexte du document, contexte de l'utilisateur). Ils ont défini un nouvel opérateur d'agrégation appelé *OLAP Rank (ORank)* qui permet de naviguer à travers tous les facteurs contextuels définis dans le modèle.

— *Khouri et al.* [109] ont proposé une approche de conception d'ED en utilisant une ontologie étendue par les informations contextuelles. Dans cette approche le contexte a été embarqué dans l'ontologie.

4.2.2 Étude comparative et synthèse

Nous comparons ici les approches présentées ci-dessus en se basant sur les critères suivants :

— *L'approche* : ce critère sert à distinguer entre les approches de construction d'ED classiques et sémantiques (basées sur les ontologie) ;

— *Le modèle* : ce critère sert à préciser le type du modèle de contexte proposé par l'approche ou sur lequel repose cette dernière. Nous rappelons qu'il existe cinq types principaux : *Clé-valeur*, *Balisage*, *Graphique*, *Logique* et *Ontologique* ;

— *La phase* : ce critère sert à déterminer la ou les phases du cycle de construction d'ED couvertes par l'approche : Modélisation conceptuelle (C), Modélisation logique (L), ETL et Exploitation (Exp) ;

— *Information contextuelle* : ce critère sert à distinguer les catégories de l'information contextuelle prises en charge par l'approche. Nous distinguons ici entre les contextes de définition (Def) et les contextes d'évaluation (Eval). Dans ce dernier cas, nous différencions les contextes internes (Int) qui dépendent des concepts locaux et les contextes externes (Ext) qui dépendent de paramètres externes comme le temps et la localisation.

La lecture de la table 3.4 illustrant cette comparaison nous a permis de faire deux constats qui concernent deux points essentiels : les approches de modélisation de contexte sur lesquelles reposent ces travaux et les méthodes de construction d'ED proposées.

— *les approches de modélisation de contexte :* les modèles de contexte sur lesquels reposent la majorité de ces travaux sont basés sur des techniques de modélisation simples qui manquent d'expressivité et de formalisme et qui ne permettent pas d'expliciter la sémantique de données. Seul le travail de *Khouri et al.* [109], qui se base sur l'utilisation des ontologies contextuelles, fait exception. Mais son inconvénient majeur est le fait qu'il ne prend pas en charge les contextes de définition et d'évaluation (seules les dépendances fonctionnelles entre les propriétés sont considérées) ;

— *les méthode de construction d'ED contextuel :* aucun des travaux présentés précédemment ne propose de méthode de conception complète qui explicite les différentes phases de construction d'un ED Contextuel.

Référence	Approche	Modèle	Phase(s)	Information Contextuelle		
				Def.	Eval.	
					Int.	Ext.
Calvanese et al. [39]	Sémantique	Clé-valeur	C, L	✗	✗	✓
Perez et al. [150]	Classique	Documents	C	✓	✗	✗
Jerbi et al. [101]	Classique	Graphique	Exp	✗	✗	✓
Garrigos et al. [76]	Classique	Graphique	C	✗	✗	✓
Pitarch et al. [157]	Classique	Graphique	Exp	✗	✓	✗
Oukid et al. [144]	Classique	Graphique	C	✓	✗	✓
Khouri et al [109]	Sémantique	Ontologique	L, ETL	✗	✓	✗

TABLE 3.4 – Analyse des travaux intégrant le contexte dans les systèmes des Entrepôts de Données

5 Conclusion

Dans ce troisième chapitre, nous avons d'abord présenté, de manière détaillée, la notion de contexte. Nous avons vu que cette notion a été étudiée par plusieurs communautés de recherche en informatique comme l'Intelligence Artificielle, l'Informatique Diffuse, la Recherche d'Informations, l'Apprentissage Automatique, les Bases de Données et les Entrepôts de Données. Chacune de ces communautés a adopté, en fonction de ses spécificités, une vision particulière du contexte. Cela nous a amené à conclure que le Contexte ne peut pas avoir une unique définition consensuelle qui soit approuvable par les différentes communautés de recherche. Nous avons également présenté les différentes catégorisations ainsi que les différentes techniques de modélisation du contexte qui ont été proposées dans la littérature.

Nous avons ensuite présenté les principaux travaux relatifs à l'intégration du contexte dans les systèmes de BDD. Nous avons constaté que la plupart de ces travaux consistent à étendre le modèle relationnel avec le contexte et qu'aucun d'entre eux ne propose d'approche de modélisation qui réponde à l'ensemble des exigences de modélisation que nous avons fixées.

Enfin, nous avons réalisé une étude comparative des travaux de recherche concernant la prise en charge du contexte dans les systèmes d'ED. Cette étude nous a permis de constater que les approches existantes présentent deux restrictions majeures. Premièrement, les modèles

de contexte proposés sont soit basés sur des techniques de modélisation qui ne permettent pas l'explicitation de la sémantique des données, soit ils ne prennent pas en charge les différentes catégories de l'information contextuelle. Deuxièmement, aucune de ces approches ne propose de méthode complète expliquant les différentes phases de construction d'un ED Contextuel.

Nous essayons, à travers le travail présenté dans cette thèse, de traiter et de résoudre toutes ces limitations sous la forme de trois contributions principales : (i) la proposition d'un modèle générique de contexte basé sur une approche d'externalisation par rapport au modèle d'ontologie, (ii) la proposition d'une approche pour la prise en charge complète du contexte dans les systèmes de BDS et enfin (iii) la proposition d'une méthode de conception des ED Contextuels. Ces trois contributions seront respectivement présentées dans les chapitres 4, 5 et 6.

Deuxième partie

Contributions

Chapitre 4

Modélisation et explicitation du contexte : une approche d'externalisation

Sommaire

1 Introduction 71
2 Fondements théoriques 72
2.1 Définition des ressources de modélisation 72
2.1.1 Standards de l'OMG 72
2.1.2 Ressources de modélisation pour notre modèle de contexte 75
2.2 Langage de modélisation EXPRESS 75
2.2.1 Le langage 75
2.2.2 Représentation des instances : fichier physique 77
2.2.3 Représentation graphique : EXPRESS-G 77
2.2.4 Choix d'EXPRESS comme langage de modélisation . 78
3 Notion de contexte 79
3.1 Notre vision du contexte 79
3.2 Notre catégorisation du contexte 80
3.2.1 Cas d'étude 80
3.2.2 Catégories du contexte 81
4 Notre approche de modélisation de contexte 84
4.1 Notre modèle de Contexte 84
4.1.1 L'entité 'Definition Context' 86
4.1.2 L'entité 'Evaluation Context' 87
4.2 Lien du modèle de contexte avec l'ontologie 94
5 Conclusion 95

1 Introduction

Dans le domaine des SI, les données manipulées par les entreprises ne sont généralement pas reliées aux contextes correspondants, comme par exemple les connaissances expertes, les profils des utilisateurs, le temps, les unités de mesures etc. Souvent, ces informations contextuelles ne sont pas rendues explicites à l'intérieur des modèles de données : elles sont, en général, soit codées en dur dans les applications et/ou intégrées dans des documents et procédures techniques, soit elles demeurent enfouies dans les rapports, ou bien encore, elles restent dans les cerveaux des experts.

Lors de la construction des modèles, les concepteurs identifient l'ensemble des données et concepts pertinents pour le domaine visé afin de définir ce qui est appelé *l'univers de discours* (UD). Cette identification est le résultat d'un processus complexe d'interprétation qui dépend de l'expertise des concepteurs et des informations contextuelles concernant les données considérées. Par conséquent, la nécessite de coupler cet univers de discours avec le contexte, tout en rendant ce dernier explicite dans les modèles conceptuels, est devenue une exigence essentielle dans le domaine de l'ingénierie des données.

L'importance de l'explicitation et de la représentation du contexte dans les modèles conceptuels a été soulignée par plusieurs chercheurs dans le domaine des Bases de Données. Chen et al. [45] ont suggéré d'étendre l'approche entité-association afin d'intégrer les deux aspects que sont la Qualité de données et le contexte. Pierra [152] a mis en évidence l'importance de l'explicitation du contexte dans les ontologies de domaine ainsi que son impact sur le processus d'intégration des Bases de Données hétérogènes.

Pour remédier à cette problématique liée au manque d'explicitation et de représentation du contexte, des travaux de recherches ont tenté de modéliser le contexte au sein des ontologies. L'inconvénient de cette solution est que le développement des ontologies est une tâche qui demande beaucoup de temps, et le fait d'incorporer le contexte dans ce développement conduit à l'augmentation de ce temps. L'approche que nous proposons dans ce travail repose sur la définition d'un modèle générique et partageable du contexte qui peut être lié à n'importe quelle ontologie de domaine.

Dans ce chapitre, en se basant sur l'état de l'art présenté dans le chapitre 3 et en capitalisant sur les travaux existants, nous décortiquons d'abord la notion de contexte en précisant notre vision et en identifiant ses différentes dimensions. Ensuite, nous définissons et discutons quelques exigences pour la modélisation du contexte dans le domaine de l'ingénierie des données. Enfin, nous proposons un modèle générique et formel du Contexte qui tient compte de l'ensemble de ces exigences et nous définissons un lien entre ce modèle et les ontologies du domaine existantes.

Ce chapitre est organisé comme suit : la section 2 présente les fondements théoriques nécessaires à la compréhension de notre proposition. Nous exposons dans cette section les deux points suivants : (i) la définition des ressources de modélisation utilisées et (ii) le langage de

modélisation EXPRESS que nous utilisons pour définir notre modèle de contexte tout en argumentant ce choix. La section 3 présente notre vision du contexte ainsi que sa classification en catégories. La section 4 présente le modèle du contexte proposé et explique son lien avec le modèle d'ontologie tout en détaillant les différents composants de ce modèle de contexte. La section 5 conclut ce chapitre.

2 Fondements théoriques

Nous traitons dans cette section les principaux fondements et notions théoriques nécessaires à la réalisation et à la compréhension de notre contribution qui consiste à proposer un modèle générique et formel du contexte. Nous présentons donc les deux principaux points suivants : la définition des ressources de modélisation et le langage de modélisation EXPRESS.

2.1 Définition des ressources de modélisation

Le terme *ressource* a d'abord été introduit pour faire référence à des pages et sites web adressés par des URLs (Uniform Resource Locator) mais son utilisation a été ensuite élargie afin de désigner n'importe quel objet identifié dans le web : un site web, une partie d'une page web ou encore un concept ontologique (adressés par des URI :Uniform Resource Identifier).

Comme nous allons le voir dans ce chapitre, notre approche consiste à définir un modèle de contexte et à l'associer à n'importe quel modèle d'ontologie. C'est pourquoi nous présentons ici les ressources de modélisation que nous allons utiliser. Nous commençons d'abord par introduire brièvement les standards "Model Driven Architecture"(MDA), "Meta-Object Facility"(MOF) et le standard "Ontology Definition Meta-Model"(ODM) qui définit des ressources génériques de modélisation et ensuite nous présentons les ressources nécessaires à la définition de notre modèle de contexte.

2.1.1 Standards de l'OMG

L'Object Management Group (OMG) [17] est un consortium à but non lucratif créé en 1989 dont l'objectif est d'établir des spécifications et des standards pour résoudre les problèmes liés à l'interopérabilité des systèmes d'information.

Nous présentons dans ce qui suit trois des standards OMG utilisés pour la modélisation et la méta-modélisation, à savoir le MDA, le MOF et l'ODM qui sont pertinents par rapport à notre travail.

17. http://www.omg.org/

Approche d'architecture dirigée par les modèles (MDA)

Le MDA est un standard qui vise à promulguer de bonnes pratiques de modélisation et d'exploitation des avantages des modèles. Son but principal consiste à s'appuyer sur le standard UML pour décrire une nouvelle façon de concevoir des applications en séparant l'architecture technique de la logique métier de l'entreprise. Comme le montre la figure 4.1, l'OMG définit MDA en s'appuyant sur différents standards dans une architecture à quatre niveaux :

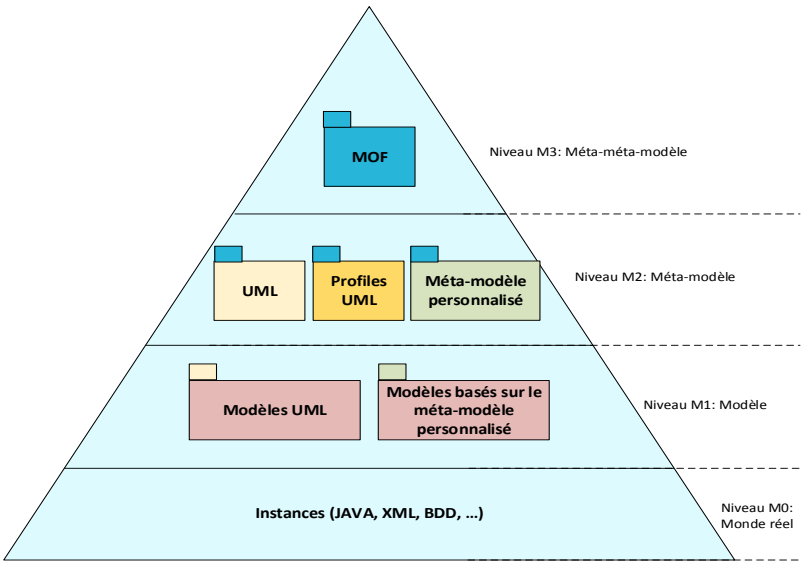


FIGURE 4.1 – Architecture à quatre niveaux de l'OMG (inspiré de [198])

— le niveau M3 (ou méta-méta-modèle) : c'est le niveau le plus élevé. Il est composé d'une seule entité qui est le MOF. Il définit la structure de tous les méta-modèles qui se trouvent au niveau M2.

— le niveau M2 (ou méta-modèle) : il définit le langage de modélisation et la grammaire de représentation des modèles M1. Typiquement, le méta-modèle UML qui est décrit dans le standard UML appartient à ce niveau.

— le niveau M1 (ou modèle) : il définit les modèles d'information qui doivent être conformes aux méta-modèles. Typiquement, un modèle UML appartient à ce niveau.

— le niveau M0 (ou instance) : c'est le niveau le plus bas de cette architecture. Il donne une représentation (modélisation) d'un système réel.

Meta-Object Facility (MOF)

Le MOF est un standard de l'OMG s'intéressant à la représentation des méta-modèles et à leur manipulation. Afin d'éviter un grand nombre de niveaux d'abstraction, le MOF a la capacité

de se définir lui même (il est réflexif), c'est-à-dire que le niveau supérieur est décrit à partir de lui-même. Cela limite l'architecture pyramidale à quatre niveaux de l'OMG (cf. Figure 4.1).

Ontology Definition Meta-Model (ODM)

L'ODM est un standard dont l'objectif est d'offrir un ensemble de configurations et de méta-modèles rapprochant le monde des ontologies de celui des méta-modèles. Basé essentiellement sur MOF et UML, il fournit une structure cohérente pour le développement des ontologies [198].
Les principaux concepts d'ODM sont :

— **RESOURCE** (ressource) : elle représente tout objet pouvant être décrit par RDFS. Comparée aux concepts de l'ontologie, elle peut être considérée comme un concept racine. En RDFS, RESOURCE est définie comme une instance de la classe MOF et est considérée comme la classe racine des autres concepts de base d'ODM.

— **ONTOLOGY** (ontologie) : c'est un concept qui regroupe d'autres concepts représentant des connaissances similaires (classes, propriétés, etc.).

— **CLASSIFIER** (classifieur) : c'est un concept permettant de regrouper des ressources possédant des caractéristiques communes. Les classifieurs ODM, contrairement aux classes traditionnelles, ne contiennent pas de méthodes. Ils peuvent être des énumérations, être définis par une restriction sur une propriété, être l'intersection, l'union, ou le complément de deux classes.

— **PROPERTY** (propriété) : permet de définir les relations et les attributs des classes. Dans ODM, PROPERTY est une instance de la classe MOF.

La figure 4.2 illustre ces concepts.

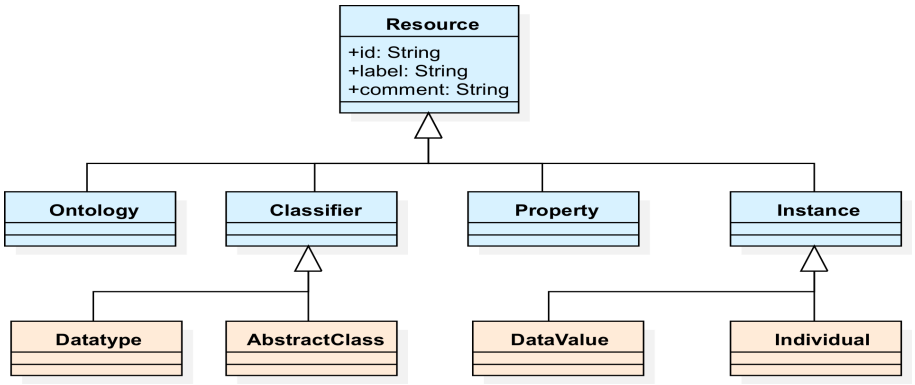


FIGURE 4.2 – Hiérarchie des concepts de base d'ODM [198]

2.1.2 Ressources de modélisation pour notre modèle de contexte

Nous spécifions ici les ressources de modélisation que nous utilisons pour la définition de notre modèle de contexte :

— **ONTOLOGY** : cette ressource regroupe les ressources représentant les différentes ontologies qui peuvent être utilisées par notre modèle de contexte. Par exemple, « UnitOntology » représente les ontologies qui définissent formellement les unités de mesures et les devises (par exemple *Ontology of Units of Measure and Related Concepts* [18]);

— **RESOURCE** : cette entité représente les ressources ontologiques définissant les ontologies : **CONCEPT** (classes), **PROPERTY** (propriétés) et **INDIVIDUAL** (individus, instances). Par exemple, PROPERTY('price') désigne la propriété ontologique « price » et CONCEPT('Patient') désigne la classe ontologique « Patient ».

2.2 Langage de modélisation EXPRESS

EXPRESS (ISO 10303-11:1994) [19] est un langage de modélisation formelle de données définit dans le cadre du standard STEP (STandard for Exchange Product model data). L'objectif principal de ce langage est de représenter des modèles de données dans le secteur de l'ingénierie. Il vise la standardisation de la représentation et de la description des données des produits pour faciliter leurs échanges.

2.2.1 Le langage

Le langage EXPRESS est un langage ensembliste et orienté objet : il supporte l'héritage et le polymorphisme et permet de manipuler les collections. EXPRESS permet de décrire, dans un formalisme clair, des concepts structuraux, descriptifs et procéduraux :

Connaissance structurelle

Le langage EXPRESS permet de modéliser la connaissance structurelle sous forme de hiérarchies d'entités (Entity) associées à un mécanisme d'héritage. EXPRESS autorise différents types d'héritage : héritage simple, multiple ou répété.

Exemple générique :

```
ENTITY Nom_entité;
SUBTYPE OF (Nom_super-entité)
... ... ...
```

18. http://www.wurvoc.org/vocabularies/om-1.8/
19. http://www.iso.org/iso/catalogue_detail.htm?csnumber=18348

```
... ... ...
END_ENTITY;
```

Connaissance descriptive

Les entités dans le langage EXPRESS sont décrites par des attributs qui caractérisent leurs instances par des valeurs (ou des collections de valeurs). Les attributs peuvent aussi être représentés par d'autres entités (ou collections d'autre entités) dans le cas de relations avec les autres entités. Il existe deux types d'attributs : (i) les attributs libres qui sont définis indépendamment de tout autre attribut et (ii) les attributs dérivés qui sont calculés à partir des valeurs d'autres attributs. A chaque attribut est associé un type de données qui est choisi parmi les quatre familles définies dans EXPRESS :

— *Les types simples* : ce sont les types de base : chaines de caractères (STRING), numériques (Real, Binary, Integer) ou logiques (Logical, Boolean) ;

— *Les types nommés* : ce sont les types construits à partir de types existants. Des noms sont associés à ces types qui peuvent être définis par des énumérations (Enumeration) ou par des sélections alternatives de types (Select) ;

— *Les types agrégats* : se sont les types collections : les ensembles (Set), les ensembles multiévalués (Bag), les listes (List) et les tableaux (Array).

— l*es types entités* : se sont les types représentant les associations.

Exemple générique :

```
ENTITY Nom_entité;
nom_attribut_1  Nom_entité_cible;
nom_attribut_2  Type_de_données;
... ... ... ...
... ... ... ...
END_ENTITY;
```

Connaissance procédurale

L'un des avantages du langage EXPRESS est qu'il est très expressif au niveau des contraintes. Ces dernières se répartissent en deux familles :

1. *Les contraintes locales* : spécifiques aux entités, elle sont déclarées à l'aide du mot clé "Where" et portent sur les attributs de l'entité concernée.
2. *Les contraintes globales* : elles nécessitent une vérification globale sur l'ensemble des instances des entités. Elles sont de trois types :

— *Les contraintes d'unicité* : elles sont déclarées par le mot clé "Unique" et permettent de spécifier l'unicité de valeur des attributs concernés sur l'ensemble de la population d'instances d'une même entité.

— *Les contraintes de cardinalité* : elles sont déclarées par le mot clé "Inverse". Elles permettent de contraindre le nombre d'entités d'un certain type qui référencent une entité donnée dans un certain rôle.

— *Les contraintes assertionnelles globales* : elles sont déclarées en dehors des entités par le mot clé "Rule". Elles s'expriment sous forme de prédicats s'appliquant à l'ensemble d'une (ou plusieurs) population d'entités.

Le langage EXPRESS permet d'implémenter des fonctions et procédures utilisées dans l'écriture des fonctions de dérivations et des prédicats servant à définir les contraintes. Il propose un ensemble de fonctions prédéfinies comme par exemple : QUERY (requête sur les instances d'une classe), SIZEOF (taille d'une collection), TYPEOF (introspection : donne le type d'un objet) et USED_IN (calcul dynamique des associations inverses).

2.2.2 Représentation des instances : fichier physique

Les modèles EXPRESS sont associés à un format textuel de représentation d'instances permettant l'échange entre systèmes informatiques appelé *fichier physique* (ISO 10303-21/1994) [20]. Chaque instance est : (1) identifiée par un identificateur d'objet (OID : Object IDentifier) qui utilise la notation '#x', (2) caractérisée par le nom de la classe instanciée et (3) décrite par la liste des valeurs de ses attributs qui sont représentées entre parenthèses.
Le fichier physique doit respecter l'ensemble des contraintes suivantes :

— l'ordre de définition des attributs dans le modèle doit être respecté ;

— les attributs optionnels non valués prennent la valeur $;

— les attributs dérivés et inverses ne sont pas représentés ;

— les attributs redéfinis sont valués à * ;

— une liste vide est représentée par ().

2.2.3 Représentation graphique : EXPRESS-G

A l'exception des contraintes, le langage EXPRESS possède un formalisme de représentation graphique appelé EXPRESS-G. Cette forme graphique a pour objectif de donner une vue synthétique d'un schéma EXPRESS et est adaptée aux phases préliminaires d'une conception. EXPRESS-G comprend trois catégories de symboles graphiques :

— *les définitions* : elles sont utilisées pour représenter les types simples, les types nommés et la déclaration des schémas. Une définition est représentée par une boîte contenant le nom

20. http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=20580

du type ou du schéma représenté. Comme le montre la figure 4.3, les boîtes diffèrent selon le type représenté, notamment par la nature du trait (pointillé ou plein). Les définitions de fonctions, de procédures ou de contraintes ne peuvent pas être représentées en EXPRESS-G.

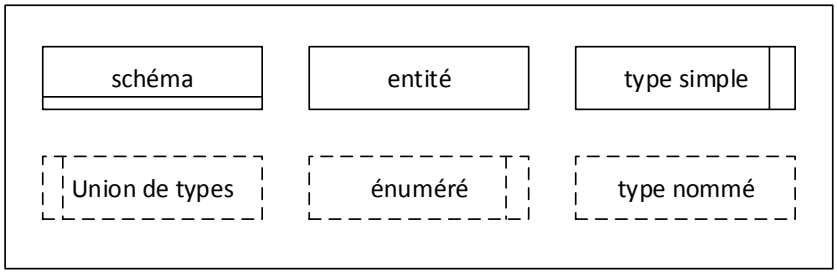


FIGURE 4.3 – La représentation des définitions en EXPRESS-G

— *les relations* : elles sont utilisées pour représenter les relations entre définitions. Les relations sont spécifiées par des lignes qui différent selon la nature des relations : (i) un trait discontinu est utilisé pour relier un attribut facultatif à son entité, (ii) une ligne à trait épais est utilisée pour la relation d'héritage et (iii) toutes les autres relations sont spécifiées par un trait continu simple. Un cercle indique la direction de la relation. Dans une relation entité - attributs, le cercle est spécifié du côté de la boite pour représenter l'attribut et dans une relation d'héritage, le cercle est dessiné du coté des sous-types. Comme le montre la figure 4.4, les relations peuvent être annotées pour indiquer les noms des attributs, pour préciser les cardinalités dans le cas des agrégations ou d'attributs inverses ou encore pour qualifier les relations d'héritage.

— *les compositions* : elles permettent aux schémas d'être représentés sur plusieurs pages.

2.2.4 Choix d'EXPRESS comme langage de modélisation

L'objectif principal de cette première contribution est de proposer un modèle générique et partageable pour représenter le contexte. Par conséquent, ce modèle doit être formellement défini afin d'éliminer toute ambiguïté possible. Nous présentons ici les raisons qui nous ont poussés à choisir EXPRESS et nous expliquons pourquoi ce langage est bien adapté pour la définition de notre modèle de contexte.

L'accent principal dans le langage EXPRESS est mis sur la précision et la rigueur dans la définition des modèles et plus particulièrement sur les contraintes que les données doivent respecter pour être approuvées comme conformes à ces modèles. Son avantage majeur est sa complétude, puisqu'il permet de décrire des concepts structuraux, descriptifs et procéduraux dans un formalisme clair.

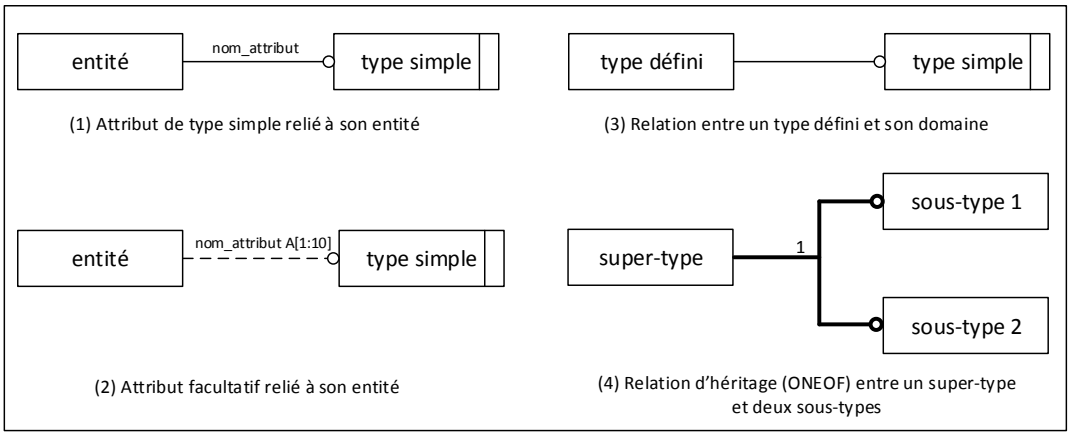


FIGURE 4.4 – La représentation des relations en EXPRESS-G

Les caractéristiques suivantes résument les motivations de notre choix pour le langage EXPRESS :

— c'est un langage de modélisation orienté objet qui est conforme à la philosophie MOF ;

— il intègre un outil de vérification de modèle ;

— il dispose d'un mécanisme puissant de spécification de contraintes ;

— il permet un prototypage rapide du fait de sa syntaxe claire ;

— sa version graphique EXPRESS-G permet de donner une vue synthétique et facilement lisible des modèles.

3 Notion de contexte

Comme nous l'avons vu dans dans le chapitre précédent, la notion de contexte a été étudiée depuis plusieurs années dans plusieurs domaines de recherche. Dans cette section, nous détaillons notre vision de cette notion. Nous commençons d'abord par présenter notre définition du contexte. Puis, en se basant sur un cas d'étude réel issu du domaine médical, nous présentons ses différentes dimensions et catégories.

3.1 Notre vision du contexte

Le contexte a été étudié dans divers domaines de l'informatique et, en fonction des spécificités du domaine d'application et des perspectives offertes, de nombreuses interprétations de cette notion ont émergé. Au regard de la littérature dont un résumé est présenté dans le chapitre

précédent, nous avons constaté qu'il n'existe pas de définition unique et acceptable par toutes les communautés de recherche. En effet, les définitions proposées sont soit très générales soit très spécifiques à un domaine particulier.

Comme nous l'avons précisé, la définition la plus répandue du contexte est celle proposée par *Abowd et al.* [2] qui le définissent comme: « *toute information pouvant être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet considéré comme pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application eux mêmes* ».

Bien que nous estimons que la vision fournie par cette définition correspond d'une manière générale au domaine de l'ingénierie des données, nous trouvons qu'elle est susceptible d'être source de conflit du fait qu'elle ne précise pas les limites du contexte. De ce fait, nous proposons la définition suivante : « *le contexte est toute information, interne ou externe aux sources de l'application, qui peut être utilisée pour caractériser les données de cette dernière en spécifiant leurs définitions, leurs utilisations et/ou leurs évaluations* ». Nous cherchons, à travers cette définition, à mettre l'accent sur la donnée. En effet, le contexte, selon notre vision, est toute information qui précise les différentes définitions, utilisations ou évaluations des concepts et propriétés modélisant les données en question.

3.2 Notre catégorisation du contexte

Vu la diversité des informations composant le contexte et afin de faciliter son utilisation, les chercheurs ont essayé, suivant différentes perspectives, de le classer par catégories. Un état de l'art détaillé de ces schémas de catégorisation est présenté dans le chapitre 3. Dans ce qui suit, nous introduisons d'abord un cas d'étude issu du domaine de la santé et nous détaillons ensuite les différentes catégories du contexte que nous avons définies tout en illustrant chacune d'elles avec un exemple.

3.2.1 Cas d'étude

Nous présentons ici un cas d'étude que nous allons utiliser tout au long de ce chapitre pour les exemples explicatifs. Ce cas d'étude, issu du domaine de la santé, repose sur l'ontologie de domaine *Healthcare Ontology* que nous avons construite et validée en collaboration avec le staff de l'institution "Assistance Publique des Hôpitaux de Marseille"(APHM) dans un ancien projet commun [109]. Cette ontologie, dont un extrait est présenté dans la figure 4.5, définit formellement le domaine de la santé. Les différentes informations concernant les activités médicales de l'APHM sont enregistrées dans des Bases de Données. Ces dernières stockent entre autres les différentes mesures prises pour les patients (températures, fréquences cardiaques, fréquences respiratoires, pression artérielle, etc) ainsi que des indicateurs concernant leurs séjours en hôpital (durée moyenne de séjours, taux d'occupation des lits, etc).

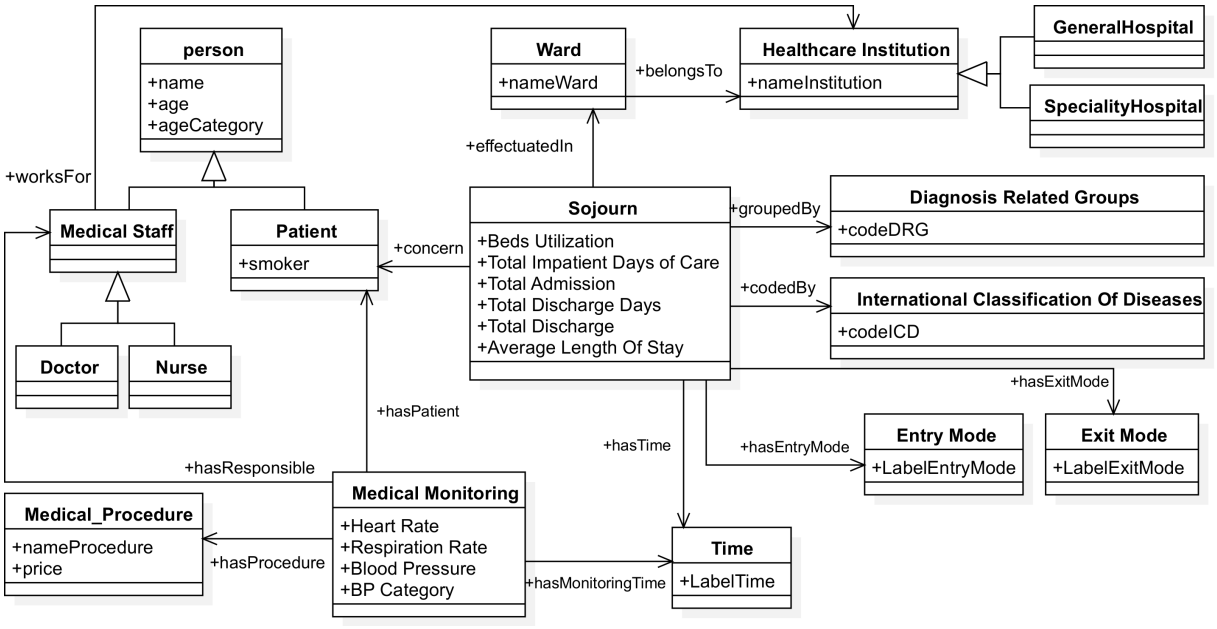


FIGURE 4.5 – Extrait de l'ontologie définissant le domaine de la santé

La table 4.1 présente un glossaire des termes médicaux utilisés dans ce chapitre ainsi que dans le reste du mémoire de thèse.

3.2.2 Catégories du contexte

Afin de mieux cerner la notion de contexte et de faciliter son utilisation, nous proposons ici, en se basant sur notre vision discutée çi-dessus, une classification des informations contextuelles par catégories tout en identifiant les différentes entités concernées par cette notion. Nous distinguons entre deux principales catégories de contexte :

Contexte de définition

Il caractérise la définition d'un concept donné. Concrètement, une *définition contextuelle* d'un concept 'c' est représentée par une expression sur les autres ressources du domaine (concepts et propriétés). Un concept peut avoir plusieurs définitions contextuelles et le regroupement de ces définitions, en utilisant par exemple des opérations comme l'union ou l'intersection, constitue *une définition globale*.

Exemple 1

(1) Dans le contexte c_1, un patient est défini comme une personne qui a effectué au moins un séjour à l'hôpital;
(2) Dans le contexte c_2, un patient est défini comme une personne qui a au moins un suivi médical.

Terme	Description
ALOS	Average Length Of Stay (Durée Moyenne de séjours) : le nombre moyen de jours que les patients passent à l'hôpital
TIDC	Total Inpatient Days Of Care : la somme des recensements quotidiens des hospitalisations pendant la période examinée
TA	Total Admissions : le nombre total des admissions dans les unités d'hospitalisation de l'hôpital au cours de la période examinée
TDD	Total Discharge Days : la somme du nombre de jours passés à l'hôpital pour chaque patient hospitalisé dont la sortie s'est effectuée au cours de la période examinée
TD	Total Discharge : le nombre des sorties de l'hôpital au cours de la période examinée
BUR	Beds Utilization Rate (Taux d'Occupation des Lits) : le taux d'occupation des lits disponibles dans l'hôpital
HR	Heart Rate (Fréquence Cardiaque) : le nombre de battements cardiaques (ou pulsations) par unité de temps
RR	Respiration Rate (Fréquence Respiratoire) : le nombre de cycles respiratoires par minute ou, plus formellement, le nombre de mouvements indicatifs d'inspiration et expiration par unité de temps

TABLE 4.1 – Glossaire des termes médicaux

La définition globale du concept "Patient" est l'union de ces deux définitions contextuelles :
$Patient_{[G]} = Patient[c_1] \cup Patient[c_2]$*:*
$Patient[c_1] = Person \sqcap (\exists hasSojourn.Sojourn)$;
$Patient[c_2] = Person \sqcap (\exists hasMonitoring.MedicalMonitoring)$.

Contexte d'évaluation

Cette catégorie concerne les informations caractérisant l'évaluation des propriétés. Concrètement, un contexte d'évaluation est représenté par une relation de dépendance entre deux types de propriétés : (1) *les propriétés contextualisées* : ce sont les propriétés dont l'évaluation dépend du contexte (par exemple le poids d'une personne), et (2) les *paramètres de contexte* (ou *propriétés contextualisantes*) : ce sont les propriétés qui caractérisent l'évaluation des propriétés contextualisées (par exemple le temps).

Nous distinguons entre trois sous-catégories :

1. Contexte Local : cette catégorie représente le contexte interne (local) dans lequel les valeurs des propriétés contextualisées sont évaluées. Elle ne concerne que les propriétés stockées dans les sources de l'application (la BDD). Nous distinguons entre deux types :

— **Fonction Mathématique :** dans ce type de contexte, la relation entre les propriétés contextualisées et les paramètres de contexte prend la forme d'une fonction mathématique.

Exemple 2

La durée moyenne de séjour des patients (ALOS) dépend du contexte. En effet, suivant la nature des séjours, sa formule de calcul change : (i) pour les longs séjours, la formule suivante est utilisée : "Average Length Of Stay = Total Discharge Days / Total Discharge", (ii) pour les courts séjours, une autre formule est utilisée : "Average Length Of Stay = Total Inpatient Days Of Care / Total Admissions".

— **Dépendance Fonctionnelle :** dans ce cas, l'information contextuelle est représentée par une relation de type dépendance fonctionnelle [21] entre la propriété contextualisée et ses paramètres de contexte.

Exemple 3

La catégorie de la tension artérielle d'un patient qui peut être faible, normale ou élevée dépend de quelques caractéristiques du patient (paramètres contextuels). En effet, la catégorie de la pression artérielle est fonctionnellement dépendante de : l'age du patient, le fait qu'il fume ou non et de la mesure de la pression elle même (voir la table 4.2).

Age	Fumeur	Tension Artérielle (TA)	Catégorie de la TA
Bébé	-	> 11	Elevée
Adulte	Oui	> 14	Elevée
Senior	Oui ou Non	> 14	Elevée
Bébé	-	Entre 9 et 11	Normale
Adulte	Oui	Entre 10 et 14	Normale
...	...	...	...

TABLE 4.2 – Extrait de la connaissance experte

2. Contexte Environnant : à l'inverse du Contexte Local, cette catégorie représente le contexte externe dans lequel les valeurs des propriétés contextualisées sont évaluées. Nous considérons dans ce travail les deux paramètres de contexte les plus utilisés dans la littérature, à savoir le *temps* et la *localisation*. Nous rappelons que d'autres paramètres peuvent être considérés suivant les besoins applicatifs de chacun.

— **Contexte Temporel :** dans cette catégorie, l'évaluation des propriétés en question dépend du paramètre de contexte "temps".

21. http://databases.about.com/cs/specificproducts/g/functdep.htm

Exemple 4

Le prix d'une consultation médicale dépend du temps (la validité des tarifs).

— **Contexte Spatial :** par analogie avec la précédente catégorie, l'information contextuelle est représentée par l'évaluation des propriétés considérées dépendamment du contexte spatial.

3. Unités de Mesure : dans cette catégorie, l'information contextuelle consiste à expliciter les unités de mesure (ou les unités monétaires) pour chaque propriété concernée.

Exemple 5

Le prix d'une consultation médicale dépend de la devise utilisée (Euro, Dollars, etc).

4 Notre approche de modélisation de contexte

Dans la sous-section 4.1.2 du chapitre précédent, nous avons conclu qu'aucune approche de modélisation de contexte ne répond à l'ensemble de nos exigences. C'est pourquoi, après avoir précisé notre vision de contexte et avoir détaillé ses différents composants, nous présentons dans cette section notre modèle générique et formel de contexte en utilisant le langage EXPRESS et ce tout en expliquant le lien de ce modèle avec les ontologies.

4.1 Notre modèle de Contexte

En se basant sur les différentes catégories de contexte que nous avons identifiées, nous proposons un nouveau modèle qui répond à l'intégralité de nos exigences. En effet, le modèle proposé :

— est conforme à notre vision du contexte et identifie clairement toutes ses dimensions (Exigence 1) ;

— est générique et peut être utilisé par n'importe quelle application dans le domaine de l'ingénierie des données (Exigence 2) ;

— est formellement défini avec le langage EXPRESS et utilise, dans la mesure du possible, des standards ou des ontologies de domaine pour définir ses différents éléments (Exigence 3);

— est défini au niveau conceptuel (Exigence 4) ;

— prend en charge les paramètres internes et externes du contexte (Exigence 5 et 6) ;

— les contextes définis sont attachés aux ontologies qui décrivent la sémantique des données manipulées (Exigence 7).

En plus, notre approche offre un mécanisme de persistance (Exigence 8) et un langage d'interrogation dédié (Exigence 9) comme nous allons le voir dans le chapitre suivant.

Nous présentons maintenant notre modèle de contexte dont la représentation en EXPRESS-G est illustrée dans la figure 4.6. Comme le montre cette dernière, l'entité racine du modèle est 'Context'. Elle caractérise chaque contexte avec deux attributs : (i) 'code' pour donner un identifiant unique et (ii) 'name' pour décrire le contexte avec un terme linguistique.

Notre modèle définit, à partir de cette entité racine, une hiérarchie d'entités représentant notre catégorisation du contexte. Nous détaillons dans ce qui suit les principaux composants de notre modèle.

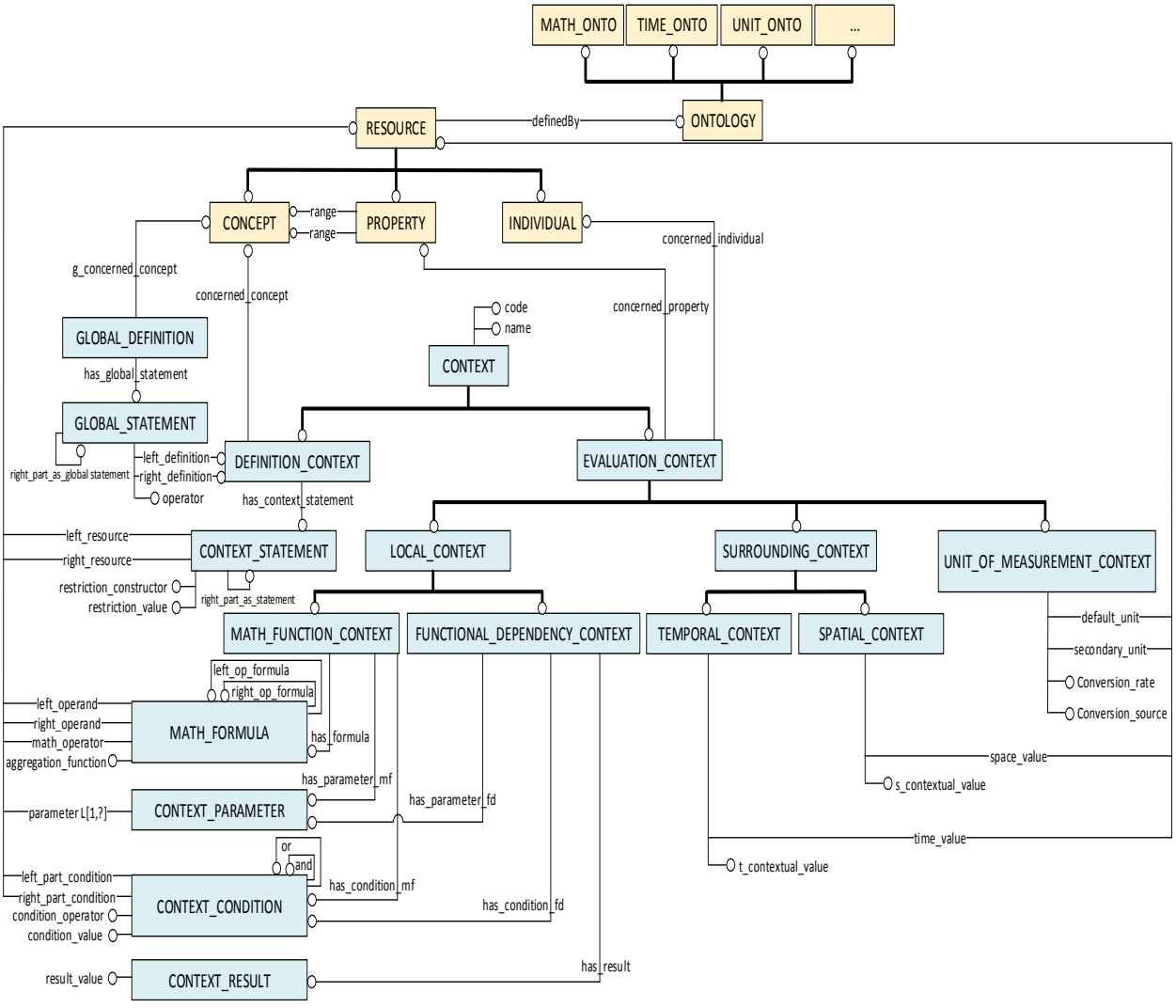


FIGURE 4.6 – Représentation EXPRESS-G de notre modèle générique du contexte

4.1.1 L'entité 'Definition Context'

Cette entité représente la catégorie *Contexte de définition*. Comme le montre la figure 4.6, elle associe, grâce à l'association *concerned_concept* qui la lie à l'entité *Concept*, les concepts ontologiques avec leurs contextes de définition correspondants. Elle est liée également avec les entités 'Context Statement', 'Global Definition' et 'Global Statement' que nous détaillons dans ce qui suit :

L'entité 'Context Statement'

Cette entité sert à représenter les expressions qui composent la définition contextuelle. En considérant qu'une expression prend la forme <Partie gauche> <Opérateur> <Partie droite>, cette entité comporte cinq attributs :

- *left_resource* et *right_resource* : ils représentent les ressources ontologiques impliquées dans l'expression contextuelle respectivement comme partie gauche et droite ;
- *restriction_constructor* : il représente l'opérateur de restriction;
- *restriction_value* : il représente la valeur de restriction ;
- *right_part_as_statement* : il sert à formuler des expressions complexes en imbriquant les expressions (eg. définir l'expression 1 comme la partie droite de l'expression 2).

La table 4.3 présente les spécifications EXPRESS des entités 'Context Definition' et 'Context Statement' et illustre l'instanciation de notre modèle pour l'exemple suivant : « $\text{Staff}_{[Context_1]}$ = Doctor ∪ Nurse ». Nous précisons que la notation Concept(c) désigne le concept ontologique 'c'.

L'entité 'Global Definition'

Cette entité sert à représenter la définition globale du concept. Elle associe, grâce à l'association *g_concerned_concept* qui la lie à l'entité *Concept*, les concepts ontologiques avec leurs définitions globales. Elle est également liée à l'entité *Global Statement*.

L'entité 'Global Statement'

Cette entité représente les expressions qui composent la définition globale. Elle comporte quatre attributs :

- *left_definition* et *right_definition* : ils représentent les définitions contextuelles qui composent la définition globale en tant que partie gauche et partie droite respectivement ;
- *operator* : il représente l'opérateur utilisé pour regrouper les définitions contextuelles ;

TYPE DL_Constructor = **ENUMERATION OF** (‘nagation’, ‘Intersection’, ‘Union’, ‘Existential quantification’,‘Universal quantification’) **END_TYPE**;
TYPE LITERAL = **SELECT** (STRING, INT, REAL) **END_TYPE**;
ENTITY CONTEXT code: INTEGER ; name: STRING ; **END_ENTITY**;
ENTITY CONTEXT_STATEMENT left_ressource: RESOURCE; right_ressource: RESOURCE; restriction_constructor: DL_Constructor; restriction_value: LITERAL; right_part_as_statement: CONTEXT_STATEMENT; **END_ENTITY**;
#10 = CONTEXT_STATEMENT (CONCEPT(Doctor), CONCEPT(Nurse), Union, $).
ENTITY CONTEXT_DEFINITION **SUBTYPE OF** (CONTEXT); concerned_concept: CONCEPT; has_context_statement: CONTEXT_STATEMENT; **END_ENTITY**;
#100 = CONTEXT_DEFINITION (100, ‘Context$_1$’, CONCEPT(Staff), #10).

TABLE 4.3 – Spécification en EXPRESS des entités ‘Context ’, ‘Context Statement’ et ‘Context Definition’

— *right_part_as_global_statement* : il sert à formuler les expressions complexes en imbriquant les expressions.

La table 4.4 présente les spécifications EXPRESS des entités ‘Global Definition’ et ‘Global Statement’ et illustre l’instanciation de notre modèle pour l’exemple suivant : « $\text{Patient}_{[G]} = \text{Patient}_{[C1]} \cup \text{Patient}_{[C2]}$ ». Afin de faciliter la lecture de la table, nous supposons ici que les définitions contextuelles C1 et C2 soient définies par les identifiants #20 et #30 .

4.1.2 L’entité ‘Evaluation Context’

Cette entité représente la catégorie *Contexte d’évaluation*. Elle associe, grâce à l’association *concerned_property* qui la relie à l’entité *Property*, les propriétés ontologiques avec leurs

TYPE OPERATOR = **ENUMERATION OF** ('Intersection', 'Union') **END_TYPE**;
ENTITY GLOBAL_STATEMENT left_definition: DEFINITION_CONTEXT; right_definition: DEFINITION_CONTEXT; operator: OPERATOR; right_part_as_global_statement: GLOBAL_STATEMENT; **END_ENTITY**;
#40 = GLOBAL_STATEMENT (#20, #30, Union, $).
ENTITY GLOBAL_DEFINITION g_concerned_concept: CONCEPT; has_global_statement: GLOBAL_STATEMENT; **END_ENTITY**;
#200 = GLOBAL_DEFINITION (CONCEPT(Patient), #40).

TABLE 4.4 – Spécification en EXPRESS des entités 'Global Statement'et 'Global Definition'

contextes d'évaluation correspondants. Elle contient l'association facultative *concerned_individual* qui représente l'individu concerné par le contexte (des exemples explicatifs sont fournis dans le paragraphe 4.1.2). L'entité 'Evaluation Context' regroupe deux entités :

L'entité 'Local Context'

Cette entité représente la catégorie *Contexte local*. Elle est divisée en deux entités 'Math Function Context' et 'Functional Dependency Context':

1. Math Function Context : cette entité représente la sous-catégorie *Fonction Mathématique* et est composée de trois entités :

— **Context Parameter :** cette entité représente les paramètres de contexte (les propriétés contextualisantes).

La table 4.5 présente la spécification en EXPRESS de cette entité et illustre des exemples d'instanciation. Nous rappelons que la notation PROPERTY(P) désigne la propriété ontologique 'P'.

— **Context Condition** : cette entité représente les conditions qui doivent être satisfaites dans l'information contextuelle. Ces conditions sont formalisées dans notre modèle comme suit : <Partie gauche (propriété)> <Opérateur> <Partie droite (propriété ou valeur)>. Nous pouvons avoir par exemple : « NomService = Psychiatrie ». En effet, l'entité 'Context Condition' de notre modèle est caractérisée par les attributs suivants :

ENTITY CONTEXT_PARAMETER parameter: LIST[1:?] OF PROPERTY; **END_ENTITY**;
#50 = CONTEXT_PARAMETER (PROPERTY (nameWard), PROPERTY (TDD), PROPERTY (TD)).
#51 = CONTEXT_PARAMETER (PROPERTY (BP), PROPERTY (Age_Category), PROPERTY (Smoker)).

TABLE 4.5 – Spécification en EXPRESS de l'entité 'Context Parameter'

- *left_part_condition* et *right_part_condition* qui représentent les propriétés ontologiques impliquées dans la condition respectivement comme partie gauche ou droite ;
- *condition_operator* : qui représente l'opérateur de comparaison ;
- *condition_value* : qui représente la valeur de comparaison ;
- *and_condition* : qui sert à imbriquer les conditions en utilisant l'opérateur 'and';
- *or_condition* : qui sert à imbriquer les conditions en utilisant l'opérateur 'or'.

La table 4.6 présente la spécification en EXPRESS de l'entité 'Context Condition' et illustre l'instanciation du modèle pour les deux exemples suivants : Condition 1 (#60) ≡ (ward_Name = 'Psychiatry') et Condition 2 (#63) ≡ [(BP ≥ 14) and (Age_Category = 'Adult') and (Smoker = true)].

- **Math Formula :** cette entité représente la formule mathématique utilisée pour calculer les propriétés contextualisées. Elle comporte six attributs :
  - *aggregation_function* : il correspond aux fonctions d'agrégation ;
  - *math_operator*: il définit les opérateurs mathématiques utilisés dans la formule de calcul ;
  - *left_operand* et *right_operand* : ils correspondent aux propriétés ontologiques jouant respectivement les rôles des opérandes gauche et droit de la formule de calcul ;
  - *left_operand_formula* et *right_operand_formula* : ces attributs servent à définir des formules complexes en utilisant une formule comme opérande gauche ou droit d'une autre formule.

La définition des opérateurs utilisés dans la formule mathématique peut être réalisée soit en définissant une classe d'énumération soit en se référant à des ontologies de domaine ou bien des standards qui définissent les opérateurs mathématiques. Nous pouvons citer à titre d'exemples, l'ontologie *EngMath* [22] qui couvre tous les aspects de la modélisation mathématique en ingé-

22. http://www-ksl.stanford.edu/knowledge-sharing/papers/engmath.html

TYPE Comp_Operator = **ENUMERATION OF** ('Equal', 'Less than', 'Greater than') **END_TYPE**;
TYPE LITERAL = **SELECT** (INT, REAL, STRING) **END_TYPE**;
ENTITY CONTEXT_CONDITION left_part_condition: PROPERTY; right_part_condition: PROPERTY; condition_operator: Comp_Operator; condition_value: LITERAL; and_condition: LIST [0:?] CONTEXT_CONDITION; or_condition: LIST [0:?] CONTEXT_CONDITION; **END_ENTITY**;
#60 = CONTEXT_CONDITION (PROPERTY(name_Ward),$,'Equal','Psychiatry',$,$).
#61 = CONTEXT_CONDITION (PROPERTY(Age_Category),$,'Equal', 'Adult',$,$).
#62 = CONTEXT_CONDITION (PROPERTY(Smoker),$,'Equal', 'True',$,$).
#63 = CONTEXT_CONDITION (PROPERTY(BP),$,'Greater than', 14,[(#61),(#62)],$).

TABLE 4.6 – Spécification en EXPRESS de l'entité 'Context Condition'

nierie ou bien le standard *OPENMATH*[23] qui sert à représenter et échanger des objets mathématiques et qui offre une description sémantique des différents opérateurs mathématiques grâce à une série de dictionnaires appelés Content Dictionaries (CDs).

La table 4.7 présente la spécification en EXPRESS de l'entité 'Math Formula' et illustre l'instanciation pour l'exemple suivant : Formule ≡ TDD ÷ TD.

Après avoir donné les spécifications EXPRESS des entités composant l'entité principale 'Math Function Context', nous définissons cette dernière comme montré dans la table 4.8. La partie inférieure de la table illustre l'instanciation du modèle pour l'exemple du contexte 'Long Sojourn' qui définit les paramètres de contexte (#50) et la condition (#60) et fixe la formule (#70) comme formule de calcul de la propriété *ALOS*.

2. Functional Dependency Context : cette entité représente la catégorie *Dépendance Fonctionnelle*. Elle est composée de trois entités :

— **Context Parameter** et **Context Condition** : ces entités sont les mêmes entités présentées précédemment. En effet, elles sont partagées par les deux entités 'Math Function Context' et 'Functional Dependency Context'.

— **Context Result** : cette entité est utilisée pour associer, suivant la condition définie, la valeur appropriée à la propriété contextualisée. Elle comporte deux attributs :

23. http://www.openmath.org/

TYPE Aggregate_Function = **ENUMERATION OF** (avg, sum, min, max) **END_TYPE**;
ENTITY MATH_FORMULA aggregate_function: Aggregate_Function; operator: EngMath_Ontology_Instance; left_operand: PROPERTY; right_operand: PROPERTY; right_operand_formula: MATH_FORMULA; **END_ENTITY**;
#70 = MATH_FORMULA ($, EngMath_Ontology_Instance (‘÷’), PROPERTY(‘TDD’), PROPERTY(‘TD’), $).

Table 4.7 – Spécification en EXPRESS de l’entité ‘Math Formula’

ENTITY EVALUATION_CONTEXT **SUBTYPE OF** (CONTEXT); concerned_property: PROPERTY; concerned_individual: INDIVIDUAL; **END_ENTITY**;
ENTITY LOCAL_CONTEXT **SUBTYPE OF** (EVALUATION_CONTEXT); **END_ENTITY**;
ENTITY MATH_FUNCTION_CONTEXT **SUBTYPE OF** (LOCAL_CONTEXT); has_parameter_mf: CONTEXT_PARAMETER ; has_condition_mf: CONTEXT_CONDITION ; has_formula: MATH_FORMULA ; **END_ENTITY**;
#300 = MATH_FUNCTION_CONTEXT (300, ‘Long Sojourn’, PROPERTY(‘ALOS’), $, #50, #60, #70).

Table 4.8 – Spécification en EXPRESS de l’entité ‘Math Function Context’

— *result_operator* : il représente l’opérateur de comparaison. Souvent, le plus utilisé est l’opérateur d’affectation ‘=’;

— *result_value* : il représente la valeur que la propriété contextualisée doit prendre conformément à la condition définie.

La table 4.9 présente la spécification en EXPRESS de l’entité ‘Context Result’ et illustre

l'instanciation du modèle pour le résultat suivant : (= 'High').

TYPE Comp_Operator = **ENUMERATION OF** (Equal, Less than, Greater than) **END_TYPE**;
TYPE LITERAL = **SELECT** (INT, REAL, STRING) **END_TYPE**;
ENTITY CONTEXT_RESULT result_operator: Comp_Operator; result_value: LITERAL; **END_ENTITY**;
#80 = CONTEXT_RESULT (Equal, High).

TABLE 4.9 – Spécification en EXPRESS de l'entité 'Context Result'

Après avoir donné les spécifications EXPRESS des entités composant l'entité principale 'Functional Dependency Context', nous la définissons de façon identique comme montré dans la table 4.10. La partie inférieure de la table illustre l'instanciation du modèle pour l'exemple du contexte 'HighBP'qui définit les paramètres (#51), la condition (#62) et le résultat (#80).

ENTITY FUNCTIONAL_DEPENDENCY_CONTEXT **SUBTYPE OF** (LOCAL_CONTEXT); has_parameter_fd: CONTEXT_PARAMETER ; has_condition_fd: CONTEXT_CONDITION ; has_result: CONTEXT_RESULT ; **END_ENTITY**;
#400 = FUNCTIONAL_DEPENDENCY_CONTEXT (400, 'HighBP ', PROPERTY ('BPCategory'), $, #51, #62, #80).

TABLE 4.10 – Spécification en EXPRESS de l'entité 'Functional Dependency Context'

L'entité 'Surrounding Context'

Cette entité de notre modèle représente la catégorie *Contexte Environnant*. Elle est composée de deux entités 'Temporal Context' et 'Spatial Context':

1. **L'entité 'Temporal Context' :** elle consiste en deux attributs :
   - *time_value* : il représente la valeur du paramètre du contexte 'temps'. Il récupère ses valeurs de l'ontologie externe *Time Ontology* [24] (TO) qui définit formellement les concepts temporels ;

24. http://www.w3.org/2006/time#

— *t_contextual_value* : il représente la valeur que doit prendre la propriété contextualisée concernée suivant la valeur du paramètre de contexte.

2. L'entité 'Spatial Context' : elle contient deux attributs :

— *space_value* : il représente la valeur du paramètre du contexte 'localisation'. Il prend ses valeurs de l'ontologie externe *Space Ontology* [25] qui définit un vocabulaire complet pour la gestion de la localisation ;

— *s_contextual_value* : il représente la valeur que doit prendre la propriété contextualisée concernée suivant la valeur du paramètre de contexte.

La table 4.11 présente la spécification en EXPRESS de l'entité 'Temporal Context' et illustre l'instanciation du modèle pour l'exemple du contexte 'Tariff_2013' qui définit l'année '2013' comme valeur du paramètre 'Temps' et affecte la valeur '20' à la propriété 'price' de l'individu 'General Practice Consultation'.

ENTITY SURROUNDING_CONTEXT **SUBTYPE OF** (CONTEXT); **END_ENTITY**;
ENTITY TEMPORAL_CONTEXT **SUBTYPE OF** (SURROUNDING_CONTEXT); time_value: TIME_ONTOLOGY_INSTANCE; t_contextual_value: LITERAL; **END_ENTITY**;
#500 = TEMPORAL_CONTEXT (500, 'Tariff_2013', PROPERTY ('price'), INDIVIDUAL ('General Practice Consultation'), TIME_ONTOLOGY_INSTANCE ('year 2013'), 20).

TABLE 4.11 – Spécification en EXPRESS de l'entité 'Temporal Context'

L'entité 'Unit Of Measurement Context'

Cette entité représente la catégorie *Unités de mesures*. Elle comporte quatre attributs :

— *default_unit* : il sert à expliciter l'unité de mesure ou la devise utilisée par défaut pour évaluer la propriété en question ;

— *secondary_unit* : il sert à expliciter l'unité de mesure ou la devise éventuelle qui peut être utilisée pour évaluer les propriétés concernées ;

— *conversion_rate* : il représente le taux de conversion entre les unités concernées ;

25. http://sweet.jpl.nasa.gov/1.2/space.owl

— *conversion_source* : il représente l'URL du service de conversion utilisé.

Cette entité est liée à l'ontologie de domaine externe: Ontology of Units of Measure and Related Concepts [26] qui définit formellement les unités de mesures et les devises.
La table 4.12 présente la spécification en EXPRESS de l'entité 'Unit Of Measure Context' et illustre l'instanciation du modèle pour l'exemple du contexte 'USDollar' qui définit, pour la propriété 'price', l'euro comme unité monétaire par défaut, le dollar américain comme une unité secondaire et fixe le taux de conversion entre ces deux dernières.

ENTITY UNIT_OF_MEASUREMENT_CONTEXT **SUBTYPE OF** (CONTEXT); default_unit: UOM_ONTOLOGY_INSTANCE; target_unit: UOM_ONTOLOGY_INSTANCE; conversion_rate: REAL; conversion_source: STRING; **END_ENTITY**;
#600 = UNIT_OF_MEASUREMENT_CONTEXT (600, 'USDollar', PROPERTY ('price'), $, UOM_INSTANCE (EURO), UOM_INSTANCE (USDollar), 1.124, $).

TABLE 4.12 – Spécification en EXPRESS de l'entité 'Unit Of Measurement Context'

4.2 Lien du modèle de contexte avec l'ontologie

Dans la sous-section précédente, nous avons décrit notre modèle de contexte. Ici, nous présentons la dernière partie de notre approche de modélisation qui consiste à lier notre modèle de contexte avec le modèle d'ontologie.

Ce lien consistant à connecter les contextes de notre modèle avec les concepts et les propriétés de n'importe quelle ontologie de domaine existante est concrétisé par les associations suivantes :

— *concerned_concept:* qui associe n'importe quel contexte de définition représenté par l'entité *Definition Context* avec le concept correspondant représenté par l'entité *Concept* ;

— *g_concerned_concept:* qui associe n'importe quelle définition globale représentée par l'entité *Global Definition* avec le concept correspondant représenté par l'entité *Concept* ;

— *concerned_property*: qui associe n'importe quel contexte d'évaluation représenté par l'entité *Evaluation Context* avec la propriété correspondante représentée par l'entité *Property*.

26. http://www.wurvoc.org/vocabularies/om-1.8/

5 Conclusion

Dans ce chapitre, nous avons présenté une approche complète de modélisation du contexte. Cette première contribution a été concrétisée par la réalisation de cinq points essentiels.

En premier lieu, nous avons détaillé notre vision de la notion de contexte et ainsi nous avons pu dégager ses différentes dimensions. En second lieu, nous avons proposé une classification du contexte en différentes catégories. En troisième lieu, nous avons mis en évidence une série d'exigences que notre approche de modélisation doit vérifier et nous avons expliqué les raisons nous poussons à proposer un nouveau modèle de contexte. En quatrième lieu, en tenant compte de tous les points précédents, nous avons proposé un modèle modulaire, partageable et générique du contexte. En effet, (i) pour assurer sa modularité et sa flexibilité, nous avons défini notre modèle de contexte séparément des données et des instances des applications des systèmes d'information exploités, (ii) pour que notre modèle soit générique, nous l'avons défini suite à la généralisation de différentes approches existantes dans le domaine et (ii) pour qu'il soit partageable, nous l'avons défini formellement en utilisant le langage de modélisation EXPRESS afin d'éliminer toute ambiguïté quant à sa définition. En dernier lieu, nous avons défini un lien entre notre modèle de contexte et le modèle d'ontologies afin d'expliciter la sémantique des données manipulées.

Cette approche de modélisation, que nous avons proposée et qui est basée sur la séparation entre la définition du contexte et le développement des ontologies, présente trois avantages principaux. Premièrement, cela évite d'altérer l'aspect consensuel de l'ontologie vu que le contexte est, dans la plupart des cas, défini localement. Deuxièmement, le processus du développement des ontologies est une tache très gourmande en temps et le fait d'y incorporer le contexte conduit à son augmentation. Troisièmement, cette externalisation du contexte offre aux concepteurs une liberté dans le choix du langage de contexte qui peut être différent de celui de l'ontologie.

Dans le chapitre suivant, nous nous concentrerons sur l'intégration de notre modèle de contexte dans les BDS afin de leurs permettre de prendre en charge les informations contextuelles dont dépendent leurs données.

Chapitre 5

Vers une contextualisation des Bases de Données Sémantiques

Sommaire

1 Introduction 100

2 Fondements théoriques 101

2.1 La BDS *OntoDB* 101

2.2 Langage OntoQL 101

2.2.1 Exploitation des ontologies 103

2.2.2 Exploitation des instances 104

2.2.3 Exploitation du méta-modèle 104

2.3 Choix du système OntoDB/OntoQL comme infrastructure de développement 105

3 Persistance du contexte dans les BDS : l'exemple d'OntoDB 105

3.1 Extension du modèle d'OntoDB avec le modèle de contexte proposé 105

3.1.1 Création des constructeurs correspondant à la catégorie "*Contexte de définition*" 106

3.1.2 Création des constructeurs correspondant à la catégorie "*Contexte d'évaluation*" 106

3.2 Mise en œuvre du lien entre le modèle de contexte et le modèle d'ontologie dans OntoDB 109

4 Prise en charge du contexte dans les requêtes : extension du langage OntoQL 110

4.1 Extension de la grammaire d'OntoQL : l'opérateur 'Using Context' 110

4.2 Extension de la sémantique d'OntoQL : Interprétation des requêtes 111

5 Validation : étude de cas 114

5.1 Persistance du fragment de l'ontologie *Healthcare* 115

5.2 Persistance et interrogation des informations contextuelles 115

6 Conclusion 123

1 Introduction

L'évolution des SI au cours de ces dernières années a été marquée par deux défis majeurs : *l'explosion de la quantité* des données manipulées et leur *hétérogénéité* pouvant se situer au niveau syntaxique et/ou sémantique. Ces deux défis caractérisant le *Big Data* ont imposé aux différents acteurs des SI de nouveaux challenges quant au stockage et à la gestion de cette avalanche de données hétérogènes.

Afin de répondre à ces besoins particuliers, les ontologies de domaine, de part leurs capacités d'explicitation de la sémantique des données et de réduction de leurs hétérogénéités, ont été largement utilisées en combinaison avec les systèmes de gestion de données : c'est ainsi que les systèmes de BDS ont été proposés. Ces nouveaux systèmes à base ontologique ont permis effectivement de réduire les problèmes liés à l'hétérogénéité des données. Toutefois, en se basant sur la vision d'ontologie qui consiste à décrire un domaine donné d'une manière consensuelle à partir d'une vision *unique* de la réalité, ces systèmes se sont révélés inefficaces pour la prise en charge d'une autre dimension importante de cette hétérogénéité qui est le contexte.

C'est pourquoi nous proposons, dans ce chapitre, de nous appuyer sur l'approche de modélisation de contexte que nous avons présentée dans le chapitre précédent afin de proposer une approche pour la prise en charge du contexte dans les BDS. Le but de cette approche est d'étendre les modèle des BDS avec notre modèle de contexte afin qu'elles puissent permettre, en plus de la spécification de la sémantique des données, de définir, d'expliciter et d'interroger les différents contextes dont dépendent leurs données.

Afin d'expliquer notre démarche, nous prenons l'exemple du système *OntoDB/OntoQL* qui représente la BDS (*OntoDB*) et son langage d'exploitation (*OntoQL*) comme notre infrastructure de développement et nous détaillons nos contributions qui se résument en deux points essentiels : (i) l'extension du modèle d'*OntoDB* avec notre modèle de contexte et (ii) l'extension du langage *OntoQL* afin de prendre en charge le contexte dans ses requêtes en proposant un opérateur spécifique.

Ce chapitre est organisé en six sections comme suit : la section 2 présente les fondements et notions théoriques sur lesquels se basent nos contributions. Nous introduisons dans cette section la BDS *OntoDB* et son langage d'exploitation *OntoQL*. La section 3 présente l'intégration de notre modèle de contexte dans OntoDB. La section 4 détaille l'extension de la grammaire et de la sémantique du langage *OntoQL*. La section 5 nous permet de valider nos propositions en présentant une expérimentation réalisée sur un cas d'étude issu du domaine médical. La dernière section conclut le chapitre et récapitule les principaux résultats.

2 Fondements théoriques

Nous présentons dans cette section le système OntoDB/OntoQL que nous avons utilisé dans nos travaux. D'abord, nous introduisons la BDS *OntoDB*. Ensuite, nous abordons le langage d'exploitation des BDS *OntoQL*. Enfin, nous justifions notre choix de ce système pour développer notre approche de contextualisation des BDS.

2.1 La BDS *OntoDB*

OntoDB est une BDS développée au sein du Laboratoire d'Informatique et d'Automatique pour les Systèmes (LIAS) [27]. Elle est implémentée sur le SGBD *PostgreSQL* et est basée sur un modèle noyau compatible avec différents formalismes ontologiques. Nous rappelons qu'*OntoDB* a été initialement développée pour stocker les données techniques et qu'elle a été étendue, par la suite, pour prendre en charge d'autres types de données, tels que les données provenant de capteurs placés sur les véhicules électriques [104] ou bien les données géologiques [137].

Comme le montre la figure 5.1, *OntoDB* utilise l'architecture *quatre quarts* qui se décompose en quatre parties : (1) la partie *méta-base*, (2) la partie *méta-schéma*, (3) la partie *ontologie* et (4) la partie *données*. Grâce à son architecture, et précisément la partie *méta-schéma*, *OntoDB* supporte l'évolution du modèle d'ontologies et l'extension de ce dernier en cas de besoin. Pour la description du modèle d'ontologie, *OntoDB* utilise deux tables : (i) la table *Entity* qui définit les entités du méta-schéma telles que *Class* et *Property* décrivant respectivement les classes et les propriétés ; et (ii) la table *Attribute* qui définit les attributs décrivant les entités telles que *name* et *scope* qui représentent respectivement le nom de l'entité et son domaine.

OntoDB stocke les ontologies dans un schéma figé à l'aide des constructeurs ontologiques correspondant à l'instanciation des entités et des attributs définis au niveau méta-schéma. Une approche horizontale est utilisée pour la persistance des données ontologiques. Pour chaque classe de l'ontologie, une table avec des colonnes correspondant au sous-ensemble de propriétés applicables de cette classe est créée.

2.2 Langage OntoQL

Le langage *OntoQL* [97] est un langage de requête pour l'exploitation des BDS développé au laboratoire LIAS. Il est implémenté sur la BDS *OntoDB* et permet, grâce à l'architecture *quatre quarts* de cette dernière, de : (i) définir, modifier et interroger les ontologies, (ii) insérer, modifier et interroger les données et (iii) étendre le modèle d'ontologies en cas de besoin. *OntoQL* utilise une syntaxe proche de celle de SQL.

27. http://lias-lab.fr/

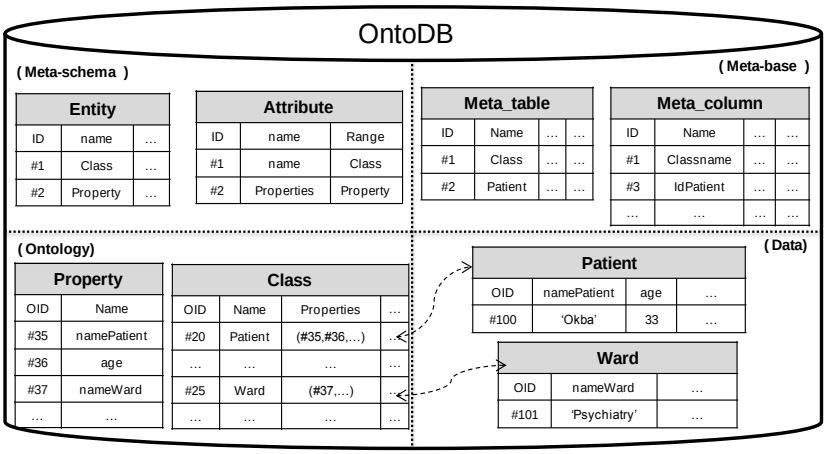


FIGURE 5.1 – Architecture quatre quarts d'OntoDB

Le langage *OntoQL* est basé sur un modèle noyau comportant les principaux constructeurs de modèles d'ontologies et qui peut être étendu. La représentation UML de ce modèle noyau est donnée en figure 5.2. *OntoQL* définit une convention grammaticale permettant de spécifier qu'un élément est du niveau méta-modèle (modèle d'ontologie). Il s'agit de préfixer l'élément en question par le caractère #.

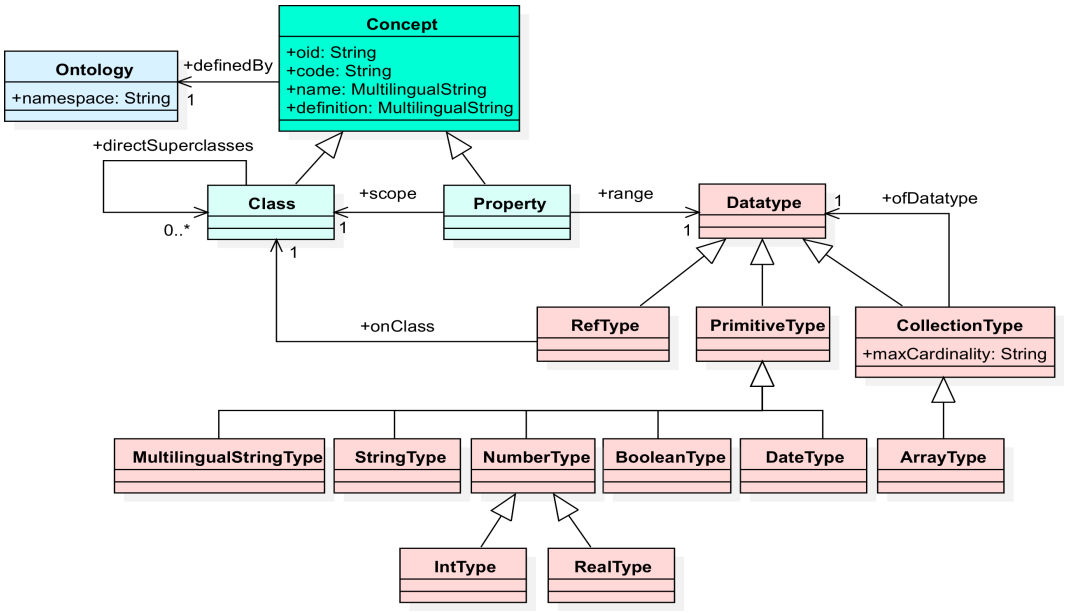


FIGURE 5.2 – Modèle d'ontologies noyau du langage *OntoQL*[99]

Nous présentons dans ce qui suit quelques exemples de requêtes *OntoQL* permettant d'exploiter *OntoDB* à différents niveaux. Nous utiliserons ces types de requêtes pour la mise en œuvre de notre approche.

2.2.1 Exploitation des ontologies

OntoQL permet d'exploiter les ontologies en créant, modifiant et interrogeant leurs classes et propriétés. Par exemple, pour créer et interroger les classes *Person* et *Patient* de notre fragment d'ontologie *Healthcare*, nous utilisons les requêtes suivantes :

Requête 1. Création de la classe *Person* :

```
CREATE #Class Person (
  DESCRIPTOR (
    #name[FR] = 'Personne',
    #version = '001')
  PROPERTIES (
    name STRING,
    age INT))
```

Explication : Cette instruction utilisant la clause **CREATE** permet d'ajouter la classe *Person* en tant qu'instance de *#Class* avec deux attributs : *#name[FR]* : un nom en français et *#version* : le numéro de version. La classe *Person* possède deux propriétés : *name* qui prend des chaînes de caractères et représente le nom de la personne et *age* qui prend des entiers comme valeurs. Elle représente l'âge de la personne.

Requête 2. Création de la classe *Patient* :

```
CREATE #Class Patient UNDER Person (
  DESCRIPTOR (
    #name[FR] = 'Patient',
    #version = '001')
  PROPERTIES (
    smoker BOOLEAN))
```

Explication : Cette instruction utilisant la clause **CREATE** permet d'ajouter la classe *Patient* en tant qu'une instance de *#Class* qui subsume la classe *Person*. Elle est créée avec les deux attributs : *#name[FR]* et *#version* et la propriété *smoker* qui représente le statut *fumeur* ou *non-fumeur* du patient. Elle prend une valeur booléenne.

Requête 3. Rechercher les classes dont la version est différente de '001' :

```
SELECT #name
FROM #class
WHERE #version <> '001'
```

Explication : Cette requête permet de rechercher toutes les classes qui ont été créées avec des attributs *#version* qui ont des valeurs différentes de '001'.

2.2.2 Exploitation des instances

OntoQL permet d'exploiter les données en créant, modifiant, supprimant et recherchant les instances des classes d'ontologies.

Requête 4. Ajouter le patient non fumeur 'Jean' qui a 25 ans :

INSERT INTO Patient (name, age, smoker)
VALUES ('Jean', 25, True)

Explication : Cette instruction utilisant la clause **INSERT INTO** permet de créer une instance de la classe *Patient* en valuant respectivement les propriétés *name*, *age* et *smoker* à 'Jean', 25 et True. Notons que le préfixe # n'est pas utilisé dans le niveau instances.

Requête 5. Rechercher les patients fumeurs :

SELECT p.name
FROM Patient **AS** p
WHERE p.smoker = True

Explication : Cette requête utilisant la clause **SELECT** permet de rechercher dans les instances de la classe *Patient* celles qui ont comme valeur de la propriété *smoker* la valeur booléenne *True* et de renvoyer leurs noms.

2.2.3 Exploitation du méta-modèle

Comme nous l'avons précisé plus haut, *OntoQL* est basé sur un modèle d'ontologie noyau susceptible d'être modifié en utilisant le langage *OntoQL* lui-même.

Requête 6. Ajouter, dans le modèle d'ontologie (méta-modèle), le quantificateur universel *AllValuesFrom* [28] de *OWL* :

CREATE ENTITY #OWLAllValuesFrom **UNDER** #Class (
    #onProperty **REF**(#Property),
    #allValuesFrom **REF**(#Class))

Explication : Cette instruction utilisant la clause **CREATE ENTITY** permet d'ajouter l'entité *OWLAllValuesFrom* au méta-modèle noyau comme une sous-entité de *Class*. Cette nouvelle entité est créée avec deux attributs *onProperty* qui prend comme valeur les identifiants des propriétés et *allValuesFrom* qui prend comme valeur les identifiants des classes.

28. https://www.w3.org/TR/owl-ref/#allValuesFrom-def

2.3 Choix du système OntoDB/OntoQL comme infrastructure de développement

Afin de valider notre proposition, nous avons besoin d'une infrastructure qui soit capable de représenter et exploiter les ontologies, leurs modèles et leurs instances avec un langage dédié. Nous avons choisi le système OntoDB/OntoQL que nous avons présenté dans les sous-sections précédentes. *OntoDB* permet de persister les ontologies et leurs instances et *OntoQL* permet de les manipuler. Trois critères fondamentaux ont motivé notre choix de cette infrastructure pour la mise en œuvre de notre approche :

1. Le système choisi doit permettre la gestion d'un grand volume de données. En effet, avec l'émergence des nouvelles technologies de l'information et de la communication et surtout grâce au développement d'Internet et des Intranets, une masse considérable de données est actuellement disponible et doit être prise en charge par les différents systèmes de gestion de données. De ce fait notre choix s'est orienté vers les BDS qui permettent de répondre à cette exigence ;
2. L'infrastructure choisie doit pouvoir prendre en charge différents formalismes ontologiques. A l'inverse des autres BDS issues des milieux académique et industriel qui ne supportent qu'un seul modèle d'ontologie (RDFS,OWL,...), *OntoDB* diffère par le fait qu'elle est basée sur un modèle noyau qui peut prendre en charge différents constructeurs issus des différents formalismes ontologiques ;
3. Comme nous voulons étendre le modèle noyau de la base de données avec notre modèle de contexte proposé dans le chapitre précédent, l'infrastructure choisie doit permettre l'évolution de son méta-modèle. Grâce au langage *OntoQL*, l'extension du modèle d'ontologie d'*OntoDB* est facilement réalisable.

3 Persistance du contexte dans les BDS : l'exemple d'OntoDB

Dans cette section, nous détaillons comment le modèle de contexte proposé dans le chapitre précédent peut être utilisé pour étendre le modèle de la BDS *OntoDB* afin de prendre en charge le contexte.

3.1 Extension du modèle d'OntoDB avec le modèle de contexte proposé

Afin de prendre en charge le modèle du contexte dans *OntoDB*, la partie *Méta-Schéma* qui permet de stocker le modèle ontologique doit être étendue avec les constructeurs de notre modèle de contexte comme illustré dans la figure 5.3. Pour cela, nous utilisons le langage d'exploitation *OntoQL* afin de manipuler ces différents composants et étendre le modèle ontologique

d'*OntoDB* avec le contexte. Nous rappelons que le langage *OntoQL* permet de manipuler l'ontologie, ses instances ainsi que son modèle.

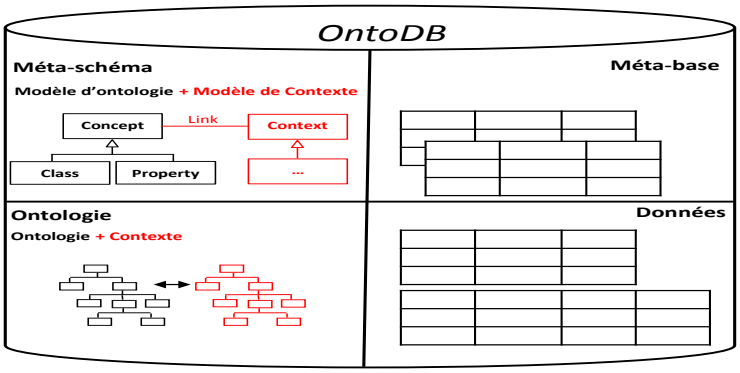


FIGURE 5.3 – Architecture d'OntoDB étendue avec le modèle du contexte

Cette extension nous amène donc à décrire un ensemble d'instructions *OntoQL* qui permettent de créer l'ensemble des éléments constituant notre modèle de contexte. Ces créations doivent être faites au niveau méta-modèle. Par exemple, la création du constructeur 'Context' au niveau du modèle d'ontologie d'*OntoDB* s'effectue en créant une nouvelle entité via la ressource 'ENTITY' du méta-modèle. Pour ceci, nous utilisons l'instruction *OntoQL* suivante :

```
CREATE ENTITY #Context(
#code INT,
#name STRING).
```

Notez que l'utilisation du symbole # signifie que la création se fait au niveau méta-modèle.

Nous présentons dans ce qui suit le processus de création de l'ensemble des constructeurs de notre modèle de contexte.

3.1.1 Création des constructeurs correspondant à la catégorie "*Contexte de définition*"

Cette catégorie de contexte, décrite dans 3.2.2, est représentée dans notre modèle de contexte par les entités suivantes : 'Definition Context', 'Context Statement', 'Global Definition'et 'Global Statement'. La table 5.1 décrit les instructions *OntoQL* qui permettent de créer l'ensemble de ces entités.

3.1.2 Création des constructeurs correspondant à la catégorie "*Contexte d'évaluation*"

La catégorie "Contexte d'évaluation" est détaillée dans 3.2.2. Comme nous l'avons vu, elle se subdivise en trois catégories.

```
CREATE ENTITY #Context_Statement(
#left_Prop REF (#Property),
#left_Concept REF (#Class),
#right_Prop REF (#Property),
#right_Concept REF (#Class),
#restriction_constructor STRING,
#restriction_value STRING,
#right_part_as_statement REF (#Context_Statement));
```

```
CREATE ENTITY #Definition_Context UNDER #Context(
#concerned_concept REF (#Class),
#has_context_statement REF (#Context_Statement));
```

```
CREATE ENTITY #Global_Statement(
#left_definition REF (#Definition_Context),
#right_definition REF (#Definition_Context),
#operator STRING,
#right_definition_as_statement REF (#Global_Statement));
```

```
CREATE ENTITY #Global_Definition(
#g_concerned_concept REF (#Class),
#has_global_statement REF (#Global_Statement));
```

TABLE 5.1 – Création des entités correspondant à la catégorie "Contexte de définition"

• **Contexte local**

Cette catégorie de contexte se divise elle même en deux sous-catégories :

1. **Fonction Mathématique :** elle est représentée dans notre modèle par les entités : 'Math Function Context', 'Math Formula', 'Context Parameter' et 'Context Condition'. La Table 5.2 illustre les instructions *OntoQL* permettant de créer l'ensemble de ces entités.
2. **Dépendance Fonctionnelle :** cette sous-catégorie est représentée dans notre modèle de contexte par les entités : 'Functional dependency Context', 'Context Parameter', 'Context Condition' et 'Context Result'. La Table 5.3 décrit les instructions *OntoQL* permettant de créer ces entités.

• **Contexte environnant**

Cette catégorie de contexte se divise à son tour en deux sous-catégories :

1. **Contexte temporel :** cette sous-catégorie est représentée dans notre modèle par l'entité 'Temporal Context'. La Table 5.4 décrit l'instruction *OntoQL* qui permet de créer cette entité.

```
CREATE ENTITY #Evaluation_Context UNDER #Context(
#concerned_property REF (#Property));
```

```
CREATE ENTITY #Local_Context UNDER #Evaluation_Context;
```

```
CREATE ENTITY #Context_Parameter (
#parameter REF (#Property) ARRAY);
```

```
CREATE ENTITY #Context_Condition (
#left_part_condition REF (#Property),
#right_part_condition REF (#Property),
#condition_operator REF (#Comp_Operator),
#condition_value STRING,
#and_condition REF (#Context_Condition),
#or_condition REF (#Context_Condition));
```

```
CREATE ENTITY #Math_Formula (
#aggregate_function REF (#Aggregate_Function),
#operator REF (#Math_Operator),
#left_operand REF (#Property),
#right_operand REF (#property),
#right_operand_Formula REF (#Math_Formula));
```

```
CREATE ENTITY  #Math_Function_Context UNDER #Local_Context (
#has_parameter_mf REF (#Context_Parameter),
#has_condition_mf REF (#Context_Condition),
#has_formula REF (#Math_Formula));
```

TABLE 5.2 – Création des entités correspondant à la catégorie “Fonction Mathématique ”

```
CREATE ENTITY #Context_Result (
#result_operator REF (#Comp_Operator),
#result_value STRING);
```

```
CREATE ENTITY  #Functional_Dependency_Context UNDER #Local_Context (
#has_parameter_fd REF (#Context_Parameter),
#has_condition_fd REF (#Context_Condition),
#has_result REF (#Context_Result));
```

TABLE 5.3 – Création des entités correspondant à la catégorie “Dépendance Fonctionnelle”

2. **Contexte spatial :** cette sous-catégorie est représentée dans notre modèle par l’entité ‘Spatial Context’. La Table 5.5 décrit l’instruction *OntoQL* qui permet de créer cette entité.

```
CREATE ENTITY #Surrounding_Context UNDER #Evaluation_Context;
```

```
CREATE ENTITY #Temporal_Context UNDER #Surrounding_Context (
#time_value STRING,
#t_contextual_value STRING);
```

TABLE 5.4 – Création de l'entité correspondant à la catégorie 'Contexte Temporel'

```
CREATE ENTITY #Spatial_Context UNDER #Surrounding_Context (
#space_value STRING)
#s_contextual_value STRING);
```

TABLE 5.5 – Création de l'entité correspondant à la catégorie 'Contexte Spatial'

• **Unités de mesure**

Cette catégorie de contexte est représentée dans notre modèle par l'entité 'Unit Of Measure Context'. La Table 5.6 décrit l'instruction *OntoQL* qui permet de créer cette entité.

```
CREATE ENTITY #Unit_Of_Measure_Context UNDER #Surrounding_Context (
#default_unit STRING,
#target_unit STRING,
#conversion_rate REAL,
#conversion_source STRING);
```

TABLE 5.6 – Création de l'entité correspondant à la catégorie 'Unité de mesure'

3.2 Mise en œuvre du lien entre le modèle de contexte et le modèle d'ontologie dans OntoDB

Le lien entre notre modèle de contexte que nous avons persisté dans *OntoDB* et le modèle d'ontologie de cette dernière est concrétisé par la création des associations :

— concerned_concept : qui associe chaque contexte de définition représenté par l'entité *Definition Context* du modèle de contexte avec le concept ontologique correspondant représenté par l'entité du méta-modèle d'OntoDB *Class* ;

— g_concerned_concept : qui associe chaque définition globale représentée par l'entité *Global Definition* du modèle de contexte avec le concept ontologique correspondant représenté par l'entité du méta-modèle d'OntoDB *Class* ;

— concerned_property : qui associe chaque contexte d'évaluation représenté par l'entité

Evaluation Context du modèle de contexte avec la propriété ontologique correspondante représentée par l'entité du méta-modèle d'OntoDB *Property*.

4 Prise en charge du contexte dans les requêtes : extension du langage OntoQL

Dans la section précédente, nous avons montré comment augmenter le modèle ontologique d'*OntoDB* avec notre modèle de contexte. Cette extension offre la possibilité aux utilisateurs de définir et de persister les contextes dans la BDD et par conséquent d'interroger cette dernière en fonction des contextes définis. Néanmoins, cette interrogation contextuelle nécessite la connaissance parfaite du modèle logique du contexte et de son lien avec le modèle ontologique.

Dans cette section, nous proposons d'étendre le langage d'exploitation *OntoQL* afin qu'il prenne en compte le contexte dans ses requêtes d'une façon transparente aux utilisateurs. Nous nous limitons ici aux contextes d'évaluation.

Le langage *OntoQL* est composé d'une partie *syntaxique* et d'une partie *sémantique*. La partie syntaxique concerne la grammaire du langage qui correspond aux différentes règles logiques et structurelles régissant la formulation des expressions *OntoQL*. La partie sémantique correspond quant à elle à l'interprétation des expressions *OntoQL* comme des opérations à faire exécuter sur la BDD. La syntaxe complète du langage *OntoQL* ainsi que l'algèbre qui définit la sémantique de ses expressions sont détaillées dans [99].

Notre extension du langage *OntoQL*, détaillée dans la suite de cette section, consiste à étendre sa grammaire et sa sémantique afin qu'il supporte formellement l'expression des contextes d'évaluation dans ses requêtes. Ceci est matérialisé par l'ajout d'un nouvel opérateur, que nous avons appelé 'USING CONTEXT', à la clause 'SELECT' du langage et le développement d'un interpréteur pour le contexte en associant une fonction d'interprétation à chaque catégorie du contexte d'évaluation décrite dans notre modèle.

4.1 Extension de la grammaire d'OntoQL : l'opérateur 'Using Context'

Le rôle du nouvel opérateur USING CONTEXT est de préciser, dans une requête OntoQL le contexte dont dépendent les données interrogées. Par conséquent, la grammaire originale de ce langage doit être modifiée.

La syntaxe générale d'une requête *OntoQL* est la suivante :

<*query specification*> ::= <*select clause*> <*from clause*> [<*where clause*>]
[<*group by clause*>] [<*having clause*>] [<*order by clause*>]
[<namespace clause>] [<language clause>]

Mise à part les clauses *Using Namespace*, qui sert à indiquer le ou les espaces de noms

utilisés dans une instruction OntoQL, et *Using Language*, qui sert à indiquer la langue naturelle utilisée, les autres clauses d'*OntoQL* sont similaires à celles de *SQL*. Ainsi, la clause *Select* précise le retour de la requête en indiquant des propriétés, des appels de fonctions ou des expressions arithmétiques. La clause *From* permet de spécifier les tables (ou classes) à interroger. La clause optionnelle *Where* permet de restreindre les n-uplets parcourus en imposant des prédicats à satisfaire. La clause optionnelle *Group By* permet de regrouper les n-uplets selon une valeur déterminée. La clause optionnelle *Having* permet de restreindre les groupes considérés en imposant des prédicats à satisfaire. Enfin, la clause optionnelle *Order By* permet de trier les résultats de la requête.

Afin d'introduire la nouvelle clause USING CONTEXT, nous modifions la syntaxe d'*OntoQL* comme suit :

<*query specification*> ::= <*select clause*> <*from clause*> [<*where clause*>]
[<*group by clause*>] [<*having clause*>]
[<*order by clause*>] [<***context clause***>]
[<namespace clause>] [<language clause>]

<***context clause***> ::= **USING CONTEXT** <***context boolean expression***>
<***context boolean expression***> ::= <***context boolean term***>
| <*context boolean expression*> OR <context boolean term>
<***context boolean term***> ::= <***context boolean factor***>
| <*context boolean term*> AND <*context boolean factor*>
<***context boolean factor***> ::= [NOT] <***contextIdentifier***>
<***contextIdentifier***> ::= *Context name*

Dans notre modèle de contexte, l'attribut *name* sert à caractériser significativement les contextes avec des descriptions textuelles et, par conséquent, il est considéré comme un identifiant : <***contextIdentifier***>.
La table 5.7 présente quelques exemples montrant les différentes utilisations possibles d'une requête générique contenant l'opérateur USING CONTEXT.

4.2 Extension de la sémantique d'OntoQL : Interprétation des requêtes

Le langage *OntoQL* est développé en Java [29] en utilisant le générateur de parseur ANTLR [30] pour effectuer les analyses lexicale et syntaxique du langage. Comme nous l'avons expliqué ci-dessus, l'extension de la grammaire d'*OntoQL* avec un nouvel opérateur implique l'extension de sa sémantique qui représente le mécanisme d'interprétation des expressions et l'exécution des opérations correspondantes sur la BDD. Notre approche consiste donc à étendre la sémantique d'*OntoQL* afin qu'il soit capable d'interpréter les expressions utilisant la nouvelle grammaire

29. http://www.java.com/fr/
30. www.antlr.org

SELECT ‘selection’ **FROM** ‘tableReference’ **USING CONTEXT** ‘contextIdentifer’
SELECT ‘selection’ **FROM** ‘tableReference’ **USING CONTEXT** ‘contextIdentifer’ **AND** ‘contextIdentifer’
SELECT ‘selection’ **FROM** ‘tableReference’ **USING CONTEXT** ‘contextIdentifer’ **OR** ‘contextIdentifer’
SELECT ‘selection’ **FROM** ‘tableReference’ **USING CONTEXT NOT** ‘contextIdentifer’

TABLE 5.7 – Requêtes OntoQL génériques utilisant l’opérateur USING CONTEXT

supportant l’opérateur USING CONTEXT.

Pour cela, nous avons développé une fonction d’interprétation pour chaque catégorie du contexte d’évaluation de telle sorte que l’interpréteur, en utilisant le <contextIdentifier> et en sollicitant les différentes entités constituant le modèle de contexte, détermine la catégorie du contexte et réécrit la requête utilisant l’opérateur USING CONTEXT en une requête classique qui prend en charge les particularités et caractéristiques de ce contexte. Quatre scénarios sont possibles :

— si la requête concerne un contexte de type “Fonction Mathématique”, l’interpréteur récupère la formule mathématique et les conditions définies pour ce contexte (en sollicitant les entités ‘Math_Formula’et ‘Context_Condition’) et réécrit la requête en en tenant compte. **Exemple** : nous prenons l’exemple de la durée moyenne de séjour (ALOS) présenté dans 2. Nous considérons la requête Q1 exprimée par un utilisateur s’intéressant à la durée moyenne de séjour des patients (ALOS) pour un contexte particulier : “les longs séjours”. En supposant que le contexte ‘Long Stay’est défini et stocké dans la BDD, la table 5.8 illustre la requête Q1 et son interprétation.

Select s.ALOS **FROM** Sojourn **AS** s **USING CONTEXT** ‘Long Stay’	=>	**SELECT** (s.TDD / s.TD) **AS** ALOS **FROM** Sojourn **AS** s, Ward **AS** w **WHERE** w.name = ‘psychiatry’

TABLE 5.8 – Interprétation de la requête Q1

— si la requête concerne un contexte de type “Dépendance Fonctionnelle”, l’interpréteur récupère les conditions définies pour ce contexte (en sollicitant l’entité ‘Context_Condition’)

et réécrit la requête en remplaçant la clause USING CONTEXT par une clause WHERE qui contient les prédicats correspondants à ces conditions.
Exemple : prenons l'exemple de la tension artérielle présenté dans 3. Nous considérons la requête Q2 qui correspond à la recherche des noms des patients ayant une tension artérielle élevée. En supposant que le contexte 'HighBP' est défini et stocké dans la base de données, la table 5.9 illustre la requête Q2 et son interprétation.

SELECT p.name, m.bloodPressure **FROM** Patient **AS** p, MedicalMonitoring **AS** m **WHERE** m.hasPatient = p.idPatient **USING CONTEXT** 'HighBP'	=>	**SELECT** p.name, m.bloodPressure **FROM** Patient **AS** p, MedicalMonitoring **AS** m **WHERE** m.hasPatient = p.idPatient m.bloodPressure > 14 AND p.smoker = True AND p.ageCategory = 'Adult'

TABLE 5.9 – Interprétation de la requête Q2

— si la requête concerne un contexte de type "Contexte Environnant", l'interpréteur réécrit la requête en une nouvelle requête SELECT qui interroge l'entité correspondante (soit 'Temporel Context' soit 'Spatial Context') pour retourner la ou les valeurs pertinentes.
Exemple : prenons l'exemple de la validité des prix des consultations médicales présenté dans 4. Nous considérons la requête Q3 qui correspond à la recherche du prix d'une consultation médicale dans l'ancienne tarification. En supposant que le contexte '2015Tariff', qui correspond à la tarification de l'année 2015, est défini et stocké dans la base de données, la table 5.10 illustre la requête Q3 et son interprétation. Pour simplifier la lecture de la requête résultante, nous utilisons la notation "RID (Individu)" pour spécifier l'identifiant RID de l'individu concerné au lieu de mettre toute la requête qui sert à obtenir cet identifiant.

SELECT m.price **FROM** MedicalProcedure **AS** m **WHERE** m.name = 'General Practice Consultation' **USING CONTEXT** '2015Tariff'	=>	**SELECT** #t.contextual_value **FROM** #Temporal_Context **AS** t **WHERE** #t.name = '2015Tariff' **AND** #t.concerned_individual = RID (General Practice Consultation)

TABLE 5.10 – Interprétation de la requête Q3

— si la requête concerne un contexte de type "Unités de Mesure", l'interpréteur, en sollicitant l'entité 'Unit_Of_Measure_Context' récupère le taux de conversion défini pour ce contexte et réécrit la requête en multipliant la valeur recherchée par ce taux.

Exemple : prenons l'exemple du prix d'une consultation médicale présenté dans le chapitre précédent 5. Nous considérons la requête Q4 qui correspond à la recherche du prix d'une consultation médicale en Dollar américain. En supposant que le contexte 'USDollars', qui correspond à la définition de la conversion entre l'unité utilisée par défaut (qui est l'Euro) et le Dollar américain, est défini et stocké dans la base de données, la table 5.11 illustre la requête Q4 et son interprétation.

SELECT m.price **AS** Price **FROM** MedicalProcedure **AS** m **WHERE** m.name = 'General Practice Consultation' **USING CONTEXT** 'USDollars'	=>	**SELECT** m.price * t.conversion_rate **AS** Price **FROM** MedicalProcedure **AS** m, #Unit_Of_Measurement_Context **AS** t **WHERE** m.name = 'General Practice Consultation' **AND** t.name = 'USDollars '

TABLE 5.11 – Interprétation de la requête Q3

5 Validation : étude de cas

Afin de valider notre approche consistant à stocker et exploiter le contexte dans les BDS, nous reconsidérons le cas d'étude 3.2.1 présenté dans le chapitre précédent. Comme nous l'avons précisé, ce cas d'étude, issu du domaine de la santé, repose sur l'ontologie de domaine *Healthcare* dont un extrait est représenté par la figure 4.5. Nous considérons le scénario suivant : notre hôpital, divisé en deux unités principales (l'unité générale des soins et l'unité spécialisée en psychiatrie), dispose d'une base de données qui enregistre les différentes informations concernant les activités médicales, et les patients et leurs séjours effectués dans l'une des deux unités de l'hôpital. Nous traitons les quatre cas suivants :

1. Cas 1 : un utilisateur (eg. un médecin) souhaite interroger la BDD pour avoir la liste des patients qui ont une tension élevée.
2. Cas 2 : un utilisateur (eg. un employé du département *Statistiques*) veut interroger la BDD pour avoir la durée moyenne des longs séjours effectués dans l'hôpital.
3. Cas 3 : un utilisateur souhaite avoir le prix d'une consultation médicale dans la tarification arrêtée en 2013.
4. Cas 4 : un utilisateur souhaite avoir le prix d'une consultation médicale en Dollars.

Après avoir créé une BDS *OntoDB*, nous présentons dans ce qui suit comment : (i) stocker et instancier l'ontologie *Healthcare*, (ii) instancier notre modèle de contexte afin de stocker les informations contextuelles et créer les liens entre chaque concept concerné et son ou ses contextes, et (iii) utiliser le langage *OntoQL* avec le nouvel opérateur USING CONTEXT afin d'interroger notre BDD en utilisant le contexte.

5.1 Persistance du fragment de l'ontologie *Healthcare*

Nous utilisons le langage *OntoQL* pour créer et stocker notre ontologie dans *OntoDB*. Par exemple, la table 5.12 illustre les instructions *OntoQL* permettant de créer les classes Medical_Procedure et Medical_Monitoring de l'ontologie *Healthcare*. Nous notons que l'instruction *OntoQL* CREATE EXTENT OF définit une extension d'une classe donnée en indiquant les propriétés utilisées pour en décrire les instances.

CREATE #CLASS Medical_Procedure (**PROPERTIES** (nameProcedure STRING, Price REAL)); **CREATE EXTENT OF** Medical_Procedure (nameProcedure, Price);
CREATE #CLASS Medical_Monitoring (**PROPERTIES** (HeartRate INT, respirationRate INT, bloodPressure INT, BPCategory Enum ('low','Normal','High'), hasPatient REF(patient))); **CREATE EXTENT OF** Medical_Monitoring (bloodPressure, BPCategory, hasPatient);

TABLE 5.12 – Création des classes *Medical_Procedure* et *Medical_Monitoring*

Afin de définir les différentes instances des classes créées, nous utilisons la clause *OntoQL* INSERT INTO. Par exemple, la table 5.13 illustre l'instruction *OntoQL* qui permet de définir une instance de la classe Medical_Procedure.

INSERT INTO Medical_Procedure (code, nameProcedure, Price) **VALUES** (10, 'Consultation',20);

TABLE 5.13 – Instanciation de la classe *Medical_Procedure*

Les instructions *OntoQL* permettant la création de l'ensemble des classes décrivant notre ontologie sont représentées dans l'annexe 2.8. Pour plus de clarté, nous utilisons, dans le reste de cette section, des figures à la place des instructions *OntoQL* pour présenter l'instanciation de notre ontologie.

5.2 Persistance et interrogation des informations contextuelles

Afin d'expliquer l'utilité de nos propositions et illustrer comment notre approche peut être utilisée pour surmonter les problèmes liés aux différentes catégories de contexte, nous utilisons les exemples suivants :

Exemple 1 : Information contextuelle de type Dépendance Fonctionnelle

La catégorie de la tension artérielle (basse, normale ou élevée) dépend de l'âge du patient, le fait qu'il fume ou pas et de la mesure de la tension elle-même. Cette connaissance experte,

qui est considérée comme un ensemble d'informations contextuelles, doit être stockée dans la BDD et reliée à la propriété correspondante qui est *BPCategory*.

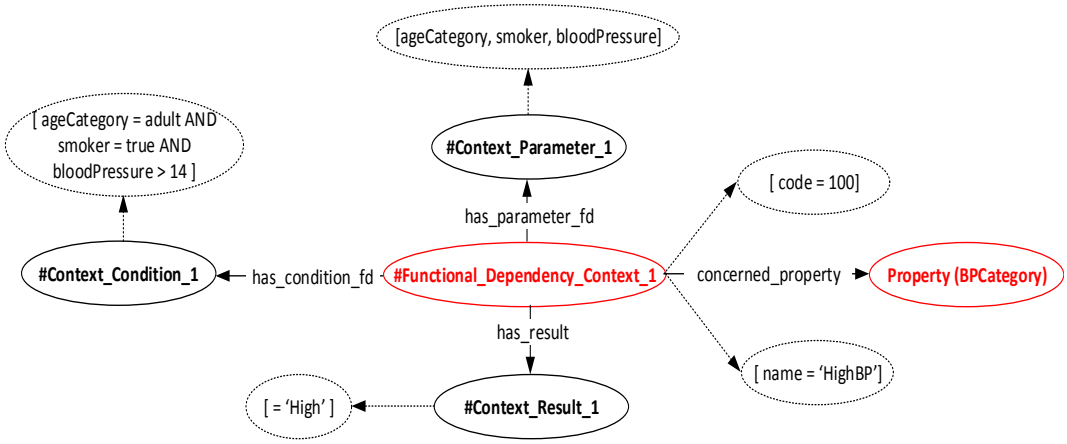


FIGURE 5.4 – Instanciation 1 : information contextuelle de type Dépendance Fonctionnelle

Nous prenons l'exemple du premier cas énoncé précédemment concernant le médecin souhaitant avoir les noms des patients qui ont eu des tensions élevées. La figure 5.4 illustre l'instanciation du modèle de contexte (déjà stocké dans *OntoDB*) pour définir le contexte 'HighBP' qui correspond à la connaissance experte caractérisant la tension artérielle élevée, et pour associer ce contexte à la propriété *BPCategory*. (les instructions *OntoQL* correspondantes sont décrites en entier dans l'annexe 2.8).

SELECT p.name, m.bloodPressure **FROM** Patient **AS** p, MedicalMonitoring **AS** m **WHERE** m.hasPatient = p.rid **USING CONTEXT** 'HighBP'	=>	**SELECT** p.name, m.bloodPressure **FROM** Patient **AS** p, MedicalMonitoring **AS** m **WHERE** m.hasPatient = p.rid AND m.bloodPressure > 14 AND p.smoker = True AND p.ageCategory = 'Adult'

TABLE 5.14 – Requête Q_{cas1} et son interprétation

Une fois l'information contextuelle stockée dans la BDD, nous pouvons utiliser l'opérateur USING CONTEXT afin de formuler une requête *OntoQL* que nous appelons Q_{cas1}. Elle spécifie le contexte du médecin qui s'intéresse aux patients ayant une tension élevée et donc correspond parfaitement à son besoin. La table 5.14 illustre la requête utilisée ainsi que son interprétation.

Pour tester nos requêtes, nous utilisons l'outil *OntoQLPlus* qui est un éditeur en ligne de commandes permettant d'exécuter des requêtes *OntoQL* sur *OntoDB*.

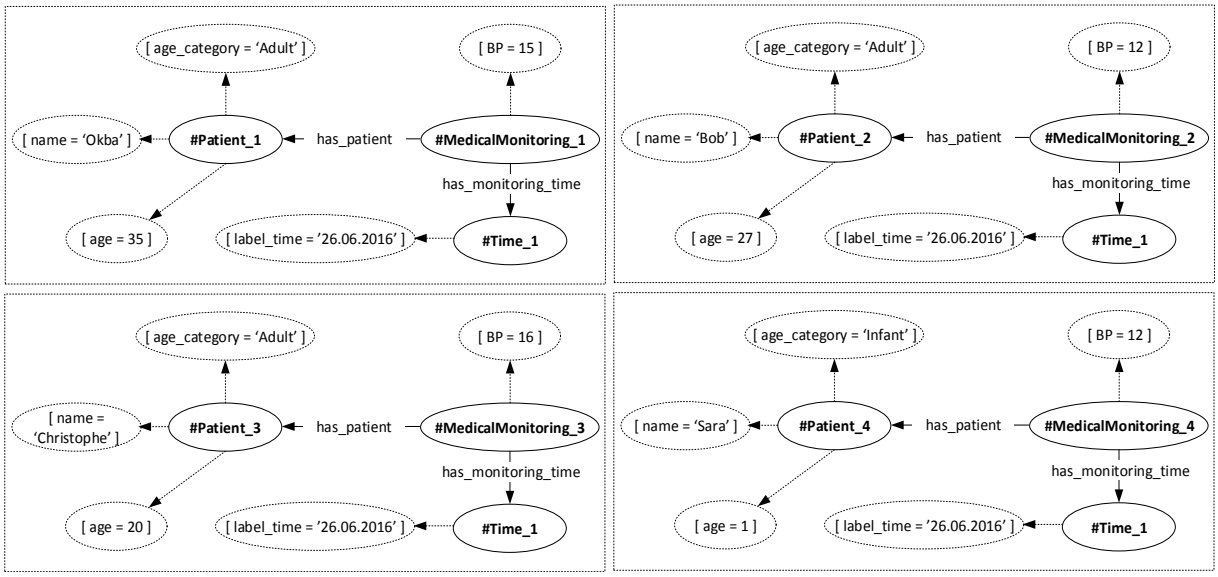
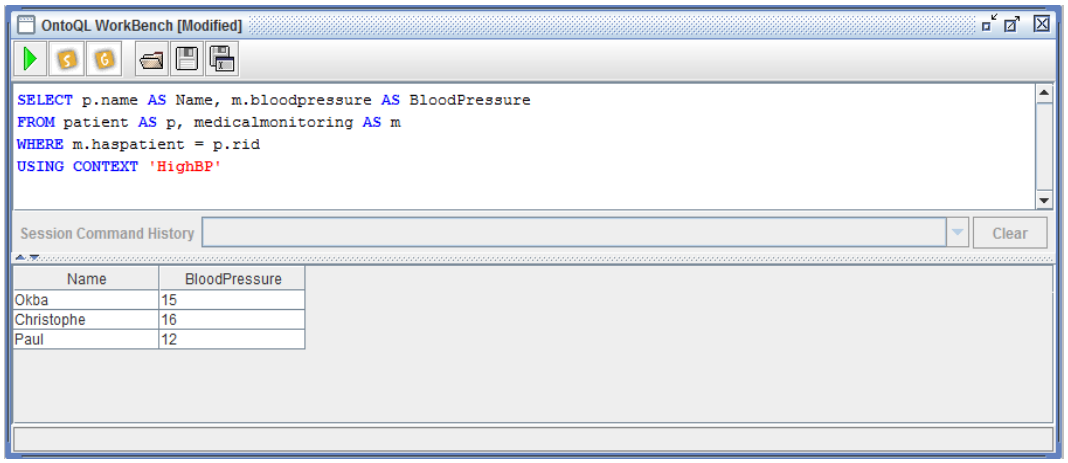


FIGURE 5.5 – Premier exemple d'instanciation de l'ontologie *Healthcare*



La requête Q_{cas1} consiste en la recherche des patients ayant des tensions artérielles élevées. L'interpréteur utilise l'information contextuelle définie dans la figure 5.4, que nous avons stockée dans notre base de données, afin de retourner des résultats pertinents comme le montre la capture d'écran ci-dessus.

TABLE 5.15 – Exécution de la requête Q_{cas1}

Sachant que notre BDD est peuplée suivant l'exemple présenté dans la figure 5.5, l'exécution de la requête Q_{cas1}, qui correspond à la recherche des patients avec des *tensions élevées*, donne le résultat commenté est illustré dans la table 5.15.

Exemple 2 : Information contextuelle de type Fonction Mathématique

La durée moyenne de séjour des patients (ALOS) dépend du contexte. En effet, la formule utilisée pour son calcul varie selon la nature du séjour :

1. *Contexte 1* : pour les longs séjours, la formule utilisée est la suivante : « Average Length Of Stay = Total Discharge Days / Total Discharge » ;
2. *Contexte 2 :* pour les courts séjours, la formule adéquate est la suivante : « Average Length Of Stay = Total Inpatient Days Of Care / Total Admissions ».

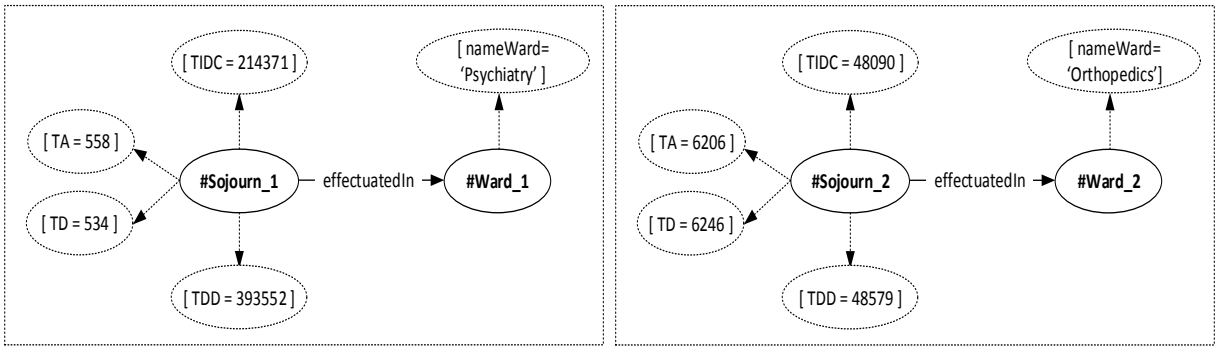


FIGURE 5.6 – Deuxième exemple d'instanciation de l'ontologie *Healthcare*

Nous expliquons comment notre approche, en utilisant ces informations contextuelles, permet de répondre parfaitement au besoin de l'utilisateur du deuxième cas qui s'intéresse à l'indicateur *ALOS* des patients qui ont effectué de longs séjours à l'hôpital. Pour cela, et afin de pouvoir commenter les résultats fournis par la requête, nous peuplons d'abord notre BDS suivant l'exemple illustré par la figure 5.6.

Ensuite, nous stockons les deux contextes 'LongStay'et 'ShortStay'qui correspondent à la connaissance experte déterminant les formules mathématiques adéquates devant être utilisées pour calculer la durée moyenne de séjours des patients respectivement dans le cas d'un court et d'un long séjour. La figure 5.7 illustre l'instanciation de notre modèle de contexte permettant de définir ces deux contextes (les instructions *OntoQL* correspondantes sont décrites en détail dans l'annexe 2.8).

SELECT averageLengthOfStay **AS** ALOS **FROM** Sojourn **USING CONTEXT** 'LongStay'	=>	**SELECT** (s.totalDischargeDays / s.totalDischarge) **AS** ALOS From Sojourn **AS** s, Ward **AS** w **WHERE** s.effectuatedIn = w.rid and w.nameWard = 'Psychiatry'

TABLE 5.16 – Requête Q_{cas2} et son interprétation

Une fois ces deux contextes stockés dans la BD, les utilisateurs peuvent utiliser l'opérateur USING CONTEXT afin de spécifier dans leurs requêtes le contexte visé. Dans ce cas-là, l'utili-

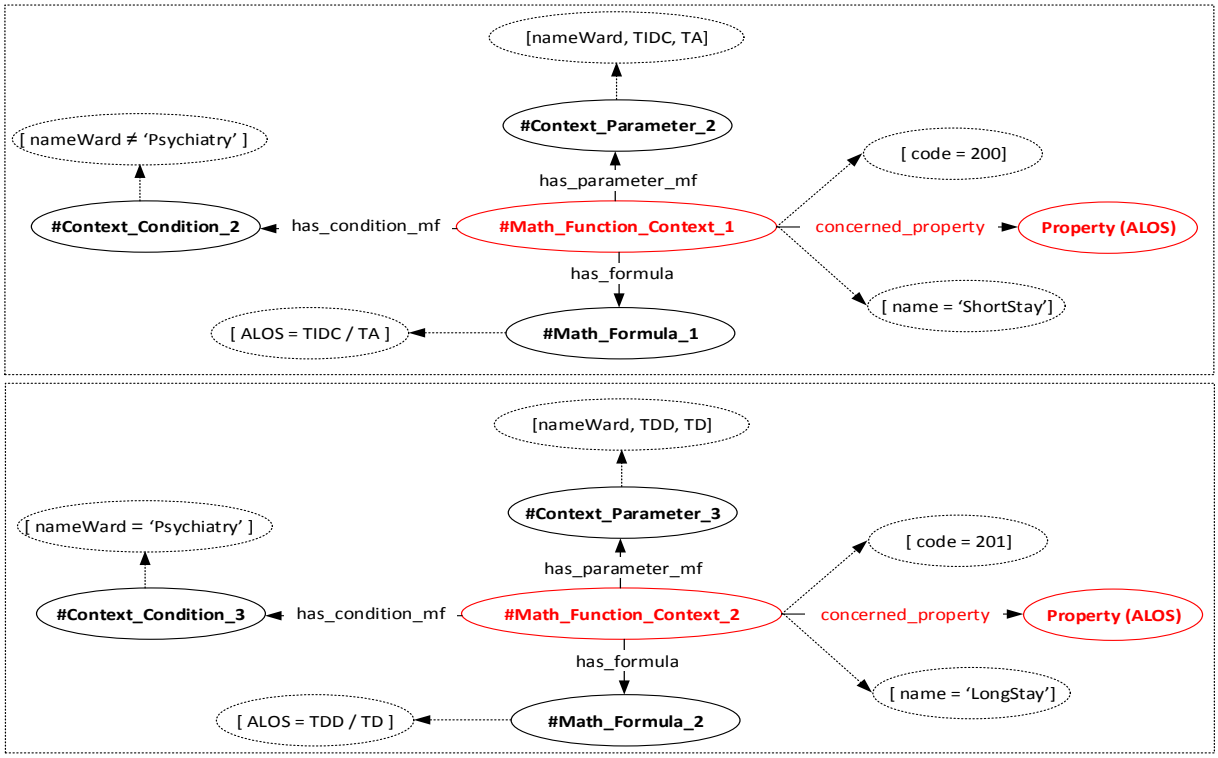


FIGURE 5.7 – Instanciation 2 : information contextuelle de type Fonction Mathématique

sateur du deuxième cas qui s'intéresse à la durée moyenne de séjours des patients ayant effectué de longs séjours au sein de l'hôpital peut formuler la requête Q_{cas2} illustrée dans la table 5.16qui répond parfaitement à son besoin. Le résultat commenté de cette requête est illustré dans la table 5.17.

Exemple 3 : Information contextuelle de type Contexte Temporel

Nous prenons dans cet exemple le cas de l'utilisateur qui, pour des besoins analytiques, souhaite avoir le prix d'une consultation médicale dans une ancienne tarification (par exemple la tarification 2013).

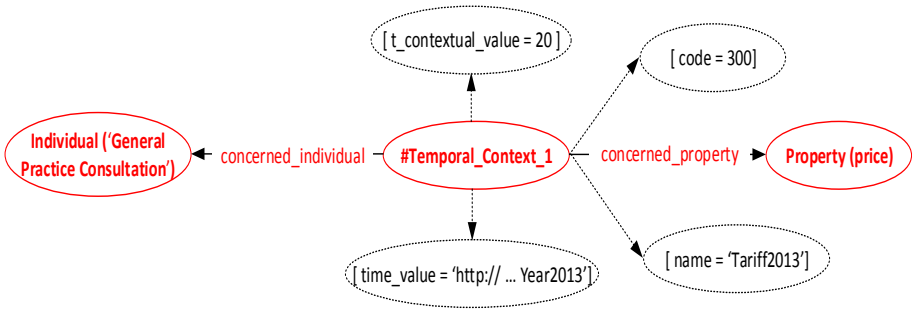
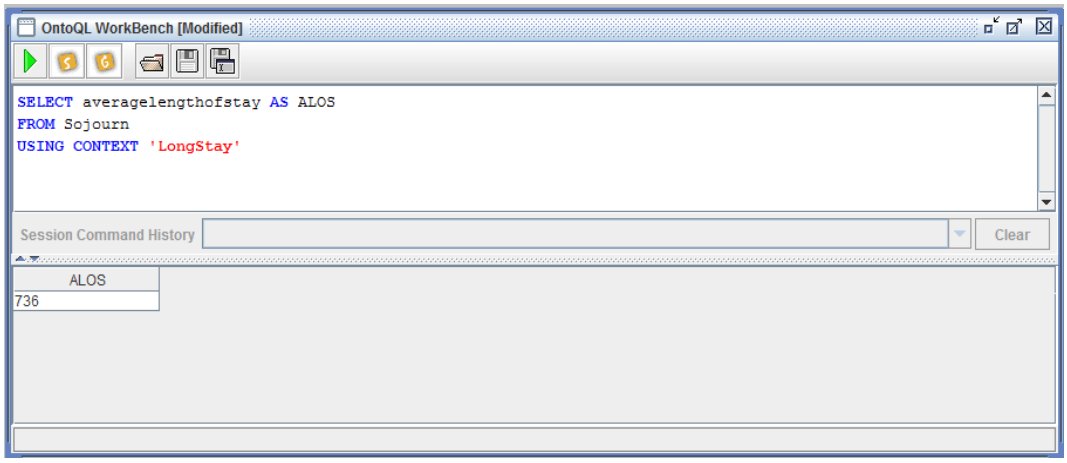


FIGURE 5.8 – Instanciation 3 : information contextuelle de type Contexte temporel

En effet, le prix peut varier avec le temps et donc nous parlons dans ce cas-là d'un contexte



La requête Q_{cas2} consiste à rechercher, pour les longs séjours effectué à l'hôpital, la valeur de l'indicateur *ALOS* correspondant à la durée moyenne de séjours des patients. L'interpréteur, en utilisant les informations définies par le contexte *LongStay*, réécrit la requête en considérant la bonne formule de calcul (*ALOS = TDD / TD*). L'exécution de cette requête renvoie la valeur 736.

TABLE 5.17 – Exécution de la requête Q_{cas2}

temporel. Grâce à notre modèle de contexte que nous avons stocké dans *OntoDB*, nous pouvons définir l'information contextuelle fixant le prix de la consultation médicale définie par la tarification arrêtée en 2013. La figure 5.8 illustre l'instanciation du modèle de contexte permettant de définir cette information contextuelle caractérisée par l'appellation 'tariff2013'et son association à la propriété correspondante 'Price' (les instructions *OntoQL* correspondantes sont décrites en détail dans l'annexe 2.8).

SELECT m.price **FROM** MedicalProcedure **AS** m **WHERE** m.name = 'General Practice Consultation' **USING CONTEXT** 'Tariff2013'	=>	**SELECT** #property_value_time_context **FROM** #Temporal_Context **WHERE** #context_attributes = (**SELECT** #rid **FROM** #Context_Uri **WHERE** #name = 'Tariff2013') AND #t_individual = (**SELECT** rid **FROM** MedicalProcedure **WHERE** name = 'General Practice Consultation')

TABLE 5.18 – Requête Q_{cas3} et son interprétation

Une fois ce contexte stocké dans notre base de données, l'utilisateur peut le spécifier dans une requête *OntoQL*, en utilisant l'opérateur USING CONTEXT, afin d'avoir le prix recherché

dans la tarification 2013. La table 5.18 illustre la requête Q_{cas3} correspondante ainsi que son interprétation.

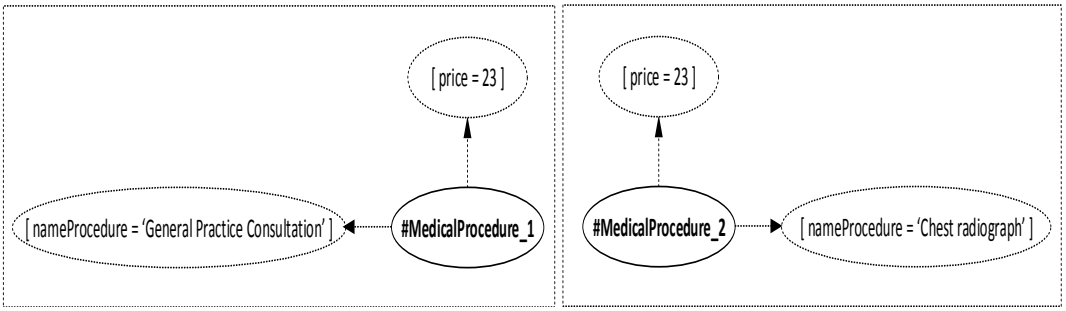
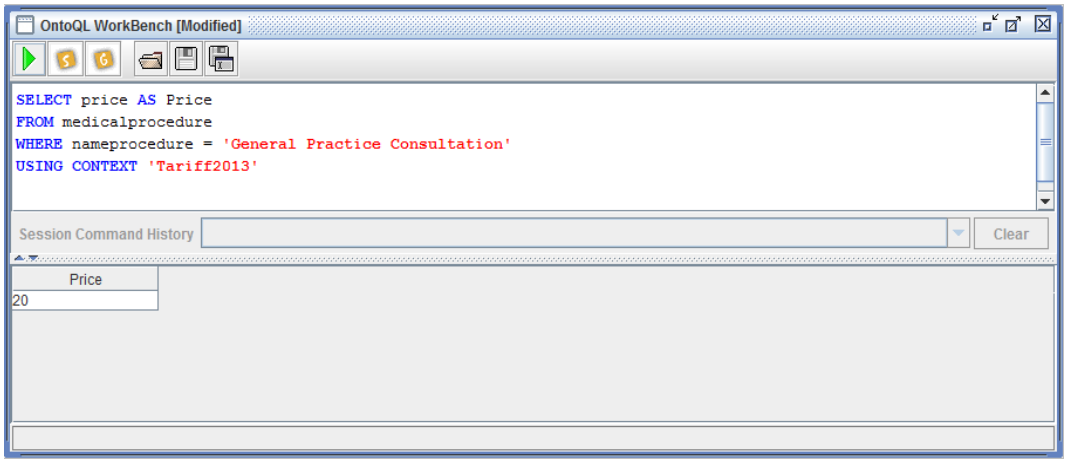


FIGURE 5.9 – Troisième exemple d'instanciation de l'ontologie *Healthcare*

Afin de tester la requête Q_{cas3}, nous peuplons notre BDS conformément à l'exemple d'instanciation de l'ontologie *Healthcare* illustré dans la figure 5.9. Le résultat obtenu est présenté et commenté dans la table 5.19.



La requête Q_{cas3} consiste à rechercher le prix de la *consultation médicale* selon la tarification en vigueur en 2013. L'interpréteur utilise l'information contextuelle définie dans la figure 5.8, que nous avons stockée dans notre base de données, afin de retourner la valeur pertinente (20 au lieu de 23) comme le montre la capture d'écran ci-dessus.

TABLE 5.19 – Exécution de la requête Q_{cas3}

Exemple 4 : Information contextuelle de type Unité de Mesure

Notre dernier exemple traite le cas de l'information contextuelle de type "Unités de Mesure". Nous supposons qu'un utilisateur souhaite avoir le prix de la radiographie du thorax en Dollar américain. En général, les bases de données (comme *OntoDB*) ne prennent pas en charge les problèmes liés aux unités de mesure et/ou unités monétaires. Nous expliquons, dans

cet exemple, comment notre approche permet de résoudre ces problème et ainsi répondre au besoin de notre utilisateur.

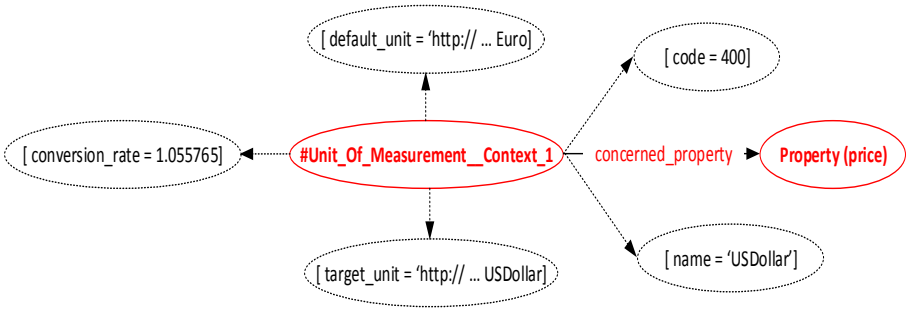


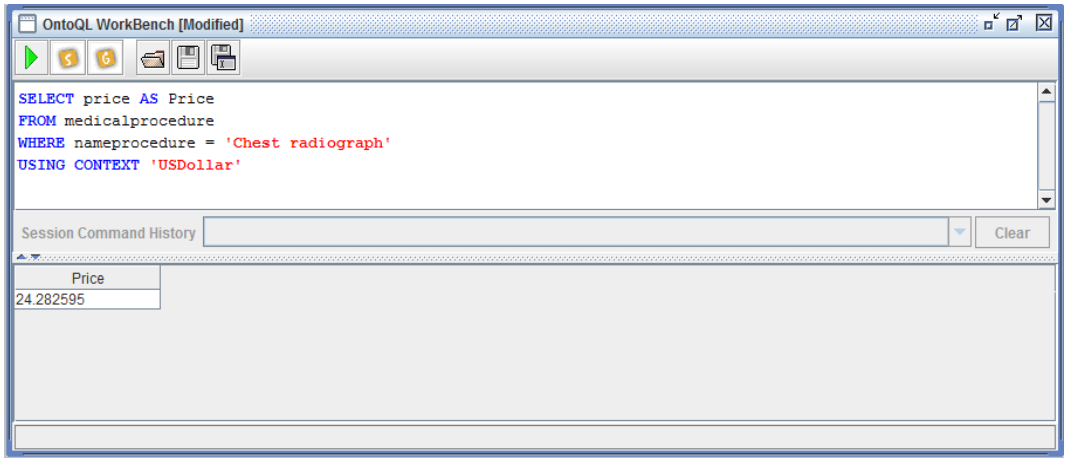
FIGURE 5.10 – Instanciation 4 : information contextuelle de type Unités de Mesure

Grâce à notre modèle de contexte persisté dans *OntoDB*, nous avons la possibilité de définir les informations contextuelles qui consistent à spécifier la monnaie utilisée par défaut pour évaluer les prix des différents actes médicaux ainsi que leurs conversions vers d'autres monnaies. Dans notre cas, nous définissons le contexte 'USDollar' qui consiste à fixer l'Euro comme monnaie utilisée par défaut et définir sa conversion vers le Dollar américain. La figure 5.10 illustre l'instanciation du modèle de contexte permettant de définir cette information contextuelle et son association à la propriété 'Price' (les instructions *OntoQL* correspondantes sont décrites en détail dans l'annexe 2.8).

SELECT m.price **FROM** MedicalProcedure **AS** m **WHERE** m.name = 'Chest radiograph' **USING CONTEXT** 'USDollar'	=>	**SELECT** (m.price ∗ u.conversion_rate) **AS** Price **FROM** MedicalProcedure **AS** m, #Unit_Of_Measure_Context **AS** u **WHERE** m.nameProcedure = 'Chest radiograph'and u.context_Attributes = (**SELECT** c.rid **FROM** #Context_Uri **AS** c **WHERE** c.name = 'USDollar')

TABLE 5.20 – Requête Q_{cas4} et son interprétation

Une fois ce contexte stocké dans notre base de données, l'utilisateur peut le spécifier en utilisant la requête *OntoQL* illustrée par la table 5.20 afin d'avoir la valeur correspondant à son contexte (le dollar américain). Afin de tester la requête Q_{cas4}, nous utilisons l'exemple d'instanciation de l'ontologie *Healthcare* du cas précédent (voir la figure 5.9). Le résultat obtenu est présenté et commenté dans la table 5.21.



La requête Q_{cas4} consiste à rechercher le prix de la *radiographie du thorax* dans une monnaie bien précise qui est le Dollar américain. L'interpréteur utilise le contexte 'USDollar' que nous avons défini et stocké dans notre base de données (Figure 5.10) afin de calculer et renvoyer la bonne valeur comme le montre la capture d'écran ci-dessus. En effet, la valeur 24.282595 correspond à la multiplication de la valeur 23 par la valeur 1.055765 qui correspondent respectivement à la valeur de la propriété *price* pour l'instance *Chest radiograph* de la classe *MedicalProcedure* et la valeur de la propriété *conversion_rate* pour l'instance *USDollar* de la classe *Unit_Of_Measure_Context*.

TABLE 5.21 – Exécution de la requête Q_{cas4}

6 Conclusion

Nous avons proposé dans ce chapitre une approche pour la contextualisation d'une base de données sémantiques afin qu'elle puisse gérer et expliciter les informations contextuelles dont peuvent dépendre ses données.

En prenant l'exemple du système *OntoDB/OntoQL*, nous avons d'abord intégré, dans le modèle d'ontologies d'*OntoDB*, le modèle de contexte proposé dans le chapitre précédent. Concrètement, ceci a été réalisé en créant, au niveau méta-modèle, l'ensemble des éléments constituant notre modèle de contexte et en mettant en œvre de son lien avec le modèle d'ontologies d'*OntoDB*. De cette manière, nous avons offert la possibilité de stocker n'importe quelle information contextuelle et de l'associer avec la ressource ontologique correspondante. Nous rappelons que cette approche est valable sur n'importe quelle base de données sémantique permettant la manipulation de son modèle d'ontologies.

Ensuite, nous avons étendu la grammaire et la syntaxe du langage *OntoQL* afin de proposer un nouvel opérateur dédié aux interrogations contextuelles. Nous avons appelé cet opérateur USING CONTEXT.

Enfin, nous avons validé notre approche de contextualisation des BDS en présentant les résultats de quelques expérimentations réalisées sur un cas d'étude issu du domaine médical.

Le prochain chapitre portera sur notre dernière contribution qui consiste à proposer une approche pour la prise en charge du contexte dans la conception des systèmes d'entreposage de données sémantiques.

Chapitre 6

Projection du contexte sur les phases conceptuelles de l'Entrepôt de Données Sémantique

Sommaire

1 Introduction 128
2 Nos hypothèses 128
3 Formalisation des entrées de notre approche 129
3.1 Ontologies contextuelles 129
3.2 Besoins des utilisateurs 131
3.3 Canevas d'intégration de sources de données sémantiques contextuelles 132
3.3.1 Schéma global $\mathcal{G}$ 132
3.3.2 Sources $\mathcal{S}$ 133
3.3.3 Mappings $\mathcal{M}$ 134
4 Notre approche de conception des EDS Contextuels 135
4.1 Modélisation Conceptuelle 135
4.1.1 Définition de l'OED 135
4.1.2 L'annotation multidimensionnelle et contextuelle de l'OED 136
4.2 Modélisation Logique 138
4.3 Processus ETL 138
4.3.1 Opérateurs ETL 138
4.3.2 Détails de l'algorithme ETL 144
4.4 Modélisation physique 147
5 Validation : expérimentation et prototype 149
5.1 Présentation de notre prototype : CONTIC-DW Design Tool . . . 149
5.1.1 Environnement de développement 149
5.1.2 Architecture fonctionnelle de CONTIC-DW Design Tool 149
5.2 Présentation de notre expérimentation 150

5.2.1 Scénario d'expérimentation 150
6 Conclusion 157

1 Introduction

Dans cette thèse, nous avons d'abord proposé, en utilisant une approche basée sur les ontologies, un modèle de contexte qui capitalise les modèles existants dans la littérature. Ensuite, dans le but d'illustrer l'utilité de ce modèle, nous avons montré comment les systèmes des BDS peuvent être étendus avec ce modèle de contexte afin de permettre une gestion effective et non ambiguë de leurs données.

Toujours dans le but de prouver la pertinence de notre proposition, nous proposons, dans ce chapitre, d'appliquer notre modèle de contexte à une dimension très importante de la gestion des données qui est *l'analyse*. Ainsi, nous proposons de projeter notre modèle de contexte sur les différentes phases de construction des systèmes des EDS. Ce choix est motivé par le fait que les EDS d'aujourd'hui sont confrontés à un nouveau challenge où les ontologies considérées deviennent de plus en plus contextuelles. Nous jugeons donc utile de revisiter le cycle de la construction des EDS en considérant et en intégrant cette nouvelle vision contextuelle des ontologies dans ce cycle de conception des EDS.

Nous montrons, dans ce chapitre, comment les différentes représentations des concepts et des propriétés dans les ontologies contextuelles impactent les différentes phases de conception des EDS, principalement les modélisations conceptuelle et logique, le processus ETL et la modélisation physique.

Ce chapitre est organisé comme suit : la section 2 énonce les hypothèses nécessaires à la faisabilité de notre approche de conception des EDS Contextuels. La section 3 présente la formalisation des entrées principales sur lesquelles repose notre approche, à savoir : les ontologies contextuelles, les besoins des utilisateurs et le canevas d'intégration de sources de données sémantiques contextuelles. La section 4 présente en détail les différentes étapes de notre approche. La section 5 valide notre approche de conception en déroulant une expérimentation sur un cas d'étude et en présentant un prototype d'outil d'aide à la conception des EDS Contextuels. La section 6 conclut ce chapitre.

2 Nos hypothèses

Trois hypothèses majeures sont considérées pour l'élaboration de notre approche :

1. **Hypothèse 1 :** nous supposons l'existence et la disponibilité d'une ontologie de domaine *partagée* (OP) entre les sources de données. Ces dernières s'engagent sur l'utilisation des concepts ontologiques définis par cette ontologie.
2. **Hypothèse 2 :** nous supposons que l'OP est augmentée par le modèle de contexte proposé. Cela veut dire que cette ontologie permet de définir n'importe quelle information contextuelle conformément à notre modèle de contexte.
3. **Hypothèse 3 :** nous supposons que chaque source référence l'OP en utilisant un ensemble

de mappings. La découverte des mappings peut être réalisée en utilisant différents scénarios :

— *a priori* au cours du processus de conception, où les correspondances entre l'OP et les ontologies locales sont définies a priori au moment de la conception des sources. Dans ce cas, les mappings sont déjà disponibles. Les concepteurs acceptent de faire des efforts lors de la conception des sources afin de faciliter le *processus ETL* au cas où les données devraient être intégrées.

— *a posteriori*, soit manuellement, soit automatiquement à l'aide d'algorithmes d'exploration. Cette problématique fait partie du domaine du Matching/Alignement des ontologies et ontologies contextuelles qui fait l'objet de plusieurs travaux [65, 64] et est hors de la portée de notre étude. Une fois que les mappings ont été découverts, le processus ressemble au premier scénario.

3 Formalisation des entrées de notre approche

Dans cette section, nous détaillons les trois entrées principales sur lesquelles repose notre proposition, à savoir : les ontologies contextuelles, les besoins des utilisateurs et le canevas d'intégration de sources de données sémantiques contextuelles.

3.1 Ontologies contextuelles

Les ontologies de domaine peuvent être formalisées suivant différents langages ontologiques tels que : *PLIB, RDF, RDFS, OIL, DAML* et *OWL*.

Nous présentons ici une formalisation générique du modèle de l'ontologie basée sur le formalisme des logiques de description (LD). Ce dernier définit des logiques pour représenter des connaissances structurelles et pour raisonner sur ces connaissances [7]. Formellement, une ontologie peut être définie comme suit [17] :

Formalisation 1

Les Ontologies de domaine :
$O : < C, \mathcal{P}, \mathcal{R}ef(C), \mathcal{R}ef'(P), \mathcal{F}ormalism >$, où :

— C : représente les classes de l'ontologie.

— $\mathcal{P}$: représente les propriétés de l'ontologie qui sont utilisées pour décrire les instances des classes C.

— $\mathcal{R}ef : C \rightarrow (Operator, \mathcal{E}xp(C,P))$ est une fonction qui associe à chaque classe un opérateur (inclusion ou équivalence) et une expression $\mathcal{E}xp$ sur d'autres classes et des propriétés. Les opérateurs utilisés dans les expressions sont : (i) les opérateurs en-

semblistes (intersectionOf (∩), unionOf (∪), complementOf (¬)), (ii) les restrictions de propriétés (AllValuesFrom ($\forall p.C$), SomeValuesFrom ($\exists p.C$), HasValue ($\ni p.C$) et (iii) les opérateurs de cardinalité ($\equiv np.C, \leq np.C, \geq np.C$). L'expression $\mathcal{E}xp$ peut être la fonction d'identité qui associe à une classe la même classe.

— *$\mathcal{R}ef' : \mathcal{P} \rightarrow C \times C$: est une fonction qui permet d'associer à chaque propriété son domaine et son rang : $\mathcal{R}ef'(p) = (\mathcal{D}om(p), \mathcal{R}ang(p))$.*

— *$\mathcal{F}ormalism$: représente le formalisme du modèle ontologique adopté : RDF, OWL, PLIB, etc.*

Afin de prendre en charge le contexte, nous avons étendu cette formalisation comme suit :

Formalisation 2

Les Ontologies Contextuelles :

$OC : < C, \mathcal{P}, \mathcal{R}ef(C), \mathcal{R}ef'(P), \mathcal{D}efContext, \mathcal{R}ef_{Def}(C), \mathcal{E}valContext, \mathcal{R}ef_{Eval}(P), \mathcal{F}ormalism >$, où :

— *$\mathcal{D}efContext = \{defcontext_1, defcontext_2, ..., defcontext_n\}$ est l'ensemble des contextes de définition tel que : $\mathcal{D}efContext : < Operator, \mathcal{E}xp(C,P) >$; $Operator$ représente un opérateur (inclusion ou équivalence) et $\mathcal{E}xp$ représente une expression sur d'autres classes et propriétés.*

— *$\mathcal{R}ef_{Def}(C) : C \rightarrow 2^{\mathcal{D}efContext}$ est la fonction qui associe à chaque concept contextuel ses contextes de définition correspondants.*

— *$\mathcal{E}valContext = \{evalcontext_1, evalcontext_2, ..., evalcontext_n\}$ est l'ensemble des contextes d'évaluation qui peuvent être de trois types :*

1. *$\mathcal{L}ocal\ Context$: ce type comporte deux sous-types :*
   (a) *$\mathcal{M}athematical\ Function\ Context : < \mathcal{P}rm, Cdt, \mathcal{F} >$, où :*
      — *$\mathcal{P}rm = \{prm_1, prm_2, ..., prm_n\}$ représente l'ensemble des paramètres de contexte ;*
      — *$Cdt = \{cdt_1, cdt_2, ..., cdt_n\}$ représente l'ensemble des conditions qui doivent être satisfaites pour le contexte en question ;*
      — *$\mathcal{F}$ est la fonction mathématique utilisée pour calculer la propriété contextualisée concernée.*
   (b) *$\mathcal{F}unctionnal\ Dependency\ Context : < \mathcal{P}rm, Cdt, \mathcal{R}slt >$, où :*
      — *$\mathcal{P}rm = \{prm_1, prm_2, ..., prm_n\}$ est l'ensemble des paramètres de contexte ;*
      — *$Cdt = \{cdt_1, cdt_2, ..., cdt_n\}$ est l'ensemble des conditions qui doivent être satisfaites pour le contexte en question ;*
      — *$\mathcal{R}slt$ est le résultat représenté par la partie droite de la dépendance fonctionnelle.*
2. *$\mathcal{S}urrounding\ Context$: ce type comporte deux sous-types :*

(a) $\mathcal{T}$emporal Context: $< \mathcal{I}, \mathcal{T}, \mathcal{V} >$, où :

— $\mathcal{I}$ est l'individu (concerné par le contexte) de la classe domaine de la propriété contextualisée ;

— $\mathcal{T}$ est le paramètre de contexte *Temps* ;

— $\mathcal{V}$ est la valeur pertinente de la propriété contextualisée.

(b) $\mathcal{S}$pace Context : $< \mathcal{I}, \mathcal{S}, \mathcal{V} >$, où :

— $\mathcal{I}$ est l'instance (concernée par le contexte) de la classe à laquelle appartient la propriété contextualisée ;

— $\mathcal{S}$ est le paramètre de contexte *Space* qui représente la localisation ;

— $\mathcal{V}$ est la valeur pertinente prise par la propriété contextualisée.

3. $\mathcal{U}$nit Of Measure Context : $< \mathcal{D}$efaultUnit, $\mathcal{T}$argetUnit, $\mathcal{R}$ate $>$, où :

— $\mathcal{D}$efaultUnit est l'unité de mesure (ou l'unité monétaire) utilisée par défaut pour évaluer la propriété contextualisée ;

— $\mathcal{T}$argetUnit est l'unité de mesure (ou l'unité monétaire) secondaire qui peut être utilisée pour évaluer la propriété contextualisée ;

— $\mathcal{R}$ate est le taux de conversion entre les deux unités précédentes.

— $\mathcal{R}ef_{Eval}(P) : \mathcal{P} \rightarrow 2^{\mathcal{E}valContext}$ est la fonction qui associe à chaque propriété contextualisée ses contextes d'évaluation correspondants.

3.2 Besoins des utilisateurs

Les besoins des utilisateurs sont essentiels afin d'identifier les données pertinentes à matérialiser dans l'ED. Par conséquent, ils jouent un rôle crucial dans notre méthode.

Pour la définition des besoins des utilisateurs, nous avons utilisé le modèle des besoins défini au sein de notre laboratoire [60]. Celui-ci est un modèle générique qui couvre les trois modèles de besoins les plus répandus, à savoir : le modèle *orienté buts* [23], le modèle des *cas d'utilisation* [94] et le *modèle conceptuel de traitements* (MCT) [154].

La figure 6.1 présente notre modèle pivot et sa connexion avec le modèle d'ontologie. L'entité principale du modèle est *Requirement*. Elle est décrite par certaines caractéristiques : *nom*, *description*, *priorité* et *type*. Un besoin est caractérisé par trois entités liées à l'entité *Requirement* par des relations d'agrégation : *Task*, *Result* et *Criterion*. Elles sont appelées les coordonnées du besoin. Chaque Besoin est caractérisé par la réalisation (influencée par des *critères*) de certaines *tâches* qui permettent d'avoir un ou plusieurs *résultats*. Il existe deux types de besoins : *fonctionnels* qui décrivent des services fournis et *non fonctionnels* qui déterminent la qualité des services (tels que la sécurité, les performances, la flexibilité, etc). Nous distinguons

trois types de relations (entre les besoins) :

1. *les relations de contributions* : dans ce type de relation, un besoin contribue partiellement à la satisfaction d'un autre besoin. Cette contribution peut être positive ou négative ;
2. *les relations de décomposition ou de raffinement* : un besoin général peut être décomposé en sous-besoins en utilisant les relations (et / ou).

Notre modèle de besoins est formellement défini comme suit :

Formalisation 3

Le modèle des besoins

$\mathcal{R}equirement_{model}$: < $\mathcal{A}$ctor, $\mathcal{R}$equirement, $\mathcal{R}$elationships >, où :

- $\mathcal{A}$ctor : {$actor_1$,$actor_2$,...,$actor_n$}: est l'ensemble des acteurs interagissant avec le système pour répondre au besoin.
- $\mathcal{R}$equirement : {$requirement_1$,$requirement_2$,...,$requirement_n$} : est l'ensemble des besoins. Un besoin est défini comme suit : $\mathcal{R}$equirement : < $\mathcal{T}$, $\mathcal{R}$, $\mathcal{C}$ >, où :
  - $\mathcal{T}$ représente les tâches à effectuer pour satisfaire les besoins. Une tâche est définie comme : $\mathcal{T}$: < $\mathcal{S}$ubject, $\mathcal{A}$ction, $\mathcal{O}$bject > où $\mathcal{S}$ubject est l'acteur qui utilise le besoin pour achever le résultat, $\mathcal{A}$ction est l'action qui doit être réalisée et $\mathcal{O}$bject est le concept (mental ou physique) concerné par le besoin.
  - $\mathcal{R}$: {r_1,r_2,...,r_n} est l'ensemble des résultats achevés pour un besoin.
  - $\mathcal{C}$: {c_1,c_2,...,c_n} est l'ensemble des critères avec lesquelles les résultats sont quantifiés.
- $\mathcal{R}$elationships : {$relation_1$,$relation_2$, ...,$relation_n$} : est l'ensemble des relations entre les besoins : (AND/OR decomposition, Contains, Requires, Refines, Conflicts and Equivalence).

3.3 Canevas d'intégration de sources de données sémantiques contextuelles

D'une manière générale, un système d'intégration de données est formellement défini par le triplet : < $\mathcal{G}$, $\mathcal{S}$, $\mathcal{M}$ > où $\mathcal{G}$ représente le schéma global, $\mathcal{S}$ représente les sources de données et $\mathcal{M}$ représente l'ensemble des correspondances (les mappings) entre $\mathcal{G}$ et $\mathcal{S}$. Afin de prendre en charge l'aspect contextuel des données dans le processus d'intégration, nous définissons le canevas dont les éléments sont formalisés comme suit :

3.3.1 Schéma global $\mathcal{G}$

Le schéma global $\mathcal{G}$ est représenté par l'ontologie partagée par les sources, appelée aussi Ontologie Globale OG. Le schéma global $\mathcal{G}$ reprend donc la formalisation du modèle ontologique proposée ci-dessus : < $\mathcal{C}$, $\mathcal{P}$, $\mathcal{R}$ef(C), $\mathcal{R}$ef'(P), $\mathcal{R}ef_{Def}$(C), $\mathcal{R}ef_{Eval}$(P), $\mathcal{F}$ormalism > 2.

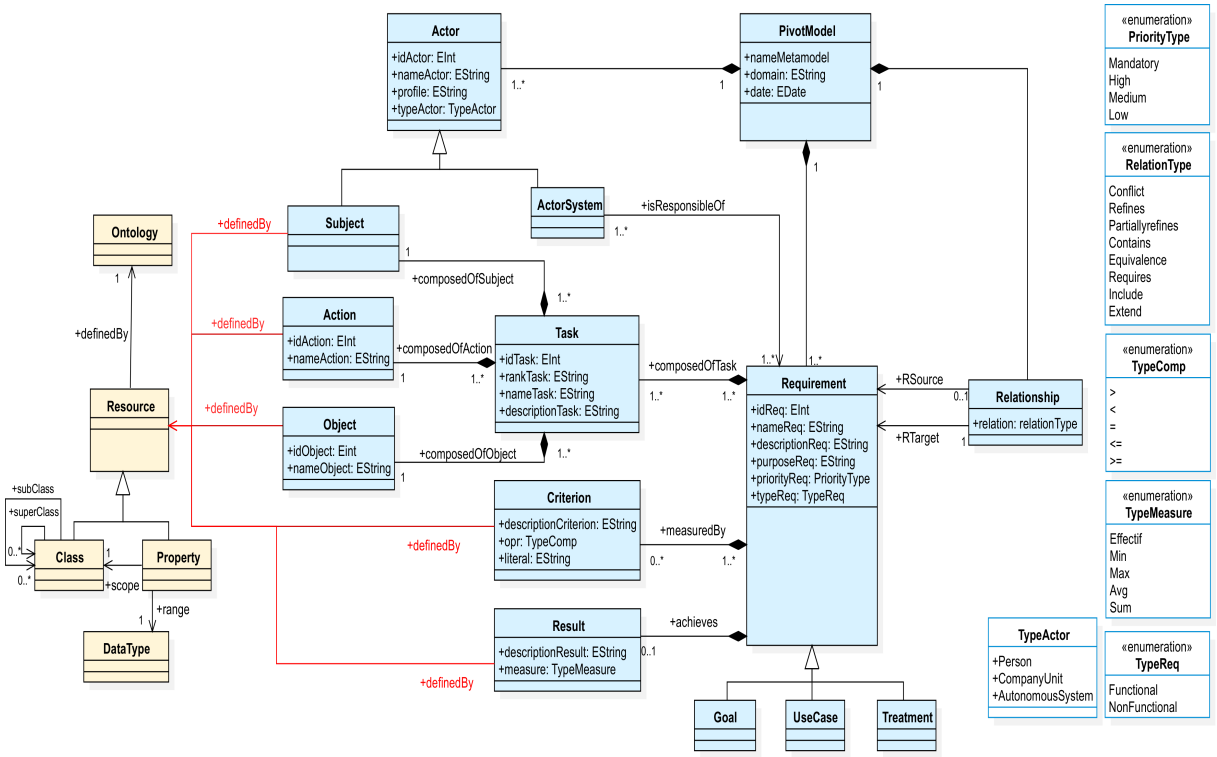


FIGURE 6.1 – Modèle de besoin connecté au modèle d'ontologie [60]

3.3.2 Sources $\mathcal{S}$

L'ensemble des sources $\mathcal{S}$ est représenté par des *BDS contextuelles*. Chaque source est définie par son schéma (ontologie locale O_i) et ses instances et est formalisée comme suit :

Formalisation 4

Les Bases de Données Sémantiques Contextuelles

$\mathcal{S}_i : < O_i, \mathcal{I}, \mathcal{P}op, \mathcal{SL}_{O_i}, \mathcal{SL}_I, \mathcal{A}r >$, où :

— O_i représente le modèle conceptuel de la source $\mathcal{S}_i$, formalisé comme suit :
$O_i : < C_i, \mathcal{P}_i, \mathcal{R}ef_i(C), \mathcal{R}ef'_i(P), \mathcal{D}efContext_i, \mathcal{R}ef_{Def_i}(C), \mathcal{E}valContext_i, \mathcal{R}ef_{Eval_i}(P), \mathcal{F}ormalism_i >$.
Nous notons que cette formalisation reste valable pour les sources non contextuelles. Dans ce cas, le modèle conceptuel d'une source non contextuelle S_j sera formalisé comme suit : $O_j : < C_j, \mathcal{P}_j, \mathcal{R}ef_j(C), \mathcal{R}ef'_j(P), \emptyset, \emptyset, \emptyset, \emptyset, \mathcal{F}ormalism_j >$.

— $\mathcal{I}$ représente l'ensemble des données (les instances des sources).

— $\mathcal{P}op : C_i \rightarrow 2^I$, est une fonction qui associe à chaque classe ses instances de l'ensemble I.

— $\mathcal{SL}_{O_i}$ est le modèle de stockage du modèle conceptuel dans la source (vertical, horizontal, etc).

— $\mathcal{SL}_I$ représente le schéma de stockage des instances. Il peut être le même modèle que

utilisé utilisé pour stocker l'ontologie ou bien un modèle différent.

— *$\mathcal{AR}$ représente l'architecture de la base de données (Deux quarts, Trois quarts ou Quatre quarts).*

3.3.3 Mappings $\mathcal{M}$

Les assertions du mapping relient des éléments d'un schéma source *S* (*MapElmS*) avec des éléments d'un schéma cible *T* (*MapElmT*).

En se basant sur la formalisation proposée dans [41] et le méta-modèle de mapping proposé dans [31], les assertions du mapping sont formalisées comme suit :

Formalisation 5

Les Mappings

$\mathcal{M}$: < $\mathcal{M}$apSchemaS, $\mathcal{M}$apSchemaT, $\mathcal{M}$apElmS, $\mathcal{M}$apElmT, $\mathcal{I}$nterpretation, $\mathcal{S}$emanticRel, $\mathcal{A}$pproach >, tel que :

— *$\mathcal{M}$apSchemaS et $\mathcal{M}$apSchemaT représentent respectivement le schéma source et le schéma cible sur lesquels seront définis les mappings.*

— *$\mathcal{M}$apElmS et $\mathcal{M}$apElmT représentent respectivement les éléments des schémas MapSchemaS et MapSchemaT qui sont concernés par le mapping. Ces éléments peuvent être des concepts, des propriétés ou des expressions.*

— *$\mathcal{I}$nterpretation représente l'interprétation intentionnelle (niveau schéma) ou extensionnelle (niveau instances) du mapping.*

— *$\mathcal{S}$emanticRel représente la relation sémantique entre les éléments $\mathcal{M}$apElmG et $\mathcal{M}$apElmS. Quatre types de relations sont possibles [31] :*

1. *Les relations d'équivalence : elles indiquent que les éléments liés par le mapping présentent la même sémantique dans le domaine décrit ;*
2. *Les relations d'appartenance exacte : elles indiquent que le premier élément du mapping est plus spécifique que le deuxième élément ;*
3. *Les relations d'appartenance complète : elles indiquent que le deuxième élément du mapping est plus spécifique ;*
4. *les relations de chevauchement : elles indiquent que les deux éléments peuvent partager une sémantique commune.*

— *$\mathcal{A}$pproach représente le sens du mapping : GaV, LaV or GLaV.*

4 Notre approche de conception des EDS Contextuels

Dans cette section, nous présentons les détails de notre approche de conception des EDS Contextuels (EDSC) dont l'architecture générale est illustrée en figure 6.2. Notre contribution concerne les quatre phases principales du cycle de conception des ED à savoir : la modélisation conceptuelle, la modélisation logique, le processus ETL et la modélisation physique [78].

4.1 Modélisation Conceptuelle

Cette phase consiste en la définition du schéma conceptuel de l'ED indépendamment de toute implémentation technique. Elle comporte deux étapes : (i) l'extraction de l'ontologie de l'ED (OED) représentant le modèle conceptuel de l'ED à partir de l'OP et (ii) l'annotation multidimensionnelle et contextuelle de l'OED.

4.1.1 Définition de l'OED

L'OED est considérée comme le modèle conceptuel de l'ED décrivant la sémantique de ses données. Elle est extraite comme un module de l'OP représentant les concepts ontologiques utilisés par les besoins des utilisateurs. L'extraction est matérialisée par l'un des trois scénarios possibles :

— $OED = OP$: l'OP correspond exactement aux besoins des utilisateurs ;

— $OED \subset OP$: l'OP couvre tous les besoins des utilisateurs. l'OED est extraite à partir de l'OP ;

— $OED \supset OP$: l'OP ne couvre pas l'ensemble des besoins des utilisateurs. Dans ce cas, le concepteur extrait le fragment de l'OP correspondant aux besoins et l'enrichit localement avec de nouveaux concepts. Par exemple, il peut utiliser des raisonneurs (par exemple Fact [90], Pellet [188], etc) afin de déduire des éléments supplémentaires étendant l'OED afin de couvrir les besoins des utilisateurs.

L'enrichissement de l'OED par des informations contextuelles doit être conforme au modèle de contexte proposé et se range sous ce dernier scénario. Deux cas sont possibles :

— Cas 1 : les informations contextuelles sont fournies explicitement lors de la phase de définition des besoins. Par exemple : l'ensemble des dépendances fonctionnelles, dépendances entre classes, etc.

— Cas 2 : les informations contextuelles sont explorées à partir de l'ontologie partagée ou à partir des sources en utilisant des algorithmes d'exploration. Par exemple, certaines études ont proposé des algorithmes pour détecter les dépendances entre les classes et les propriétés de différentes sources [174, 43].

Dans les deux cas, la définition des informations contextuelles est réalisée conformément au modèle de contexte que nous avons proposé.

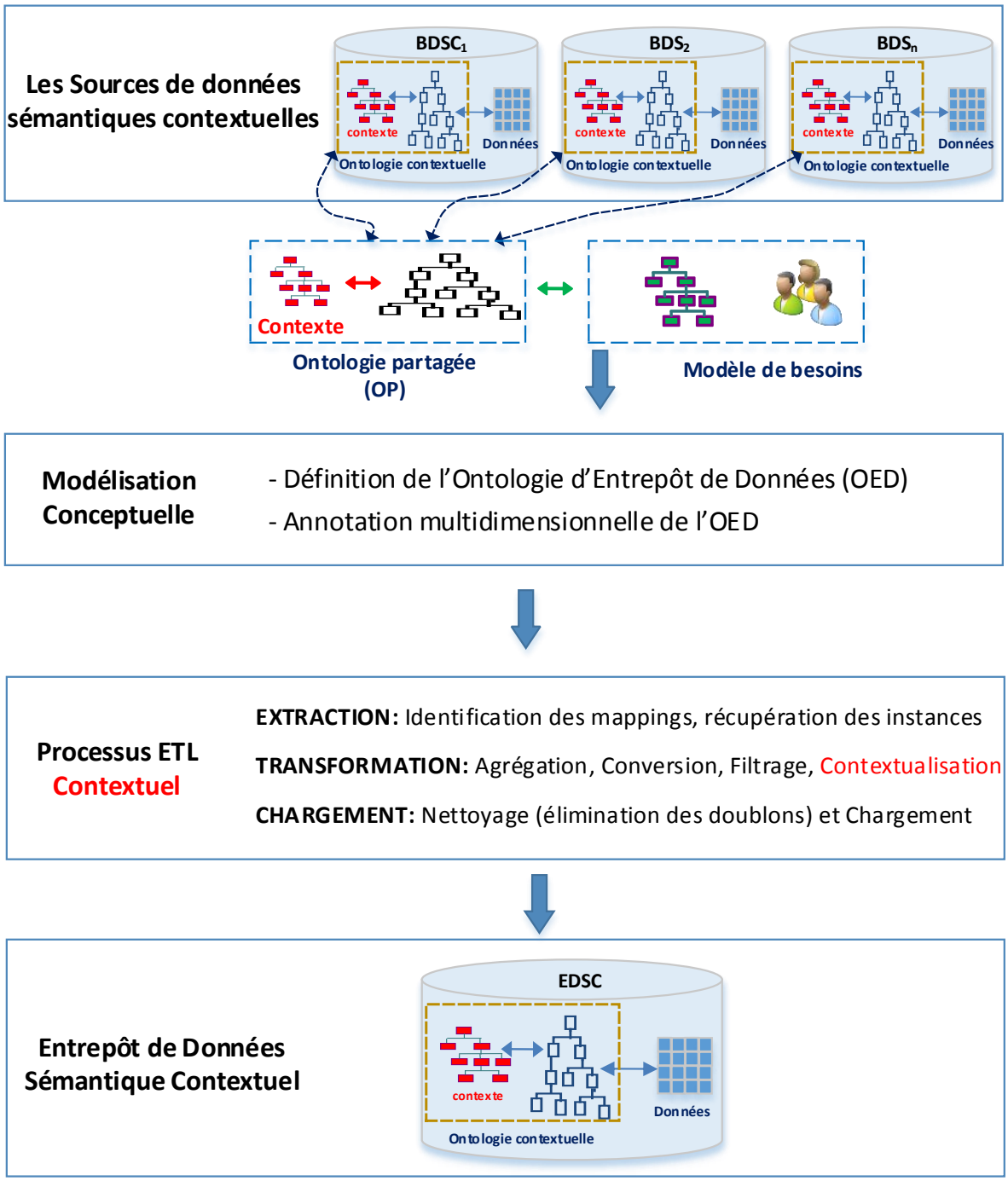


FIGURE 6.2 – Architecture générale de l'approche d'intégration proposée

4.1.2 L'annotation multidimensionnelle et contextuelle de l'OED

Cette étape consiste d'abord en l'identification des rôles multidimensionnels des concepts (faits, mesures, dimensions et attributs des dimensions) à travers l'analyse des besoins. En effet, dans les projets réels de développement des ED, le modèle multidimensionnel doit répondre

aux exigences et besoins des décideurs.

D'après [111], une modélisation multidimensionnelle se base sur l'identification des : *mesures* qui représentent des indicateurs de performances caractérisant les sujets d'analyse (*les fais*) et des *dimensions* qui représentent l'axe selon lequel ces mesures sont analysées. Notre démarche consiste à faire une projection de cette définition sur les besoins des utilisateurs. Effectivement, d'après notre vision, un besoin se compose de *résultats* qui correspondent aux indicateurs à analyser et des *critères* qui représentent les axes d'analyse. Nous utilisons ensuite les relations entre les concepts de l'OED pour valider les liens entre les faits et les dimensions ainsi que les hiérarchies de dimensions. La littérature relative à la construction des modèles multidimensionnels [116, 135, 172, 173] a proposé des configurations qui consistent à associer un fait à sa dimension par une relation *Un à Plusieurs* (relation (0..*) du côté du fait et (1..1) du côté de la dimension), et à associer une relation *Un à Plusieurs* entre le niveau de dimension inférieur (D_i) et son niveau supérieur (D_s) dans une hiérarchie de dimension (relation (1..*) du coté de D_i et (1..1) du coté de D_s).

Après avoir annoté tous les concepts et propriétés par leurs rôles multidimensionnels, nous les annotons par leurs contextes correspondants.

Nous proposons l'algorithme 1 pour formaliser cette étape. Il comporte trois étapes :

1. **Annotation des faits et de leurs mesures :**

   — Les ressources *Result* de chaque besoin B sont des candidates à représenter des *mesures* si elle sont des propriétés ou des *faits* si elle sont des classes.

   — La classe domaine d'une propriété identifiée comme *mesure* est un *fait*.

   — Les contextes correspondants à chaque fait (classe) ou mesure (propriété) sont identifiés.

   Nous définissons pour cette partie : la fonction $\mathcal{A}\text{nnot}_F(\text{C})$ qui annote la classe C par l'annotation *Fait*, la fonction $\mathcal{A}\text{nnot}_{Mes}(\text{P})$ qui annote la propriété P par l'annotation *Mesure* et les fonctions $\mathcal{A}\text{nnot}_{ContextF}(\text{F})$ et $\mathcal{A}\text{nnot}_{ContextMes}(\text{M})$ qui annotent le fait F et la mesure M respectivement par leurs contextes de définition et contextes d'évaluation s'ils existent.

2. **Annotation des dimensions et de leurs attributs :**

   — Les ressources *Criterion* du besoin B sont des candidates à représenter des *attributs de dimension* si elles sont des propriétés ou des *dimensions* si elles sont des classes.

   — La classe domaine d'une propriété identifiée comme *attribut de dimension* est une *dimension*.

   — Les contextes correspondants à chaque dimension (classe) ou attribut de dimension (propriété) sont identifiés.

   Nous définissons pour cette partie : la fonction $\mathcal{A}\text{nnot}_D(\text{C})$ qui annote la classe C par l'annotation *Dimension*, la fonction $\mathcal{A}\text{nnot}_{AttD}(\text{P})$ qui annote la Propriété P par l'annotation

attribut de dimension et les fonctions $\mathcal{A}\text{nnot}_{ContextD}$(D) et $\mathcal{A}\text{nnot}_{ContextAttD}$(AttD) qui annotent la dimension D et l'attribut AttD respectivement par leurs contextes de définition et contextes d'évaluation s'ils existent.

3. **Validation des liens entre les faits et leurs dimensions, et de la hiérarchie de dimensions :** la validation de ces liens se fait conformément aux deux contraintes de cardinalité énoncées ci-dessus (entre le fait et sa dimension et entre les niveaux de dimensions). Nous ne gardons que les classes qui respectent ces exigences. Nous définissons pour cette partie : la fonction $\mathcal{R}\text{ef}_{FD}$(C) qui retourne l'ensemble des classes liées à la classe C par une relation respectant la contrainte de cardinalités entre le fait et sa dimension, et la fonction $\mathcal{R}\text{ef}_{Hr}$(C) qui retourne l'ensemble des classes liées à la classe C par une relation respectant la contrainte de cardinalités entre les niveaux de dimensions.

4.2 Modélisation Logique

La modélisation logique consiste à traduire l'ontologie (représentant le schéma conceptuel) obtenue dans l'étape précédente en un schéma logique relationnel. Plusieurs méthodes de traduction d'une ontologie (définie par un formalisme donné : PLIB, RDF, OWL, etc) en un schéma logique relationnel ou relationnel-objet ont été proposées dans la littérature [74]. Cette traduction peut être réalisée selon les trois approches : représentation verticale, représentation horizontale ou représentation binaire.

Nous proposons les algorithmes 3 et 2 formalisant les étapes de la traduction d'une ontologie contextuelle en un schéma relationnel respectivement selon les approches horizontale et binaire.

4.3 Processus ETL

Parmi les étapes de conception des ED, le processus ETL est considéré comme la tâche la plus critique [72] et la plus fastidieuse [185, 200]. Le but de cette étape est de peupler le schéma de l'ED à partir des sources de données. Cela consiste, dans notre approche, en l'extraction, la transformation et le chargement des données à partir des sources dans l'ED (dont le schéma correspond à l'OED obtenue à l'étape précédente) tout en prenant en compte les différentes situations contextuelles qui peuvent exister.

Nous proposons ici un algorithme ETL qui se base sur un ensemble d'opérateurs conceptuels définis par *Skoutas et Simitsis* [189].

4.3.1 Opérateurs ETL

Les travaux de Skoutas et Simitsis [189] ont permis de proposer une algèbre définissant un ensemble d'opérateurs conceptuels généralement rencontrés dans un processus ETL :

Entrées : OED + ensemble de besoins
Sorties : OED annotée par les concepts multidimensionnels et les contextes

1 f, d : ressources;
2 **pour** *chaque Besoin B* **faire**
3 /⋆ Annotation des faits et de leurs mesures à partir des Besoins ⋆/;
4 **si** *Result(B) non annotée* **alors**
5 **si** *Result(B)* $\in P$ **alors**
6 $\mathcal{A}nnot_{Mes}(Result(B))$;
7 $\mathcal{A}nnot_{ContextMes}(Result(B))$;
8 $\mathcal{A}nnot_{F}(Dom(Result(B)))$;
9 $\mathcal{A}nnot_{ContextF}(Dom(Result(B)))$;
10 $f := Dom(Result(B))$;
11 **finsi**
12 **sinon si** *Result(B)* $\in C$ **alors**
13 $\mathcal{A}nnot_{F}(Result(B))$;
14 $\mathcal{A}nnot_{ContextF}(Result(B))$;
15 $f := Result(B)$;
16 **finsi**
17 **finsi**
18 /⋆ Annotation des dimensions et de leurs attributs à partir des besoins ⋆/;
19 d := Criterion(B);
20 **si** $((d \in P) \wedge (Dom(d) \in \mathcal{R}ef_{FD}(f)))$ **alors**
21 $\mathcal{A}nnot_{AttD}(d)$;
22 $\mathcal{A}nnot_{ContextAttD}(d)$;
23 $\mathcal{A}nnot_{D}(Dom(d))$;
24 $\mathcal{A}nnot_{ContextD}(Dom(d))$;
25 **finsi**
26 **sinon si** $((d \in C) \wedge (d \in \mathcal{R}ef_{FD}(f)))$ **alors**
27 $\mathcal{A}nnot_{D}(d)$;
28 $\mathcal{A}nnot_{ContextD}(d)$;
29 **finsi**
30 **finpour**
31 Rejeter les classes faits sans dimensions;
32 /⋆ Annotation des hiérarchies de dimensions ⋆/;
33 **pour** *chaque classe dimension d* **faire**
34 $\mathcal{A}nnot_{Hr}(d, \mathcal{R}ef_{Hr}(d))$;
35 **finpour**

Algorithme 1 : Annotation multidimensionnelle et contextuelle de l'OED

Entrées : Ontologie OED
Sorties : Schéma relationnel selon l'approche binaire

/⋆ Traduction des classes ⋆/
pour *chaque Classe C de l'OED (y compris les classes appartenant au modèle de contexte)* **faire**
    La classe devient une table;
    Associer la clé primaire à cette table (peut être créée automatiquement);
finpour
/⋆ Traduction des propriétés de type *Attribut* ⋆/
pour *chaque propriété Pa de type attribut (mono-valuée ou multi-valuée) (y compris les propriétés de type Attribut appartenant au modèle de contexte)* **faire**
    Pa est représentée par une table d'association nommée NomClasse_NomPropriété contenant les champs suivants :
    - une colonne de la clé primaire, nommée rid, identifiant les individus de la classe domaine ;
    - une deuxième colonne contenant les valeurs de propriété ;
    - une troisième colonne, nommée rid_Table, de type chaîne de caractères permettant de référencer la table de l'individu ;
finpour
/⋆ Traduction des propriétés de type *Relations* ⋆/
pour *chaque propriété Pr de type relation (mono-valuée ou multi-valuée) (y compris les propriétés de type Relations appartenant au modèle de contexte)* **faire**
    Pr est représentée par une table d'association nommée NomClasse_NomPropriété contenant les champs suivants :
    - une colonne pour la clé primaire, nommée rid, représentant les identifiants des instances de la classe source (domaine) ;
    - une colonne représentant les identifiants de la classe cible (co-domaine) ;
    - Deux colonnes de type chaîne de caractères spécifiant la table des instances source et cible ;
finpour

Algorithme 2 : Traduction d'une ontologie contextuelle en un modèle relationnel selon l'approche binaire

Entrées : Ontologie OED
Sorties : Schéma relationnel selon l'approche horizontale

```
/⋆ Traduction des classes ⋆/
pour chaque chaque Classe C de l'OED (y compris les classes appartenant au modèle de contexte) faire
    - La classe devient une table; - Associer la clé primaire à cette table;
    - Créer une colonne pour chaque propriété applicable à cette classe;
finpour
/⋆ Traduction des propriétés de type Attribut ⋆/
pour chaque propriété Pa de type attribut (y compris celles appartenant au modèle de contexte) faire
    si Pa est mono-valuée (i.e de cardinalité (0..1 ou 1..1)) alors
        Pa devient une colonne dans la table correspondante;
    alors
        si Pa est multi-valuée (i.e une collection) alors
            Pa devient une table d'association portant le nom (NomClasse_NomP);
            Deux colonnes sont créées dans la table :
            - une clé étrangère référençant la table correspondant à la classe définissant la propriété P ;
            - un champ contenant chacune des valeurs de la collection.
        finsi
    finsi
finpour
/⋆ Traduction des propriétés de type Relations ⋆/
pour chaque propriété Pr de type relation (y compris celles appartenant au modèle de contexte) faire
    si Pr représente une association (1..*) alors
        Pr est représentée par une clé étrangère dans la table rang référençant la clé primaire de la table domaine. Le nom de cette colonne est Nom(Pr)_ref;
        Une deuxième colonne, nommée Nom(Pr)_refTable, est ajoutée pour préciser le nom de la table de la classe domaine à laquelle appartient l'instance;
    alors
        si Pr représente une association (n..m) alors
            Pr est représentée par une nouvelle table d'association, nommée Nom(ClasseDomaine)_Nom(Pr), qui contient trois champs :
            - une clé étrangère référençant la table correspondant à la classe domaine;
            - une colonne nommée Nom(Pr)_ref pour l'identifiant de l'instance cible;
            - une colonne nommée Nom(Pr)_refTable spécifiant la table qui contient cette instance;
            La clé primaire de la table = (clé étrangère + identifiant de l'instance cible);
        finsi
    finsi
finpour
```

Algorithme 3 : Traduction d'une ontologie contextuelle en un modèle relationnel selon l'approche horizontale

— **RETRIEVE (S) :** cet opérateur permet de récupérer les données (les enregistrements) de la source S.

— **EXTRACT (I) :** cet opérateur permet d'extraire une partie (I) des données récupérées.

— **MERGE ($\mathbf{I_i}$, $\mathbf{I_j}$) :** cet opérateur permet la fusion des deux ensembles d'enregistrements I_i et I_j.

— **FILTER (I) :** cet opérateur permet de filtrer les données récupérées selon les contraintes définies par I.

— **CONVERT ($\mathbf{I_i}$, $\mathbf{I_j}$) :** cet opérateur permet de convertir les enregistrements récupérés du format défini par I_i au format défini par I_j .

— **AGGREGATE (F, $\mathbf{a_1}$,..., $\mathbf{a_n}$) :** cet opérateur permet de regrouper et d'agréger les enregistrements récupérés sur les attributs a_1,..., a_n en y appliquant la fonction F.

— **UNION (I, I') :** cet opérateur permet de fusionner les deux enregistrements I et I' appartenant à différentes sources.

— **DD (I) :** cet opérateur permet de détecter et de supprimer les doublons existant dans l'enregistrement I.

— **JOIN (I, I') :** cet opérateur permet de faire l'union (la jointure) des deux enregistrements liés par des attributs communs.

— **STORE (I) :** cet opérateur permet le chargement (le stockage) des enregistrement récupérés dans le schéma cible.

Comme notre schéma de l'ED est représenté par une ontologie (OED) et afin de prendre en charge l'aspect sémantique des sources de données, nous avons redéfini ces opérateurs ETL en les traduisant en des requêtes ontologiques à l'aide du langage *SPARQL* :

— L'opérateur **RETRIEVE** correspond, au niveau ontologique, à la recherche et la récupération des instances associées à la classe *C* et appartenant à la source *S*.
RETRIEVE (S,C) ⇒ *Select ?instances Where {?instances rdf:type Namespace:C}*
Example : rechercher les instances de la classe *Patient* appartenant à la source S_1 dont l'espace de nom est défini par le préfixe $NameSpace_1$
⇒ *Select ?instances Where {?instances rdf:type* $NameSpace_1$*:Student}*.

— L'opérateur **EXTRACT** correspond, au niveau ontologique, à l'extraction des instances de la classe *C* de la source *S* selon la condition *valeur* définie sur la propriété *DataProperty*.
EXTRACT (S,C) ⇒ *Select ?instances Where {?instances rdf:type NameSpace:C. ?instances NameSpace:DataProperty valeur}*
Example : extraire les patients adultes
⇒ *Select ?instances Where {?instances rdf:type* $NameSpace_1$*:Patient.*
?instances $NameSpace_1$*:age* ≥ *18}*.

— L'opérateur **MERGE** correspond, au niveau ontologique, à la fusion des instances des classes C_1 et C_2 appartenant à la source *S*.

MERGE (S,C_1,C_2) ⇒ *Select ?instances Where {{?instances rdf:type NameSpace:C_1} Union {?instances rdf:type NameSpace:C_2}}*
Example : fusionner les instances des classes *GeneralHospital* et *SpecialityHospital*.
⇒ *Select ?instances Where {{?instances rdf:type NameSpace$_1$:GeneralHospital} Union {?instances rdf:type NameSpace$_1$:SpecialityHospital}}*.

— L'opérateur **FILTER** consiste, au niveau ontologique, à filtrer les instances de la classe *C* de la source *S* en autorisant uniquement celles qui vérifient la condition définie sur la propriété *P*.
FILTER (S,C,P) ⇒ *Select ?instances ?p Where {?instances rdf:type NameSpace:C. ?instances NameSpace:P ?p. FILTER (?p Operator value)}*
Example : filtrer les instances de la classe *Patient* autorisant uniquement celles ayant la propriété age ≥ 30.
⇒ *Select ?instances ?p Where {?instances rdf:type NameSpace$_1$:Patient.*
?instances NameSpace$_1$:age ?p. FILTER (?p ≥ 30)}.

— L'opérateur **AGGREGATE** correspond, au niveau ontologique, à l'application de la fonction *F* (SUM, AVG, MAX, MIN, COUNT) sur les instances de la classe *C* appartenant à la source *S*.
AGGREGATE (S,C,F) ⇒ *Select (F(?instances) AS ?f) Where {?instances rdf:type NameSpace:C}*
Example : calculer le nombre des patients.
⇒ *Select (count(?instances) AS ?count) Where {?instances rdf:type NameSpace$_1$:Patient}*

— L'opérateur **UNION** correspond, au niveau ontologique, à l'union des instances associées aux classes C_1 et C_2 appartenant à des sources différentes.
UNION (S_1, C_1, S_2, C_2) ⇒ *Select ?instances Where {?instances rdf:type NameSpace$_1$:C_1} Union {?instances rdf:type NameSpace$_2$:C_2}*
Example : fusionner les instances des classes *Patient* (S_1) et *Patient* (S_2).
⇒ *Select ?instances Where {?instances rdf:type NameSpace$_1$:Patient} Union {?instances rdf:type NameSpace$_2$:Patient}*

— L'opérateur **DD** correspond, au niveau ontologique, à la détection et à la suppression des instances en double.
DD (S,C_1,C_2) ⇒ *Select Distinct ?instances Where {{?instances rdf:type NameSpace:C_1} Union {?instances rdf:type NameSpace:C_2}}*.
Example: détecter et supprimer les instances en double associées aux classes *GeneralHospital* et *HealthCareInstitution*.
⇒ *Select Distinct ?instances Where {?instances rdf:type NameSpace$_1$:GeneralHospital} Union {?instances rdf:type NameSpace$_1$:HealthCareInstitution}*.

— L'opérateur **JOIN** consiste, au niveau ontologique, à faire la jointure des instances associées aux classes C_1 et C_2 de la source *S* qui sont liées par la propriété *ObjectProperty*.
JOIN (S,C_1,C_2) ⇒ *Select ?instances1 ?instances2 Where*

{?instances1 rdf:type namespace:C_1. ?instances2 rdf:type namespace:C_2. ?instance1 namespace:ObjectProperty ?instance2}
Example: faire la jointure des classes Sojourn et HealthCareInstitution liées par la propriété effectuatedIn.
⇒ *Select ?instances1 ?instances2 Where*
{?instances1 rdf:type NameSpace$_1$:Sojourn.
?instances2 rdf:type NameSpace$_1$:HealthCareInstitution.
?instances1 NameSpace$_1$:ObjectProperty ?instances2}.

— L'opérateur **STORE** correspond, au niveau ontologique, au chargement des instances *I* associées à une classe *C* de l'*OED*. Nous utilisons, pour la traduction de cet opérateur, le langage SPARQL/Update (SPARUL).
STORE (OED,C,I) ⇒ *Insert NameSpaceED:C rdf:ID ?instances*
Example : stocker les instances (?instances) dans la classe Patient de l'ED.
⇒ *Insert NameSpace$_{ED}$:Patient rdf:ID ?instances*

Comme nos sources de données sont contextuelles et afin d'assurer l'extraction et le stockage des instances du contexte, nous avons redéfini les opérateurs **EXTRACT**, **RETRIEVE** et **STORE** comme suit (en assumant que *Context(C)* définit l'ensemble des instances de contextes représentés par l'entité *Context*) :

• **RETRIEVE**$_{context}$ (S,C) = **JOIN** (**RETRIEVE** (S,C), Context(C), concerned_property) : ce nouvel opérateur permet de récupérer les instances du concept *C* ainsi que les instances de ses contextes.
• **EXTRACT**$_{context}$ (S,C) = **JOIN** (**EXTRACT** (S,C),Context(C), concerned_property) : ce nouvel opérateur permet d'extraire, selon une condition, les instances du concept *C* ainsi que les instances de ses contextes.
• **STORE**$_{context}$ (S,C,I)= **STORE**(S,C,I) ∪ **STORE**(S, Context,Context(c)) : ce nouvel opérateur stocke les instances du concept C ainsi que les instances de ses contextes.

4.3.2 Détails de l'algorithme ETL

Avant de présenter les détails de notre algorithme ETL, nous discutons d'abord comment les mappings définis entre le schéma de l'ED et les schémas des sources impactent son exécution. Deux scénarios principaux sont identifiés :

— *Scénario 1 :* pour chaque concept *C* de l'OED, les mappings sont définis conformément à chaque contexte cxt$_i$ défini pour *C* dans l'OED. Ainsi, chaque concept *C* de l'OED défini dans un contexte cxt$_i$ est mappé avec les sources. Dans ce cas, l'algorithme ETL peut être exécuté afin de peupler chaque concept contextuel et le peuplement du concept global (englobe les définitions contextuelles) peut être réalisé en utilisant des raisonneurs ontologiques.
Exemple : le concept *Patient* est défini dans l'OED conformément à deux contextes : Patient [cxt$_1$] ⊔ Patient [cxt$_2$] tel que Patient [cxt$_1$] = Person ⊓ (∃ hasSojourn. Sojourn) et

Patient [cxt_2] = Person ⊓ (∃ hasMonitoring. MedicalMonitoring). En supposant que nous avons deux sources S_1 et S_2, ces mappings peuvent être définis comme suit :
Patient [cxt_1]$_{OED}$ ≡ [Person [S_1] ⊓ (∃ hasSojourn [S_1]. Sojourn [S_1])] ⊔ [Person [S_2] ⊓ (∃ hasHospitalization [S_2]. Hospitalization [S_2])] et
Patient [cxt_2]$_{OED}$ ≡ [Person [S_1] ⊓ (∃ hasMonitoring [S_1]. MedicalMonitoring [S_1])] ⊔ [Person [S_2] ⊓ (∃ hasMonitoring [S_2]. Monitoring [S_2]).
Ces mappings seront utilisés pour peupler les concepts Patient [cxt_1] et Patient [cxt_2]. Les instances du concept global Patient$_{OED}$ peuvent facilement être déduites en utilisant un raisonneurs.

— *Scénario 2 :* pour chaque concept *C* de l'OED, les mappings sont définis conformément à sa définition globale (et non conformément à chaque contexte). Dans ce cas, l'algorithme ETL est exécuté afin de peupler chaque concept global. Les instances des concepts contextuels peuvent être peuplées si les définitions contextuelles de chaque concept sont fournies.
Exemple : le concept Patient défini comme toute personne ayant effectué un séjour (Person ⊓ ∃ hasSojourn.Sojourn) peut être peuplé en utilisant ces mappings :
Patient$_{OED}$ ≡ [Person [S_1] ⊓ (∃ hasSojourn [S_1]. Sojourn [S_1])];
Patient$_{OED}$ ≡ [Person [S_2] ⊓ (∃ hasSojourn [S_2]. Sojourn [S_2])];

Nous détaillons dans ce qui suit les étapes de l'algorithme ETL (cf. Algorithme 4) que nous avons proposé.

Cet algorithme, basé sur les mappings définis entre le schéma de l'ED (OED) et les schémas des sources (OL_i) et sur les opérateurs conceptuels présentés ci-dessus, comporte trois étapes que nous détaillons ci-après :

— **Première étape :** elle consiste à l'identification, pour chaque concept C_{OED} de l'OED, des concepts C_{S_i} des sources dont les instances doivent être extraites pour le peupler. Ces concepts C_{S_i} sont ceux liés au concept C_{OED} par des assertions de mappings.

— **Deuxième étape :** elle consiste d'abord à (i) déterminer le type du mapping qui lie les concepts C_{S_i} des différentes sources au concept C_{OED} de l'OED et ensuite, suivant ce type, (ii) identifier les opérations de transformation à appliquer aux instances des concepts sources :

1. L'identification des assertions de mappings. En se référant au canevas d'intégration proposé dans 3.3, quatre types de mappings sont possibles : Mappings d'équivalence, Mappings d'appartenance exacte, Mappings d'appartenance complète et Mappings de chevauchement.
2. L'identification des opérations de transformation :
   (a) Le cas d'un mapping d'équivalence ($C_{OED} \equiv C_{S_i}$) ou d'appartenance exacte ($C_{S_i} \subset C_{OED}$) : toutes les contraintes du concept C_{OED} sont vérifiées par les instances du concept source. Aucune transformation des instances n'est effectuée, mais un processus d'identification des opérations ensemblistes est lancé. Les instances (les données avec leurs contextes) sont d'abord extraites (Opérateur **EXTRACT**$_{context}$) à

Entrées : (1) OED : schéma de l'ED + (2) S_i : ensemble de sources + (3) M : mappings;
Sorties : OED : Schéma + Instances;

pour *Chaque concept C de OED* **faire**
  C' : classe temporaire; $I_{OED} = \emptyset$;
  pour *Chaque source* S_i **faire**
    si *C existe dans M* **alors**
      C_S : le concept de la source S_i utilisé dans le mapping M(C) ;
      C_S : RETRIEVE$_{context}$(S_i,C_S); /⋆ récupération des instances avec contextes ⋆/
      si $C_S \equiv C$ *Or* $C_S \subset C$ /⋆ *Equivalence ou Appartenance exacte* ⋆/ **alors**
        C'= C_S ;
        sinon si $C_S \supset C$ *ou mappings de chevauchement* **alors**
          si *format*(C_S) ≠ *format*(C) **alors**
            C'= CONVERT (C_S , C);
          finsi
          si *C représente une contrainte d'agrégation définie par une fonction F* **alors**
            C' = AGGREGATE (F, C, C_S);
          finsi
          si *C présente des contraintes de sélection ou de cardinalités* **alors**
            C'= FILTER (C, C_S);
          finsi
          si C_S *dépend d'un contexte d'évaluation cxt* **alors**
            C'= CONTEXTUALIZE (C_S ,cxt);
          finsi
        finsi
      finsi
    finsi
    I_{S_i}= RETRIEVE(S,C'); /⋆ récupération des instances transformées ⋆/
    si *Plus d'une instance est identifiée dans la même source* **alors**
      I_{OED} = MERGE(I_{OED},I_{S_i});
    finsi
    si *Les instances sont associées à des concepts liés par la même propriété* **alors**
      I_{OED} = JOIN(I_{OED},I_{S_i});
    finsi
    si *Les instances sont associées à des classes de différentes sources* **alors**
      I_{OED} = UNION(I_{OED},I_{S_i});
    finsi
  finpour
  STORE$_{context}$(OED,C,DD(I_{OED}));
finpour

Algorithme 4 : Notre algorithme ETL

partir des sources et stockées dans une zone temporaire de préparation (Data Staging Area) et ensuite fusionnées (**MERGE**), unifiées (**UNION**) ou jointes (**JOIN**).

(b) Le cas d'un mapping d'appartenance complet ($C_{S_i} \supset C_{OED}$) ou de chevauchement : les instances du concept source ne vérifient pas toutes les contraintes requises par le concept C_{OED}. Dans ce cas, certaines instances doivent être transformées. Un processus d'identification des opérations de transformation (agrégation, conversion, filtrage) est lancé. Le processus de transformation est assuré par les opérateurs **AGGREGATE**, **CONVERT** et **FILTER**. Ensuite, les données transformées sont fusionnées, unifiées ou jointes.

A ces transformations explicitement engendrées par les contraintes définies par les mappings (contraintes d'agrégation, de conversion et de filtrage) s'ajoutent des transformations supplémentaires requises par les contraintes contextuelles. En effet, chaque propriété contextualisée doit être évaluée conformément aux contextes d'évaluation correspondants. Pour cela, notre algorithme ETL introduit un nouvel opérateur nommé **CONTEXTUALIZE** pour réaliser cette tâche. En effet, cet opérateur, suivant le type du contexte en question, évalue la propriété concernée comme suit :

— Contexte de type Fonction Mathématique : la propriété concernée est évaluée en utilisant la formule définie par ce contexte.

— Contexte de type Dépendance Fonctionnelle : la propriété concernée est évaluée conformément à la DF définie.

— Contexte de type Temps ou Localisation : la propriété concernée est évaluée suivant le paramètre de contexte défini.

— Contexte de type Unités de Mesure : la propriété concernée est évaluée conformément à la fonction de conversion définie.

— **Troisième étape :** elle consiste à récupérer les instances transformées (**RETRIEVE**$_{context}$), les nettoyer en éliminant les doublons (**DD**) et les charger dans l'ED (**STORE**$_{context}$).

La figure 6.3 illustre ces étapes pour une itération de notre algorithme.

4.4 Modélisation physique

Pour le déploiement de l'ED, et comme l'OED représentant le schéma de l'ED a été définie indépendamment de toute contrainte d'implémentation, nous pouvons utiliser une structure de BDS, à condition qu'elle soit augmentée par le modèle de contexte proposé (voir chapitre 5). En effet, la structure des BDS contextuelles permet de représenter et stocker, selon l'une des trois architectures possibles (deux quarts, trois quarts ou quatre quarts), les données de l'ED final avec leurs contextes et également l'ontologie explicitant leur sémantique.

Nous montrerons dans la section suivante comment nous avons utilisé le SGBD sémantique Oracle pour le déploiement d'un EDS contextuel.

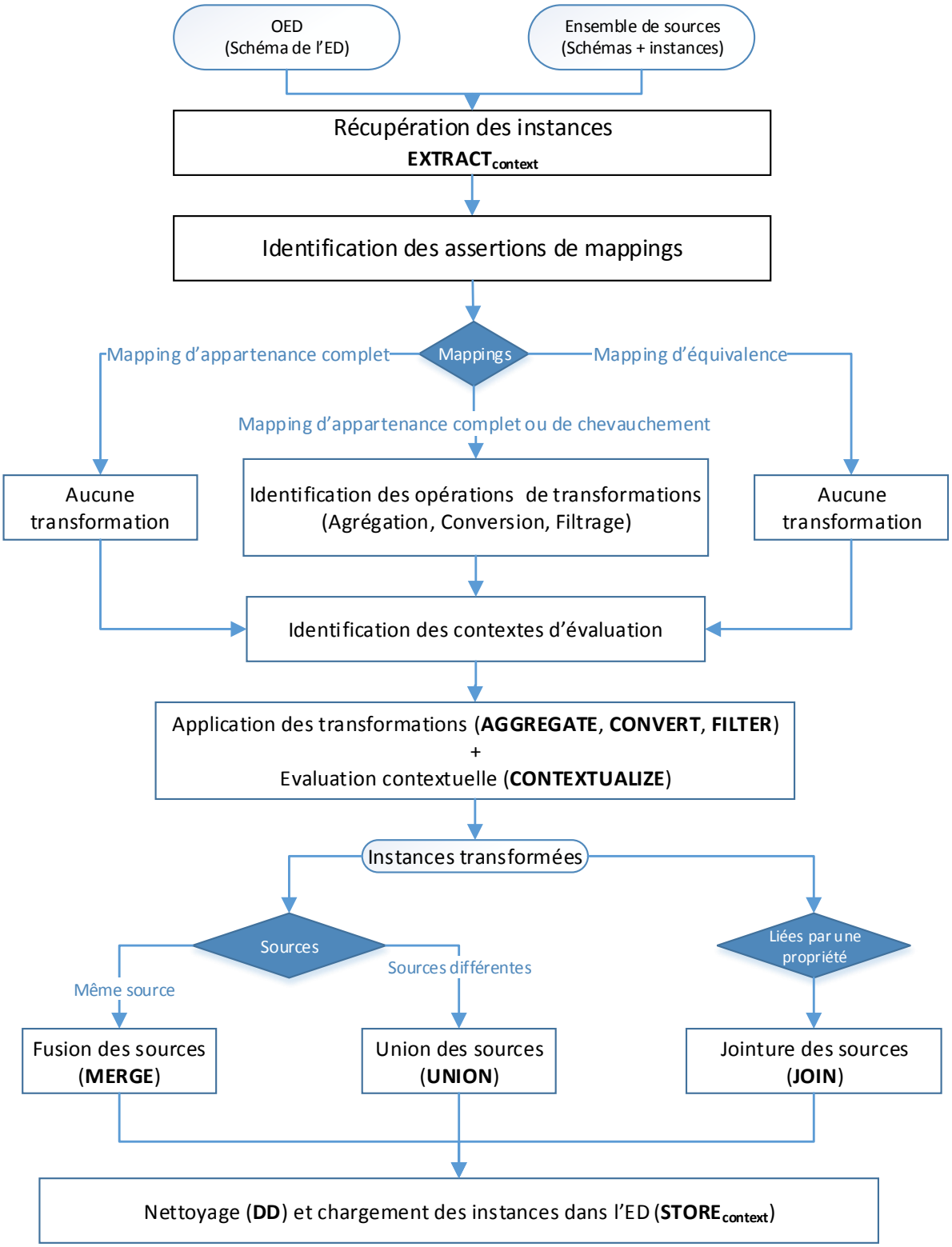


FIGURE 6.3 – Une itération de notre algorithme ETL

5 Validation : expérimentation et prototype

Dans cette section, afin de valider notre proposition et de démontrer sa faisabilité, nous introduisons d'abord un prototype d'outil implémentant notre approche de conception des EDSC. Ensuite, nous présentons une expérimentation que nous avons conduite en utilisant cette outil.

5.1 Présentation de notre prototype : CONTIC-DW Design Tool

Notre outil appelé CONtextual semanTIC Data Warehouse Design (CONTIC-DW) est un outil d'aide à la conception des EDSC qui implémente toutes les étapes présentées précédemment.

5.1.1 Environnement de développement

Pour la mise en oeuvre de notre prototype, nous avons utilisé :

— le langage *Java* avec l'environnement de développement Eclipse. Ce choix a été motivé par les avantages qu'offre ce langage en termes de portabilité, de robustesse ainsi que de disponibilité de nombreuses bibliothèques.

— l'API *Jena* [31] qui fournit un ensemble de bibliothèques Java dédiées au développement des applications orientées web sémantique. Dans notre projet, nous l'avons principalement utilisée pour manipuler les ontologies.

— l'API *GraphViz* [32] qui consiste en une collection d'outils de visualisation de graphes. Nous avons utilisé cette API pour la visualisation de nos modèles.

5.1.2 Architecture fonctionnelle de CONTIC-DW Design Tool

Comme le montre la figure 6.4, l'architecture générale de notre outil comprend cinq modules correspondant aux différentes étapes de construction d'un EDSC :

— *M1. Identification des sources de données :* ce module permet à l'utilisateur de sélectionner et de charger les sources de données participant au processus d'intégration.

— *M2. Identification de l'Ontologie Partagée (OP) :* ce module sert à identifier, charger et afficher le schéma global représenté par l'ontologie partagée.

— *M3. Définition des besoins et génération du modèle multidimensionnel :* ce module permet d'abord à l'utilisateur de spécifier ses besoins conformément à notre modèle de besoin discuté dans 3.2. Une fois les différents besoins définis, l'outil les projette sur l'OP

31. https://jena.apache.org/
32. www.graphviz.org

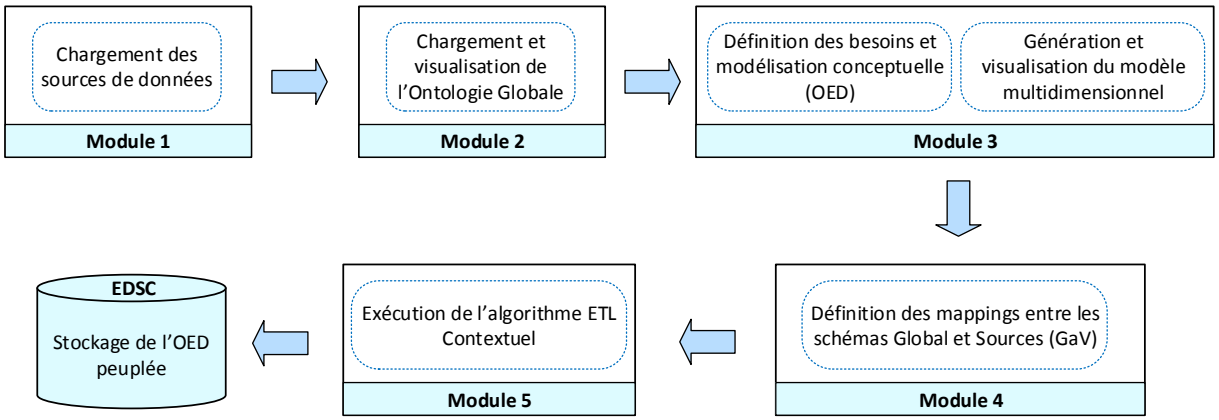


FIGURE 6.4 – Architecture fonctionnelle de l'outil *CONTIC-DW Design Tool*

afin d'extraire le schéma de l'ED (OED). Des mécanismes de raisonnement sont implémentés en utilisant le raisonneur Pellet [188] afin d'assurer sa cohérence et sa complétude. Ensuite, l'outil exécute l'algorithme 1 afin d'identifier les rôles multidimensionnels des ressources de l'OED et de les annoter. L'outil offre la possibilité de visualiser le modèle multidimensionnel généré en spécifiant les faits, les dimensions et leurs hiérarchies et les contextes.

— *M4. Définition des mappings :* ce module permet de définir les assertions de mappings entre le schéma global obtenu à l'étape précédente et les schémas des sources. Suivant l'approche GaV, le concepteur définit ces mappings en utilisant les constructeurs de LD. Ce module permet de spécifier les assertions de mappings suivant les deux scénarios définis dans 4.3.2 (suivant les définitions contextuelles ou bien suivant les définitions globales des concepts).

— *M5. Processus ETL :* en se basant sur les mappings définis, ce module, implémentant les opérateurs ETL discutés dans 4.3.1, permet d'exécuter l'algorithme 4 afin d'**extraire** les instances ontologiques des sources, les **transformer** selon les contraintes (conversion, agrégation, filtrage) ou les problèmes de contextualisation rencontrés, et, enfin, stocker les données transformées dans le schéma de l'ED.

5.2 Présentation de notre expérimentation

Nous présentons dans cette sous-section une instanciation du Framework d'intégration défini dans 3.3 pour un cas d'étude issu du domaine médical.

5.2.1 Scénario d'expérimentation

Nous avons considéré le scénario d'expérimentation suivant : il s'agit d'un organisme directeur (Ministère de la santé) qui impose aux différents centres hospitaliers qui sont sous sa tutelle

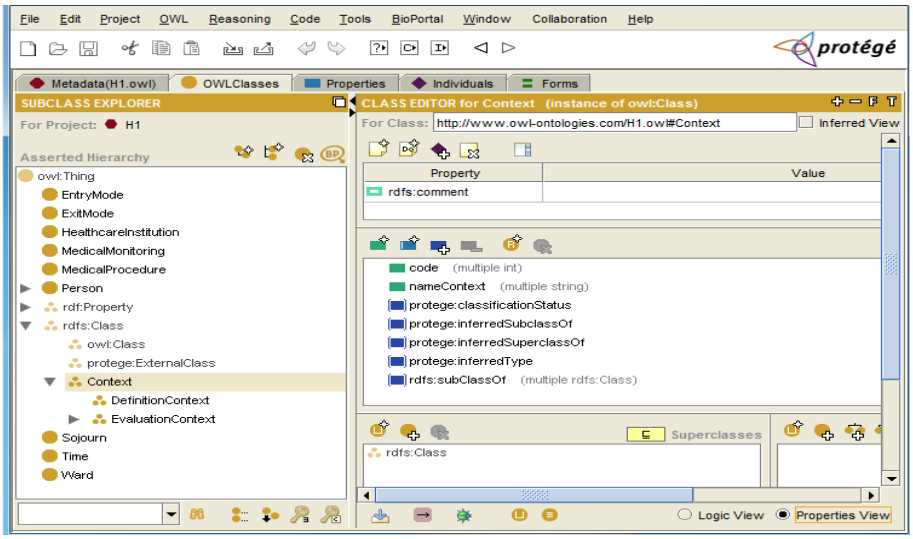


FIGURE 6.5 – Modèle d'ontologie OWL enrichi avec le modèle de contexte (sous Protégé)

d'utiliser un même vocabulaire représenté par l'ontologie partagée *Healthcare*. Chaque centre hospitalier doit : (i) se référer au même schéma global (*Healthcare Ontology*), (ii) extraire son schéma local à partir de ce schéma global en utilisant des mappings, et (iii) peupler ce schéma localement.

Dans le cadre de la prise de décision et dans le but d'améliorer le système de santé et de mesurer sa performance, cet organisme directeur doit effectuer, périodiquement, des études et des analyses sur un ensemble d'indicateurs de performances (tels que ALOS, BUR, TOI, etc) concernant les différents séjours effectués au sein de ses sites hospitaliers. Pour cela, cet organisme souhaite intégrer ces sources de données dans un Entrepôt de Données.

Afin de pouvoir dérouler notre approche de conception, nous avons créé et peuplé localement trois ontologies contextuelles représentant nos sources de données : Hôpital 1 (H1), Hôpital 2 (H2) et Hôpital 3 (H3). Comme le montrent respectivement les figures 6.5 et 6.6, nous avons enrichi le méta-modèle d'OWL avec les constructeurs de notre modèle de contexte et nous avons défini nos ontologies locales comme des fragments de l'ontologie *Healthcare* en utilisant des correspondances simples.

Dans ce qui suit, nous illustrons d'abord les différentes situations contextuelles que nous avons créées. Ensuite, nous montrons comment notre méthode pour construire l'ED cible a été appliquée.

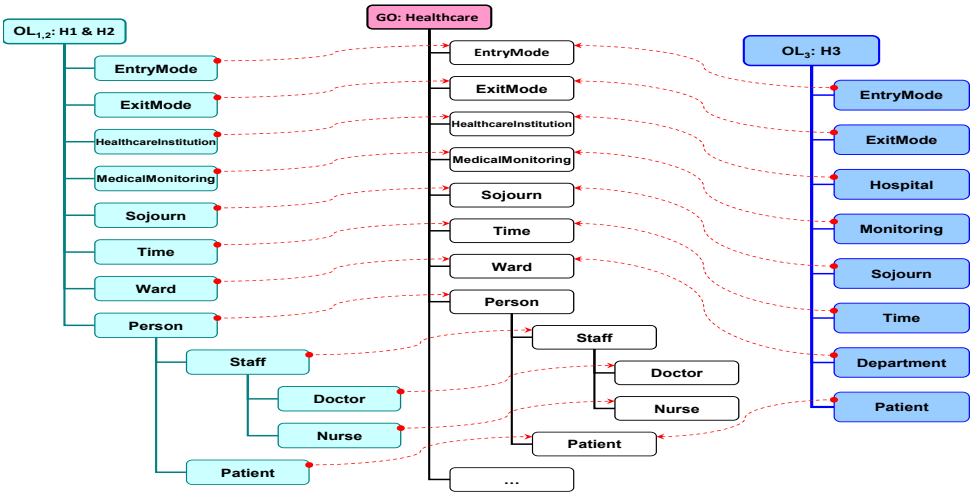


FIGURE 6.6 – Définition des OLs comme des fragments de l'OP

	Contexte de définition		Contextes d'évaluation	
	Nom	Concept	Nom	Propriété
Hôpital 1	DefCtx_1	Patient	EvalCtx_1	ALOS
	DefCtx_2	Patient	EvalCtx_2	ALOS
Hôpital 2	DefCtx1	Patient	EvalCtx1	ALOS
	DefCtx2	Patient	-	-
Hôpital 3	-	-	-	-
Ontologie Globale	DefCtx001	Patient	EvalCtx001	ALOS
	DefCtx002	Patient	EvalCtx002	ALOS

TABLE 6.1 – Échantillon de situations contextuelles définies sur les sources et l'OP

Situations contextuelles

Nous avons défini plusieurs situations contextuelles de deux types : contextes de définition et contextes d'évaluation. Au total, 46 contextes ont été définis entre les sources H1 et H2. Nous n'avons défini aucun contexte pour la source H3. La table 6.1 illustre un échantillon de ces situations contextuelles.

Déroulement de notre approche de conception

En utilisant l'outil CONTIC-DW que nous avons présenté précédemment, nous avons déroulé étape par étape notre approche de conception des EDSC :

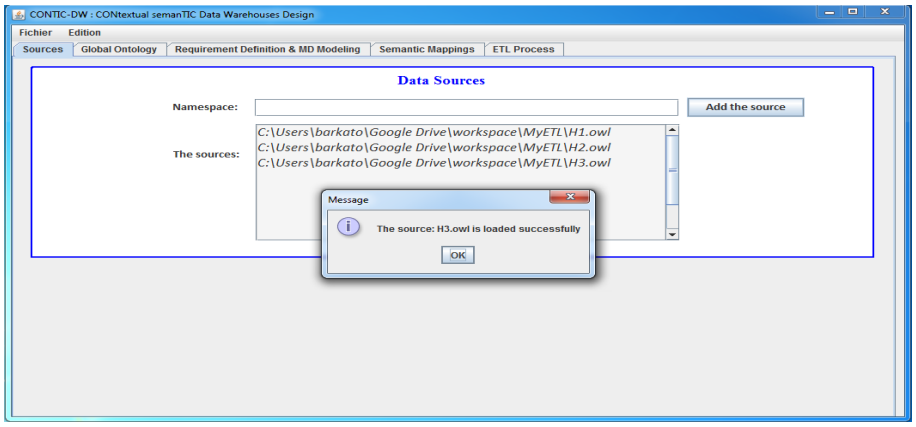


FIGURE 6.7 – Interface de l'identification et du chargement des sources de données

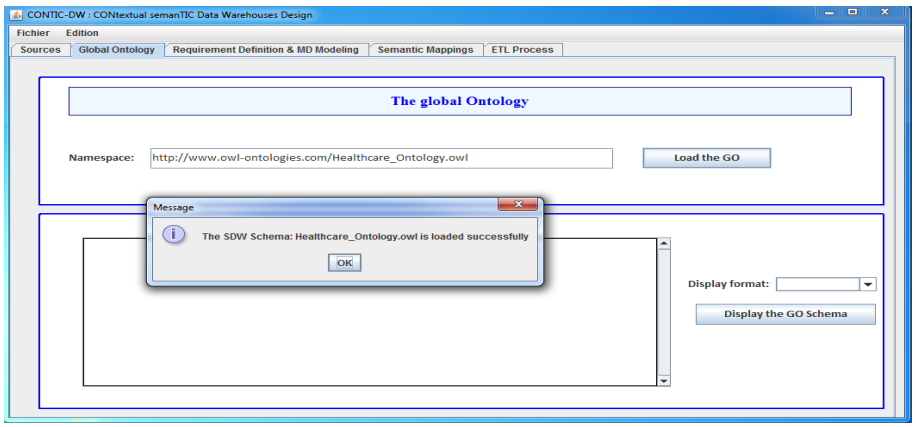


FIGURE 6.8 – Interface du choix et du chargement de l'OP

1. **Sélection des sources de données par le concepteur :** comme le montre la figure 6.7, nous avons chargé nos trois sources de données H1, H2 et H3 créées auparavant.
2. **Choix de l'ontologie partagée :** nous avons sélectionné l'ontologie *Healthcare* qui joue le rôle de l'ontologie partagée (Figure 6.8). Nous notons que cette ontologie a été enrichie par les constructeurs de notre modèle de contexte.
3. **Définition des besoins et modélisation conceptuelle :** nous avons considéré dans cette expérimentation l'ensemble des besoins suivants :

```
Req 1 : The system shall measure the value of ALOS for a given service ;
Req 2 : The system shall measure the value of ALOS for a given period ;
Req 3 : The system shall evaluate the value of ALOS per service and time ;
Req 4 : The system shall measure the BUR per service and time ;
Req 5 : The system shall provide the Blood Pressure category per patient
        and time.
```

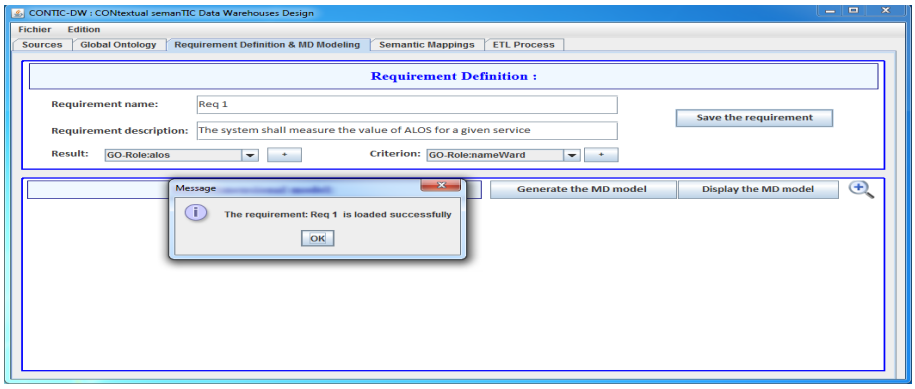


FIGURE 6.9 – Interface de définition des besoins

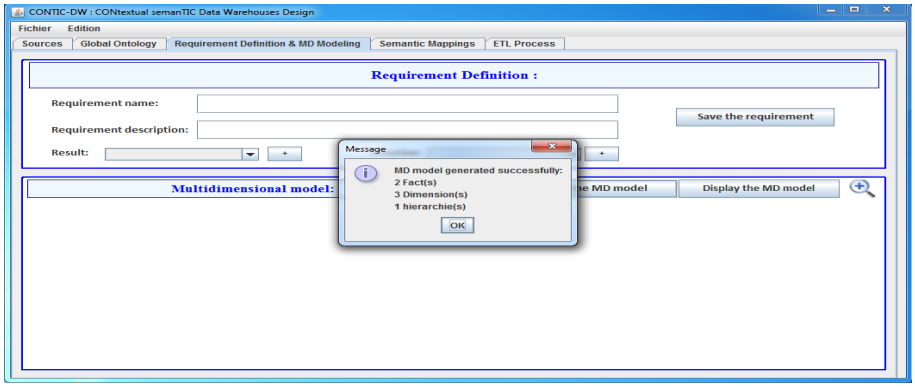


FIGURE 6.10 – Interface de génération du modèle multidimensionnel

La figure 6.9 illustre le module prévu à cet effet. Nous notons que l'ensemble des besoins définis a permis de définir l'OED comme un module de l'OP *Healthcare*.

4. **Génération du modèle multidimensionnel :** les figures 6.10 et 6.11 illustrent le modèle multidimensionnel obtenu après l'exécution de l'algorithme 1 d'annotation par notre outil. Nous constatons que ce modèle comporte deux faits (*Sojourn* et *MedicalMonitoring*), trois dimensions (*Ward*, *Time* et *Patient*), une hiérarchie (*HealthcareInstitution*) et quatre contextes (deux contextes de définition qui concernent le concept *Patient* et deux contextes d'évaluation qui concernent la propriété *ALOS*).
5. **Définition des mappings :** en se basant sur les constructeurs de la logique de description, nous avons défini les assertions de mappings illustrées dans la figure 6.6 conformément à l'approche Global as View (GaV) (cf. Figure 6.12).
6. **Processus ETL :** comme le montre la figure 6.13, nous avons lancé l'exécution de notre algorithme ETL qui, en se basant sur l'ensemble des mappings définis précédemment entre le schéma de l'OED et les schéma des sources, a procédé à l'extraction automatique des instances provenant des sources H1, H2 et H3, à leur transformation (filtrage, conversion et agrégation), au calcul des valeurs *dépendantes du contexte* et au chargement des données transformées dans l'OED.

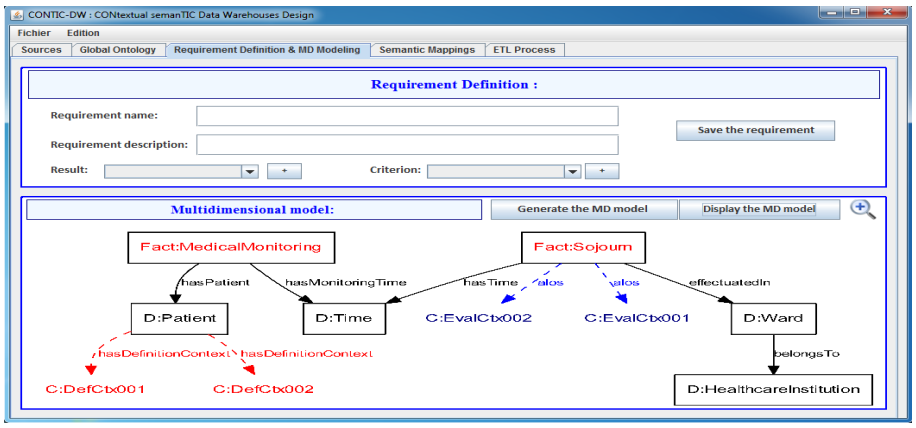


FIGURE 6.11 – Interface de visualisation du modèle multidimensionnel

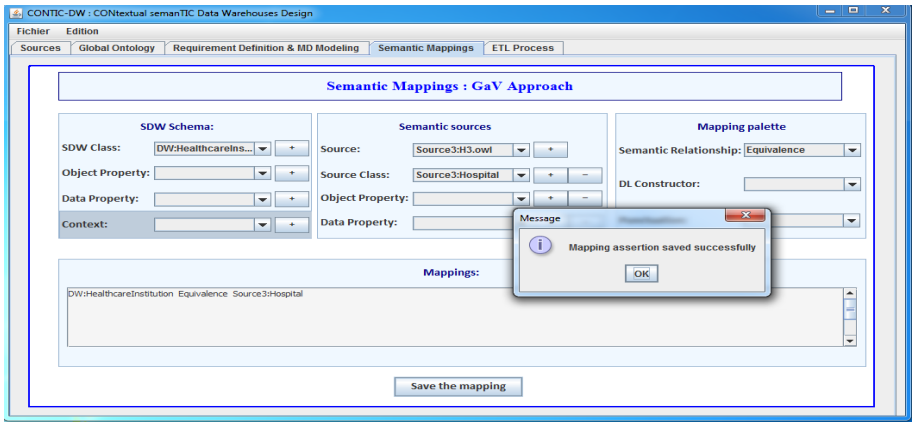


FIGURE 6.12 – Définition des OLs comme des fragments de l'OP

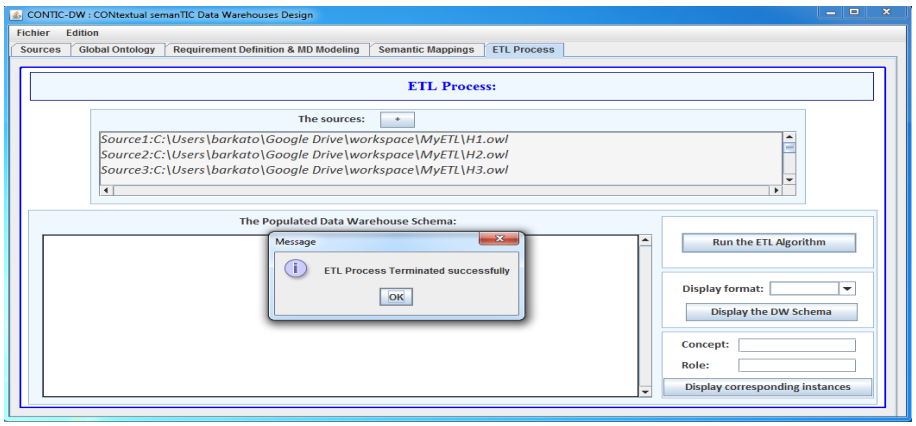


FIGURE 6.13 – Interface d'exécution du processus ETL

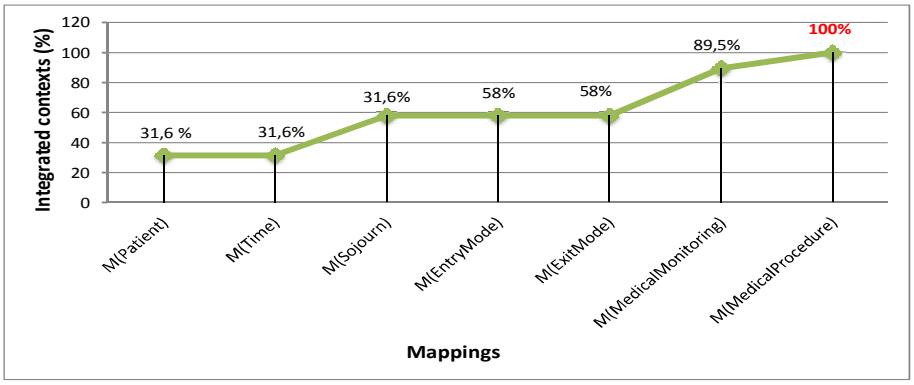


FIGURE 6.14 – Intégration des contexte par Mapping

La figure 6.14 illustre le pourcentage de contextes qui ont été intégrés en fonction des mappings. Nous notons que chaque itération de l'algorithme correspond à la population d'un concept de l'ED selon l'approche GaV. Ce résultat montre que les instances contextuelles ont été totalement intégrées dans la dernière itération de l'algorithme.

7. **Déploiement de l'EDS contextuel final :** une fois l'OED peuplée avec les instances provenant des sources, nous avons procédé à son stockage dans l'ED final. Nous avons utilisé pour le déploiement de notre entrepôt de données le SGBD sémantique Oracle 12c. Celui-ci, ayant une architecture deux quarts (cf. Chapitre 2), permet de stocker les ontologies et leurs instances dans une même table sous forme de triplets en utilisant une représentation verticale. Pour cela nous avons suivi les étapes suivantes :
   - **Conversion de notre ontologie (OED peuplée) :** Oracle ne permettant que le chargement des données au format N-TRIPLE (.nt), nous avons commencé par convertir notre ontologie au format OWL en un fichier N-TRIPLE. Nous avons utilisé pour cela l'outil *rdfcat* fourni par l'API Jena.
   - **Préparation au chargement :** cette phase de préparation au chargement concerne le SGBD Oracle. Elle consiste à effectuer les étapes suivantes : (a) activation des modules sémantiques, (b) création du réseau sémantique, (c) création d'une table sémantique et (d) création du modèle sémantique.
   - **Chargement de l'OED :** nous avons opté pour une technique de chargement dite *chargement global*. Cette dernière, connue pour sa rapidité de chargement, consiste à charger les données ontologiques dans une base de données Oracle en utilisant l'utilitaire *SQLLoader*.

   Comme le schéma de notre ED est enrichi avec les constructeurs du modèle de contexte et que notre algorithme ETL a bien intégré l'ensemble des instances contextuelles, des analyses contextuelles peuvent facilement être effectuées en utilisant des requêtes OLAP. Par exemple, la figure 6.15 illustre la requête SPARQL *R1* correspondant à la requête OLAP « la recherche des valeurs de la durée moyenne des séjours (ALOS) par service » et montre les résultats de son exécution sur notre ED.

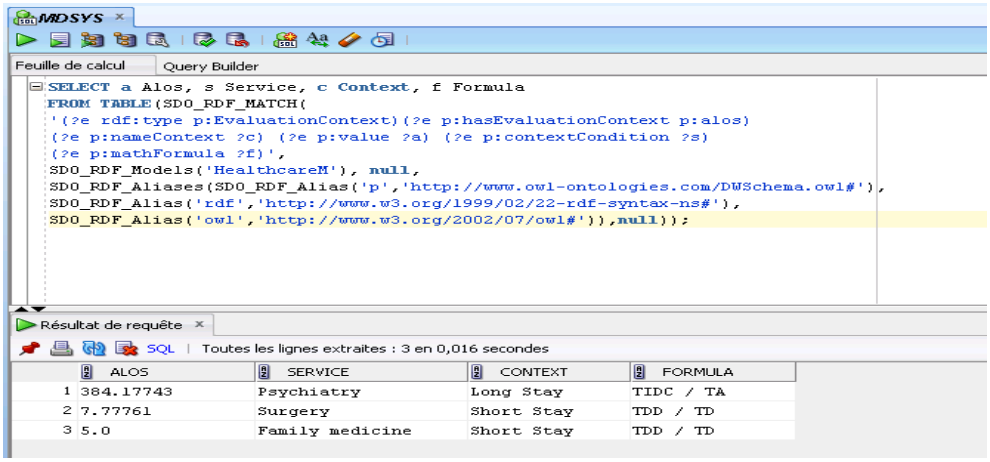


FIGURE 6.15 – Résultats de l'exécution de la requête *R1* (Oracle 12c)

Les valeurs de la mesure *ALOS* obtenues révèlent que, lors du processus d'intégration, notre algorithme ETL a bien utilisé les bonnes formules de calcul conformément aux différents contextes. En effet, comme le montre la figure 6.15 pour le service *Psychiatrie* correspondant au contexte *long séjour*, la valeur de la mesure *ALOS* a été obtenue en utilisant la formule de calcul suivante $TDD \div TD$, et pour les services *Urgences* et *Médecine générale* correspondant au contexte *court séjour*, les valeurs de la mesure *ALOS* ont été obtenues en utilisant la deuxième formule de calcul $TIDC \div TA$,

6 Conclusion

L'analyse de la littérature concernant les méthodes de conception des EDS nous a permis de constater que ces dernières ne prennent en charge le contexte que d'une manière *partielle* où l'intégralité des phases de conception n'est pas considérée et/ou d'une manière *spécifique* où la réutilisabilité et la généralité de l'approche ne sont pas possibles.

Afin de pallier ce problème dû au manque de connexions entre les ED, les Ontologies et le contexte, nous avons proposé dans ce chapitre une nouvelle approche de conception des EDS en revisitant les principales phases de construction des EDS, en l'occurrence la modélisation conceptuelle, la modélisation logique et le processus ETL afin d'intégrer la notion d'ontologies contextuelles.

Nous avons d'abord proposé des formalisations pour (i) les ontologies contextuelles, (ii) les besoins des utilisateurs définis sur les ontologies contextuelles et (iii) le canevas d'intégration de sources sémantiques contextuelles. Nous avons ensuite présenté les détails de notre approche de conception qui s'appuie principalement sur :

— l'existence d'une ontologie globale partagée par les sources de données ;

— la génération du schéma conceptuel de l'EDSC à partir des besoins des utilisateurs ;

— la définition conformément aux contextes définis de mappings entre ce schéma global et les schémas des sources ;

— l'annotation multidimensionnelle du schéma de l'ED et la définition du modèle logique ;

— un processus ETL contextuel se basant sur un ensemble d'opérateurs conceptuels définis au niveau ontologique.

Afin de valider notre proposition, nous avons déroulé, étape par étape, l'approche proposée en instanciant notre canevas d'intégration pour un cas d'étude issu du domaine médical. Cela a consisté à intégrer dans un ED trois sources sémantiques contextuelles référençant l'ontologie de domaine partagée *Healthcare*. L'expérimentation a été réalisée en utilisant un outil d'aide à la conception, appelé CONTIC-DW, qui implémente les différentes étapes de l'approche de conception proposée. Les résultats obtenus ont montré que l'intégralité des situations contextuelles a été intégrée avec succès et que l'EDSC final offre la possibilité d'effectuer des analyses contextuelles pertinentes.

Chapitre 7

Conclusion

Sommaire

1 **Conclusion** 161
2 **Perspectives** 163

Nous présentons dans ce chapitre un bilan général synthétisant nos principales contributions, ainsi qu'un ensemble de perspectives qui s'ouvrent à l'issue de notre étude.

1 Conclusion

Dans nos travaux de thèse, nous avons d'abord mis l'accent sur le rôle des ontologies dans la conception des bases de données traditionnelles/techniques et les entrepôts de données. Plusieurs travaux récents ont été menés dans les cadres académique et industriel dans cette perspective. Les ontologies ont également évolué par leur prise en compte du contexte. Malheureusement, la communauté de bases/entrepôts de données n'a pas considéré ces ontologies dans les processus de construction et d'exploitation. Au vu de la diversité et de la multidisciplinaire de nos travaux, nous avons établi un état de l'art portant sur les trois aspects de notre thèse, à savoir les systèmes de stockage de données, les ontologies et le contexte. Après des efforts de comparaison et d'analyse de ces travaux, nous avons identifié une série de constats qui ont motivé notre travail et qui peuvent être résumés par les trois points majeurs suivants :

— le manque de description et d'explicitation des dimensions du contexte et les relations qui peuvent exister entre elles. Cette situation pénalise son incorporation dans le processus de construction d'ontologies ;

— le processus de construction des ontologies est coûteux car il doit suivre un cycle de vie bien identifié qui comporte les phases suivantes [75] : la collecte des besoins, la spécification, la conceptualisation, la formalisation, le déploiement/diffusion, l'utilisation, l'évaluation, l'évolution et la maintenance. Intégrer le contexte dans ce cycle de vie complexifie le processus de construction, surtout si les dimensions de contexte ne sont pas bien explicitées ;

— passer d'une vision mono-contexte des bases/entrepôts de données à une vision multicontextuelle, ce qui permet de rendre les entrepôts de données sensibles au contexte.

Afin de répondre aux problématiques découlant de ces constats, nous avons proposé les contributions suivantes :

1.1 Approche générique pour la modélisation du contexte

Le premier constat présenté ci-dessus nous a amenés à proposer une approche de modélisation du contexte qui repose sur les points essentiels suivants. En premier lieu, nous avons présenté et discuté, à travers l'analyse des travaux existants, une série d'exigences qu'une approche de modélisation du contexte dans le domaine de l'ingénierie de données doit vérifier. Cela nous a permis d'établir une *roadmap* à suivre pour la proposition de notre approche de modélisation du contexte. En second lieu, nous avons précisé notre vision de la notion de contexte en fournissant une définition claire et compréhensible et en dégageant les différentes dimensions

et catégories du contexte. Cela a permis d'éliminer toute ambiguïté quant à l'interprétation de cette notion. En troisième lieu, nous avons proposé, en tenant compte des exigences fixées, un modèle générique du contexte que nous avons défini formellement en utilisant le langage de modélisation EXPRESS, et nous avons décrit comment ce modèle de contexte peut être lié aux ontologies. Notre modèle de contexte, à l'inverse des travaux existants relatifs aux ontologies contextuelles, est défini à l'extérieur de l'ontologie. Trois raisons principales nous ont poussés à séparer la modélisation du contexte du développement des ontologies. Premièrement, cette approche ne risque pas d'altérer l'aspect consensuel de l'ontologie vu qu'à l'inverse de cette dernière, les contextes sont, dans la plupart des cas, définis localement. Deuxièmement, le développement des ontologies est une tâche très gourmande en temps et le fait d'y incorporer le contexte conduit à son augmentation. Troisièmement, cette externalisation du contexte offre plusieurs avantages pour les concepteurs, notamment en leur donnant la liberté de choix du langage de contexte qui peut être différent de celui de l'ontologie.

Le deuxième constat nous a amené à exploiter notre modèle de contexte pour proposer deux approches consistant respectivement en l'intégration du contexte dans les BDS et en sa prise en charge dans les systèmes d'EDS. Ces deux propositions ont fait l'objet de nos deuxième et troisième contributions.

1.2 Contextualisation des BDS

Dans la deuxième contribution, nous avons proposé une approche complète pour la contextualisation des BDS en les renforçant avec la possibilité de définir, de gérer et d'interroger les différents contextes dont peuvent dépendre leurs données.

Concrètement, nous avons pris le système *OntoDB/OntoQL*, dont l'architecture est très similaire à l'architecture du MOF, comme une infrastructure de développement pour notre approche et avons réalisé les deux points suivants. Premièrement, nous avons enrichi le modèle d'ontologie de la BDS *OntoDB* avec les constructeurs de notre modèle de contexte. Cela a permis de l'augmenter avec une nouvelle fonctionnalité qui consiste à permettre la persistance et la gestion des différentes informations contextuelles dont peuvent dépendre ses données. Nous rappelons que notre approche s'applique sur n'importe quelle BDS permettant la manipulation de son modèle d'ontologie. Deuxièmement, nous avons étendu la syntaxe et la sémantique du langage *OntoQL* avec un nouvel opérateur dédié à l'interrogation contextuelle.

1.3 Contextualisation des EDS

Dans la dernière contribution, nous avons revisité les phases du cycle de vie de construction des EDS en projetant notre modèle de contexte sur chacune d'elles. Nous avons d'abord proposé un framework d'intégration sémantique générique <G, S, M> qui prend en charge les différents modèles ontologiques et les sources de données sémantiques contextuelles. Ensuite, nous avons

fourni un algorithme ETL contextuel basé sur un ensemble d'opérateurs conceptuels. Enfin, nous avons proposé une approche complète de conception d'EDS Contextuel qui explicite l'apport de la prise en charge du contexte dans les phases suivantes : la modélisation conceptuelle, la modélisation logique et la phase ETL. Un prototype d'outil implémentant l'ensemble des étapes de notre proposition a été fourni afin de faciliter les diverses tâches de conception et de guider le concepteur tout au long de ce processus.

L'ensemble de ces contributions a été validé expérimentalement en utilisant un cas d'étude issu du domaine médical.

2 Perspectives

Les travaux présentés dans ce manuscrit ouvrent un grand nombre de perspectives tant sur le plan théorique que pratique. Nous présentons dans ce qui suit une liste non exhaustive de celles qui nous paraissent être les plus intéressantes.

2.1 Enrichissement de notre modèle de contexte

Notre première contribution a consisté à proposer un modèle générique pour représenter le contexte. Nous avons pris en considération deux catégories principales, à savoir : le contexte de définition des concepts et le contexte d'évaluation des propriétés. Il serait intéressant de revoir cette catégorisation en proposant d'autres types de contexte (surtout en ce qui concerne les contextes d'évaluation) afin d'enrichir le modèle proposé. Une piste intéressante sur laquelle nous travaillons actuellement est de proposer à d'autres communautés, comme la recherche d'information, d'utiliser ce modèle et de le tester afin d'identifier s'il couvre toutes les dimensions étudiées dans leur contexte. Cela permettra de le rendre plus mature et surtout partagé par un nombre important de communautés développant des applications sensibles au contexte.

2.2 Extension du langage OntoQL

L'extension du langage *OntoQL* que nous avons proposée dans le Chapitre 5 ne concerne que la partie *interrogation* du contexte. Il serait intéressant de proposer de nouveaux constructeurs dédiés à la définition et à la manipulation des informations contextuelles de manière à ce que l'utilisateur soit capable de manipuler le contexte de façon transparente, sans avoir à connaitre les détails du modèle logique de contexte.

2.3 Proposition d'un langage OLAP contextuel

Dans nos travaux nous nous sommes concentrés sur les phases de conception d'un entrepôt de données sémantiques et contextuelles. Nous n'avons pas pu explorer la phase d'exploitation de cet entrepôt de données. Cette exploitation pourrait se faire à l'aide d'un langage de requêtes OLAP contextuel. Nous pourrions aller plus loin dans cette réflexion et étudier l'impact de ce langage sur toute la phase physique de l'entrepôt de données qui est considérée comme un tunnel pour toutes les phases de son cycle de vie, car elle est toujours liée à la performance de l'entrepôt. La présence du contexte impactera automatiquement la définition des structures d'optimisation comme les index et le partitionnement.

2.4 Développement d'un outil CASE contextuel

Les outils de conception de bases/entrepôts de données actuels comme Power AMC, AnalyseSI, Toad Data Modeler, etc n'offrent pas aux concepteurs un modèle de contexte riche afin de l'intégrer dans le processus de conception. Souvent le contexte est codé en dur par l'ajout des concepts et des propriétés contextuelles. Tout en s'inspirant des fonctionalités de ces outils payants, nous envisageons d'étendre l'outil CONTIC-DW proposé dans le cadre de cette thèse, afin d'intégrer le contexte comme composante principale dans la conception.

2.5 Les besoins et le contexte

Une thèse est en cours au laboratoire LIAS sur l'intégration de besoins hétérogènes dans le cadre des entreprises étendues, où chaque acteur peut définir ses besoins avec son propre formalisme et vocabulaire. Les besoins fonctionnels (définissant les concepts et propriétés de l'entrepôt) et non fonctionnels (définissant des contraintes qui doivent être assurées par le système d'entreposage de données telles que la sécurité, la consommation énergétique, les performances, etc.) sont au coeur du cycle de vie de construction d'un entrepôt de données. Une connexion entre le modèle de contexte et celui des besoins permettra de raffiner les traitements associés aux besoins non fonctionnels, et donc fournir un entrepôt de données assurant les contraintes imposées par ces besoins.

2.6 Expérimentation et validation de l'approche

Nous sommes en train de mener d'autres expérimentations en considérant un nombre important de sources afin d'étudier le passage à l'échelle de notre approche.

2.7 Évolution de l'ontologie et du contexte

Dans nos travaux, nous avons considéré des ontologies statiques. Ces dernières évoluent pour répondre à des nouveaux besoins et exigences. Les dimensions du contexte sont susceptibles d'évoluer. Il serait intéressant d'étudier l'impact de ces évolutions sur les applications cibles.

2.8 La considération d'autres types de données

La réflexion de cette thèse a été menée par la distinction de deux types de données, à savoir les données traditionnelles et les données techniques. Cela est motivé par l'origine et l'usage des ontologies dans deux domaines qui sont le Web sémantique et l'ingénierie. L'ensemble des nos contributions peut être revisité afin de considérer d'autres types de données tels que les données spatio-temporelles [124], les documents, les données scientifiques et statistiques, etc. Un effort de généralisation de notre approche est envisagé.

Bibliographie

[1] Alberto Abello, Oscar Romero, Torben Bach Pedersen, Rafael Berlanga, Victoria Nebot, Maria Jose Aramburu, and Alkis Simitsis. Using semantic web technologies for exploratory olap: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):571–588, 2015.

[2] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[3] Youcef Aklouf, Guy Pierra, Yamine Ait Ameur, and Habiba Drias. Plib ontology: A mature solution for products characterization in b2b electronic commerce. *International Journal of IT Standards and Standardization Research (IJITSR)*, 3(2):66–81, 2005.

[4] Sofia Alexaki, Vassilis Christophides, Gregory Karvounarakis, Dimitris Plexousakis, and Karsten Tolle. The ics-forth rdfsuite: Managing voluminous rdf description bases. In *Proceedings of the Second International Conference on Semantic Web*, pages 1–13. CEUR-WS. org, 2001.

[5] Stefan Anderlik, Bernd Neumayr, and Michael Schrefl. Using domain ontologies as semantic dimensions in data warehouses. In *International Conference on Conceptual Modeling*, pages 88–101. Springer, 2012.

[6] Yigal Arens, Chun-Nan Hsu, and Craig A Knoblock. Query processing in the sims information mediator. *Advanced Planning Technology*, 32:78–93, 1996.

[7] Franz Baader. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[8] James Bailey, François Bry, Tim Furche, and Sebastian Schaffert. Web and semantic web query languages: A survey. In *Proceedings of the First international conference on Reasoning Web*, pages 35–133. Springer-Verlag, 2005.

[9] Nick Bassiliades and Ioannis P Vlahavas. Capturing rdf descriptive semantics in an object oriented knowledge base system. In *WWW (Posters)*, 2003.

[10] C Bastien. Contexte et situation. *Houdé, O., Kayser, D., Koenig, O., Proust, J. & Rastier, F., Dictionnaire des Sciences Cognitives. Paris: PUF*, 1998.

[11] Ladjel Bellatreche, Yamine Ait-Ameur, and Chedlia Chakroun. A design methodology of ontology based database applications. *Logic Journal of IGPL*, 19(5):648–665, 2011.

[12] Ladjel Bellatreche, Nguyen Xuan Dung, Guy Pierra, and Dehainsala Hondjack. Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry*, 57(8):711–724, 2006.

[13] Ladjel Bellatreche, Selma Khouri, Ilyès Boukhari, and Rima Bouchakri. Using ontologies and requirements for constructing and optimizing data warehouses. In *2012 Proceedings of the 35th International Convention, MIPRO 2012, Opatija, Croatia, May 21-25, 2012*, pages 1568–1573, 2012.

[14] Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, and Maurizio Vincini. The momis approach to information integration. 2001.

[15] D Benslimane and A Arara. The multi-representation ontologies: a contextual description logics approach. In *The proceeding of the 15th Conference on Advanced Information Systems Engineering (CAISE 03), Klagenfurt/Velden, Austria*, pages 16–20, 2003.

[16] Djamal Benslimane, Ahmed Arara, Gilles Falquet, Zakaria Maamar, Philippe Thiran, and Faïez Gargouri. Contextual ontologies. In *Advances in Information Systems, 4th International Conference, ADVIS 2006, Izmir, Turkey, October 18-20, 2006, Proceedings*, pages 168–176, 2006.

[17] Nabila Berkani, Selma Khouri, and Ladjel Bellatreche. Generic methodology for semantic data warehouse design: From schema definition to etl. In *4th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pages 404–411. IEEE, 2012.

[18] Tim Berners-Lee. N3ql-rdf data query language. *Online only*, 2004.

[19] Cristiana Bolchini, CA Curino, Giorgio Orsi, Elisa Quintarelli, Rosalba Rossato, Fabio A Schreiber, and Letizia Tanca. And what can context do for data? *Communications of the ACM*, 52(11):136–140, 2009.

[20] Cristiana Bolchini, Elisa Quintarelli, and Letizia Tanca. Carve: Context-aware automatic view definition over relational databases. *Information Systems*, 38(1):45–67, 2013.

[21] Alex Borgida and Luciano Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, pages 36–53. Springer, 2002.

[22] Kamel Boukhalfa. *De la conception physique aux outils d'administration et de tuning des entrepôts de données*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2009.

[23] Ilyes Boukhari. *Intégration et exploitation de besoins en entreprise étendue fondées sur la sémantique*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2014.

[24] Djallel Bouneffouf. Context-based information retrieval in risky environment. *Austr. J. Intelligent Information Processing Systems*, 14(1), 2014.

[25] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing ontologies. *Journal of Web Semantics*, 1(4):1–19, 2004.

[26] Paolo Bouquet, Fausto Giunchiglia, Frank Van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-owl: Contextualizing ontologies. In *The Semantic Web-ISWC 2003*, pages 164–179. Springer, 2003.

[27] Erol Bozsak, Marc Ehrig, Siegfried Handschuh, Andreas Hotho, Alexander Maedche, Boris Motik, Daniel Oberle, Christoph Schmitz, Steffen Staab, Ljiljana Stojanovic, et al. Kaon-towards a large scale semantic web. In *E-Commerce and Web Technologies*, pages 304–313. Springer, 2002.

[28] Patrick Brézillon. Context in problem solving: a survey. *The Knowledge Engineering Review*, 14(01):47–80, 1999.

[29] Patrick Brézillon. Context dynamic and explanation in contextual graphs. In *Modeling and Using Context*, pages 94–106. Springer, 2003.

[30] Dan Brickley and Ramanathan V Guha. Rdf vocabulary description language 1.0: Rdf schema. 2004.

[31] Saartje Brockmans, Peter Haase, Luciano Serafini, and Heiner Stuckenschmidt. Formal and conceptual comparison of ontology mapping languages. In *Modular Ontologies*, pages 267–291. Springer, 2009.

[32] Jeen Broekstra and Arjohn Kampman. Serql: a second generation rdf query language. In *Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, pages 13–14, 2003.

[33] Jeen Broekstra, Arjohn Kampman, and Frank Van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *The Semantic Web-ISWC 2002*, pages 54–68. Springer, 2002.

[34] Peter J Brown. Triggering information by context. *Personal Technologies*, 2(1):18–27, 1998.

[35] Peter J Brown, John D Bovey, and Xian Chen. Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE*, 4(5):58–64, 1997.

[36] Robert Bruckner, Beate List, and Josef Scheifer. Developing requirements for data warehouse systems with use cases. *AMCIS 2001 Proceedings*, page 66, 2001.

[37] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi. Data integration through {\ textit {DL-Lite} _ {\ mathcal A}} ontologies. In *International Workshop on Semantics in Data and Knowledge Bases*, pages 26–47. Springer, 2008.

[38] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. A principled approach to data integration and reconciliation in data warehousing. In *DMDW*, volume 99, page 16, 1999.

[39] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Data integration in data warehousing. *International Journal of Cooperative Information Systems*, 10(03):237–271, 2001.

[40] Jeremy J Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th international conference on World Wide Web*, pages 613–622. ACM, 2005.

[41] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(04):375–398, 1993.

[42] Stefano Ceri, Mauro Negri, and Giuseppe Pelagatti. Horizontal data partitioning in database design. In *Proceedings of the 1982 ACM SIGMOD international conference on Management of data*, pages 128–136. ACM, 1982.

[43] Chedlia Chakroun. *Contribution à la définition d'une méthode de conception de bases de données à base ontologique*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2013.

[44] Matthew Chalmers. A historical view of context. *Computer Supported Cooperative Work (CSCW)*, 13(3-4):223–247, 2004.

[45] Bor-sen Chen, Chia-hung Chang, and Hsiao-ching Lee. The entity-relationship approach. In *in Information Technology in Action: Trends and Perspectives*. Citeseer, 1993.

[46] Guanling Chen, David Kotz, et al. A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[47] Suan Khai Chong, Ian McCauley, Seng Wai Loke, and Shonali Krishnaswamy. Context-aware sensors and data muffling. *Context awareness for self-managing systems (devices, applications and networks) proceeding*, pages 103–117, 2007.

[48] Hong-Tai Chou and David J DeWitt. An evaluation of buffer management strategies for relational database systems. *Algorithmica*, 1(1-4):311–336, 1986.

[49] Edgar F Codd, Sharon B Codd, and Clynch T Salley. Providing olap (on-line analytical processing) to user-analysts: An it mandate. *Codd and Date*, 32, 1993.

[50] Dan Connolly, Frank Van Harmelen, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, and Lynn Andrea Stein. Daml + oil (march 2001) reference description. 2001.

[51] P Coret, J Richard, E Talavet, and Trofimoya. Introduction a owl, langage xml d'ontologies. Technical report, Technical report, 2006.

[52] Nadine Cullot, Christine Parent, Stefano Spaccapietra, and Christelle Vangenot. Ontologies: A contribution to the dl/db debate. In *Proc. of the 1st International Workshop on the Semantic Web and Databases, 29th International Conference on Very Large DataBases*, number LBD-CONF-2003-005, 2003.

[53] Souripriya Das. Rdf support in oracle rdbms. *Oracle Technical Presentation*, 2009.

[54] Simon Zayas David. *A framework for the management of heterogeneous models in system engineering*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2012.

[55] Hondjack Dehainsala, Guy Pierra, and Ladjel Bellatreche. Ontodb: An ontology-based database for data intensive applications. In *Advances in Databases: Concepts, Systems and Applications*, pages 497–508. Springer, 2007.

[56] Anind K Dey, Gregory D Abowd, and Andrew Wood. Cyberdesk: A framework for providing self-integrating context-aware services. *Knowledge-Based Systems*, 11(1):3–13, 1998.

[57] Anind K Dey and Context-Aware Computing. The cyberdesk project. aaai 1998 spring symposium on intelligent environments. Technical report, Technical Report. 1998 (SS-98-02), 1998.

[58] Tapucu Dilek. *A generic model for handling preferences in Ontology Based Databases*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2010.

[59] Zouhir Djilani, Nabila Berkani, and Ladjel Bellatreche. Towards functional requirements analytics. In *International Symposium on Leveraging Applications of Formal Methods*, pages 358–373. Springer, 2016.

[60] Zouhir Djilani and Selma Khouri. Understanding user requirements iceberg: Semantic based approach. In *MEDI Conference, Springer*, pages 297–310, 2015.

[61] Mourad El Hadj Mimoune. *Contribution à la modélisation explicite et à la représentation des données de composants industriels: application au modèle PLIB*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2004.

[62] Lama El Sarraj. *Exploitation d'un entrepôt de données guidée par des ontologies: application au management hospitalier*. PhD thesis, Aix-Marseille, 2014.

[63] Hicham G Elmongui, Walid G Aref, and Mohamed F Mokbel. Chameleon: Context-awareness inside dbmss. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1335–1338. IEEE, 2009.

[64] Jérôme Euzenat, Jérôme David, Angela Locoro, and Armen Inants. Context-based ontology matching and data interlinking. Technical report, Lindicle, 2015.

[65] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[66] Chimène Fankam. *OntoDB2: un système flexible et efficient de Base de Données à Base Ontologique pour le Web sémantique et les données techniques*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2009.

[67] Chimène Fankam, Ladjel Bellatreche, Dehainsala Hondjack, Yamine Ait Ameur, and Guy Pierra. Sisro, conception de bases de données à partir

d'ontologies de domaine. *Technique et Science Informatiques*, 28(10):1233–1261, 2009.

[68] Ling Feng, Peter MG Apers, and Willem Jonker. Towards context-aware data management for ambient intelligence. In *Database and expert systems applications*, pages 422–431. Springer, 2004.

[69] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[70] David Franklin and Joshua Flaschbart. All gadget and no representation makes jack a dull environment. In *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments*, pages 155–160, 1998.

[71] Marcelo Freitas, Jimmy Silva, Davi Bandeira, Antônio Mendonça, Damires Souza, and Ana Carolina Salgado. A user context management approach for query personalization settings. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, pages 333–335. IEEE, 2012.

[72] T Friedman. Gartner says more than 50 percent of data warehouse projects will have limited acceptance or will be failures through 2007. *Retrieved February*, 21:2007, 2005.

[73] Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon. Ajax: an extensible data cleaning tool. In *ACM Sigmod Record*, volume 29, page 590. ACM, 2000.

[74] Anuradha Gali, Cindy X Chen, Kajal T Claypool, and Rosario Uceda-Sosa. From ontology to relational databases. In *Conceptual modeling for advanced application domains*, pages 278–289. Springer, 2004.

[75] Fabien Gandon. *Ontology Engineering: a survey and a return on experience*. PhD thesis, INRIA, 2002.

[76] Irene Garrigós, Jesús Pardillo, Jose-Norberto Mazón, and Juan Trujillo. A conceptual modeling approach for olap personalization. In *International Conference on Conceptual Modeling*, pages 401–414. Springer, 2009.

[77] Martin Glinz. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 21–26. IEEE, 2007.

[78] Matteo Golfarelli. Data warehouse lifecycle and design. In *Encyclopedia of Database Systems*, pages 658–664. Springer, 2009.

[79] Matteo Golfarelli and Stefano Rizzi. *Data warehouse design: Modern principles and methodologies*. McGraw-Hill, Inc., 2009.

[80] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[81] gu jun zhong. Context aware computing. *Journal of East China Normal University*, 5:1–20, 2009.

[82] Donghai Guan, Weiwei Yuan, Sungyoung Lee, and Young-Koo Lee. Context selection and reasoning in ubiquitous computing. In *Intelligent Pervasive Computing, 2007. IPC.*

The 2007 International Conference on, pages 184–187. IEEE, 2007.

[83] Pierra Guy, Ait-Ameur Yamine, Bellatreche Ladjel, Dehainsala Hondjack, Jean Stéphane, Fankam Chimène, and Nguyen Xuan Dung. Données à base ontologique: gestion, interrogation, intégration. In *Première édition des Journées Francophones sur les Ontologies (JFO 2007)*, 2007.

[84] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *6th international conference on Pervasive computing and applications (ICPCA)*, pages 363–366. IEEE, 2011.

[85] Stephen Harris and Nicholas Gibbins. 3store: Efficient bulk rdf storage. 2003.

[86] Albert Held, Sven Buchholz, and Alexander Schill. Modeling of context information for pervasive computing applications. In *Procceding of the World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

[87] Karen Henricksen. *A framework for context-aware pervasive computing applications*. University of Queensland Queensland, Australia, 2003.

[88] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. Context-awareness on mobile devices-the hydrogen approach. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 10–pp. IEEE, 2003.

[89] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[90] Ian Horrocks. Using an expressive description logic: Fact or fiction? *KR*, 98:636–645, 1998.

[91] Richard Hull, Philip Neaves, and James Bedford-Roberts. Towards situated computing. In *First International Symposium on Wearable Computers*, pages 146–153. IEEE, 1997.

[92] William H Inmon. *Building the data warehouse*. John wiley & sons, 2005.

[93] Intellidimension. Rdf gateway. *Online only*, 2004.

[94] Ivar Jacobson, Kurt Bittner, and I Spence. Use case modeling, 2002.

[95] Patrick Jagou. *Concurrent engineering: la maîtrise des coûts, des délais et de la qualité*. Hermès, 1993.

[96] S Jean. Langage d'exploitation de base de données ontologiques. *Mémoire pour l'obtention du DEA T3IA, Université de Poiters*, 2004.

[97] Stéphane Jean, Yamine Aït Ameur, and Guy Pierra. Querying ontology based databases-the ontoql proposal. In *SEKE*, pages 166–171, 2006.

[98] Stéphane Jean, Hondjack Dehainsala, Dung Nguyen Xuan, Guy Pierra, Ladjel Bellatreche, and Yamine Aït Ameur. Ontodb: It is time to embed your domain ontology in your database. In *Advances in Databases: Concepts, Systems and Applications, 12th International Conference on Database Systems for Advanced Applications, DASFAA 2007, Bangkok, Thailand, April 9-12, 2007, Proceedings*, pages 1119–1122, 2007.

[99] Stéphane Jean. *OntoQL, un langage d'exploitation des bases de données à*

base ontologique. PhD thesis, Université de Poitiers, 2007.

[100] Stéphane Jean, Guy Pierra, and Yamine Ait-Ameur. Domain ontologies: A database-oriented analysis. In *Web Information Systems and Technologies*, pages 238–254. Springer, 2007.

[101] Houssem Jerbi, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Management of context-aware preferences in multidimensional databases. In *Third International Conference on Digital Information Management, 2008. ICDIM 2008*, pages 669–675. IEEE, 2008.

[102] Lalana Kagal, Vladimir Korolev, Sasikanth Avancha, Anupam Joshi, Tim Finin, and Yelena Yesha. Centaurus: an infrastructure for service management in ubiquitous computing environments. *Wireless Networks*, 8(6):619–635, 2002.

[103] Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. Rql: a declarative query language for rdf. In *Proceedings of the 11th international conference on World Wide Web*, pages 592–603. ACM, 2002.

[104] Royer Kevin. *Vers un entrepôt de données et des processus : le cas de la mobilité électrique chez EDF*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2015.

[105] Selma Khouri. *Cycle de vie sémantique de conception de systèmes de stockage et manipulation de données*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2013.

[106] Selma Khouri, Sabrina Abdellaoui, and Fahima Nader. Avoiding ontology confusion in etl processes. In *East European Conference on Advances in Databases and Information Systems*, pages 119–126. Springer, 2015.

[107] Selma Khouri, Ladjel Bellatreche, Stéphane Jean, and Yamine Ait-Ameur. Requirements driven data warehouse design: We can go further. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, pages 588–603. Springer, 2014.

[108] Selma Khouri, Ilyès Boukhari, Ladjel Bellatreche, Eric Sardet, Stéphane Jean, and Mickaël Baron. Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. *Computers in Industry*, 63(8):799–812, 2012.

[109] Selma Khouri, Lama El Saraj, Ladjel Bellatreche, Bernard Espinasse, Nabila Berkani, Sophie Rodier, and Thérèse Libourel. Cidhouse: Contextual semantic data warehouses. In *Database and Expert Systems Applications*, pages 458–465. Springer, 2013.

[110] Oleksiy Khriyenko and Vagan Terziyan. Context description framework for the semantic web. In *Proceedings Context 2005 Context representation and reasoning workshop, Paris (FR)*, 2005.

[111] Ralph Kimball and Margy Ross. *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons, 2011.

[112] Pyrros Koletsis and Euripides GM Petrakis. Sia: Semantic image annota-

tion using ontologies and image content analysis. In *International Conference Image Analysis and Recognition*, pages 374–383. Springer, 2010.

[113] Darl Kuhn, Sam R Alapati, and Bill Padfield. Sql access advisor. In *Expert Indexing in Oracle Database 11g*, pages 233–248. Springer, 2012.

[114] Bellatreche Ladjel. *Contributions à la Conception et l'Exploitation des Systèmes d'Intégration de Données*. PhD thesis, Mémoire d'habilitation à diriger la recherche, Université de Poitiers, 2009.

[115] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[116] Wolfgang Lehner, Jens Albrecht, and Hartmut Wedekind. Normal forms for multidimensional databases. In *Proceedings. Tenth International Conference on Scientific and Statistical Database Management*, pages 63–72. IEEE, 1998.

[117] Justin J Levandoski, Mohamed E Khalefa, and Mohamed F Mokbel. An overview of the caredb context and preference-aware database system. *IEEE Data Eng. Bull.*, 34(2):41–46, 2011.

[118] Justin J Levandoski, Mohamed F Mokbel, and Mohamed E Khalefa. Caredb: A context and preference-aware location-based database system. *Proceedings of the VLDB Endowment*, 3(1-2):1529–1532, 2010.

[119] Wen-Syan Li, Junho Shim, K Selcuk Candan, and Yoshinori Hara. Webdb: A web query system and its modeling, language, and implementation. In *Proceedings. IEEE International Forum on Research and Technology Advances in Digital Libraries. ADL 98*, pages 216–227. IEEE, 1998.

[120] Omar López-Ortega and Moramay Ramírez-Hernández. A step-based manufacturing information system to share flexible manufacturing resources data. *J. Intelligent Manufacturing*, 16(3):287–301, 2005.

[121] Jing Lu, Li Ma, Lei Zhang, Jean-Sébastien Brunner, Chen Wang, Yue Pan, and Yong Yu. Sor: a practical system for ontology storage, reasoning and search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 1402–1405. VLDB Endowment, 2007.

[122] Li Ma, Zhong Su, Yue Pan, Li Zhang, and Tao Liu. Rstar: an rdf storage and query system for enterprise resource management. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 484–491. ACM, 2004.

[123] Cristina Maier, Debabrata Dash, Ioannis Alagiannis, Anastasia Ailamaki, and Thomas Heinis. Parinda: an interactive physical designer for postgresql. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 701–704. ACM, 2010.

[124] Marwa Manaa, Ladjel Bellatreche, Jalel Akaichi, and Selma Khouri. Towards an ontology-based pivot model for spatio-temporal sources. In *11th Asia-Pacific Conference on Conceptual Modelling,*

APCCM 2015, Sydney, Australia, January 2015, pages 109–114, 2015.

[125] M Marchiori, A Epifani, and S Trevisan. Metalog v2. 0: Quick user guide. *rap. tech., Technical report, W3C*, 2004.

[126] Davide Martinenghi and Riccardo Torlone. Querying context-aware databases. In *Flexible Query Answering Systems*, pages 76–87. Springer, 2009.

[127] Davide Martinenghi and Riccardo Torlone. A logical approach to context-aware databases. In *Management of the Interconnected World*, pages 211–219. Springer, 2010.

[128] Keita Matsuyama, Michael Kraus, Kazuhiro Kitagawa, and Nobuo Saito. A path-based rdf query language for cc/pp and uaprof. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pages 3–7. IEEE, 2004.

[129] Jose-Norberto Mazón and Juan Trujillo. Enriching data warehouse dimension hierarchies by using semantic relations. In *British National Conference on Databases*, pages 278–281. Springer, 2006.

[130] Brian McBride. Jena: Implementing the rdf model and syntax specification. In *SemWeb*, 2001.

[131] John McCarthy. Notes on formalizing context. In *Proceedings of the 13th international joint conference on Artifical intelligence*, pages 555–560. Morgan Kaufmann Publishers Inc., 1993.

[132] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[133] Peter Mika. Ontologies are us: A unified model of social networks and semantics. In *International semantic web conference*, pages 522–536. Springer, 2005.

[134] Libby Miller, Andy Seaborne, and Alberto Reggiori. Three implementations of squishql, a simple rdf query language. In *International Semantic Web Conference*, pages 423–435. Springer, 2002.

[135] Daniel L Moody and Mark AR Kortink. From enterprise models to dimensional models: a methodology for data warehouse and data mart design. In *DMDW*, page 5, 2000.

[136] Chuck Murray, Nicole Alexander, Souri Das, George Eadon, and Siva Ravada. Oracle spatial resource description framework (rdf). *Oracle Corporation*, 2005.

[137] Belaid Nabil. *Modélisation de services et de workflows sémantiques à base d'ontologies de services et d'indexations. Application à la modélisation géologique.* PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2011.

[138] Felix Naumann, Alexander Bilke, Jens Bleiholder, and Melanie Weis. Data fusion in three steps: Resolving schema, tuple, and value inconsistencies. *IEEE Data Eng. Bull.*, 29(2):21–31, 2006.

[139] Victoria Nebot and Rafael Berlanga. Building data warehouses with semantic web data. *Decision Support Systems*, 52(4):853–868, 2012.

[140] Bernd Neumayr, Stefan Anderlik, and

Michael Schrefl. Towards ontology-based olap: datalog-based reasoning over multidimensional ontologies. In *Proceedings of the fifteenth international workshop on Data warehousing and OLAP*, pages 41–48. ACM, 2012.

[141] Dung Nguyen Xuan. *Intégration de bases de données hétérogènes par articulation a priori d'ontologies: application aux catalogues de composants industriels*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2006.

[142] Marko Niinimäki and Tapio Niemi. An etl process for olap using rdf/owl ontologies. In *Journal on Data Semantics XIII*, pages 97–119. Springer, 2009.

[143] U Ogbuji. Thinking xml: Basic xml and rdf techniques for knowledge management: Part 6: Rdf query using versa. *Online only*, 2002.

[144] Lamia Oukid, Ounas Asfari, Fadila Bentayeb, Nadjia Benblidia, and Omar Boussaid. Cxt-cube: contextual text cube model and aggregation operator for text olap. In *Proceedings of the sixteenth international workshop on Data warehousing and OLAP*, pages 27–32. ACM, 2013.

[145] Zhengxiang Pan and Jeff Heflin. Dldb: Extending relational databases to support semantic web queries. Technical report, DTIC Document, 2004.

[146] Jesús Pardillo and Jose-Norberto Mazón. Using ontologies for the design of data warehouses. *CoRR*, abs/1106.0304, 2011.

[147] Myung-Jae Park, Jihyun Lee, Chun-Hee Lee, Jiexi Lin, Olivier Serres, and Chin-Wan Chung. An efficient and scalable management of ontology. *Advances in Databases: Concepts, Systems and Applications*, pages 975–980, 2007.

[148] Jason Pascoe. Adding generic contextual capabilities to wearable computers. In *Second International Symposium on Wearable Computers, 1998*, pages 92–99. IEEE, 1998.

[149] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454, 2014.

[150] Juan Manuel Pérez, Rafael Berlanga, María José Aramburu, and Torben Bach Pedersen. A relevance-extended multi-dimensional model for a data warehouse contextualized with documents. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 19–28. ACM, 2005.

[151] Guy Pierra. Context explication in conceptual ontologies: the plib approach. In *ISPE CE*, pages 243–253, 2003.

[152] Guy Pierra. Context representation in domain ontologies and its use for semantic integration of data. In *Journal on data semantics X*, pages 174–211. Springer, 2008.

[153] Guy Pierra, Dehainsala Hondjack, Yamine Ait Ameur, and Ladjel Bellatreche. Bases de données à base ontologique. principe et mise en oeuvre. *Ingénierie des systemes d'information*, 10(2):91–115, 2005.

[154] G Pierre. Merise: Modélisation de systèmes d'information., 2005.

[155] Francis Pisani and Dominique Piotet. *Comment le web change le monde: l'alchimie des multitudes*. Pearson Education France, 2008.

[156] Yoann Pitarch, Cécile Favre, Anne Laurent, and Pascal Poncelet. Context-aware generalization for cube measures. In *DOLAP 2010, ACM 13th International Workshop on Data Warehousing and OLAP, Toronto, Ontario, Canada, October 30, 2010, Proceedings*, pages 99–104, 2010.

[157] Yoann Pitarch, Cécile Favre, Anne Laurent, and Pascal Poncelet. Enhancing flexibility and expressivity of contextual hierarchies. In *FUZZ-IEEE 2012, IEEE International Conference on Fuzzy Systems, Brisbane, Australia, June 10-15, 2012, Proceedings.*, pages 1–8, 2012.

[158] Evaggelia Pitoura, Kostas Stefanidis, and Panos Vassiliadis. Contextual database preferences. *IEEE Data Eng. Bull.*, 34(2):19–26, 2011.

[159] Evaggelio Pitoura, Kostas Stefanidis, and AB Zaslavsky. Context in databases. *University of Ioannina, Greece*, pages 1–19, 2004.

[160] Paul Prekop and Mark Burnett. Activities, context and ubiquitous computing. *Computer Communications*, 26(11):1168–1176, 2003.

[161] Eric Prud'hommeaux. Algae rdf query language. *Online only*, 2004.

[162] Eric Prud'Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[163] Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca. A principled approach to context schema evolution in a data management perspective. *Information Systems*, 49:65–101, 2015.

[164] Vijayshankar Raman and Joseph M Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, volume 1, pages 381–390, 2001.

[165] Thomas Rebele, Fabian M. Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*, pages 177–185, 2016.

[166] Raymond Reiter. On closed world data bases. In *Logic and data bases*, pages 55–76. Springer, 1978.

[167] Dave Reynolds. Rdf-qbe: a semantic web building block. *rap. tech., Technical Report HPL-2002-327, HP Labs*, 2002.

[168] Stamatia Rizou, Kai Haussermann, Frank Durr, Nazario Cipriani, and Kurt Rothermel. A system for distributed context reasoning. In *Sixth International Conference on Autonomic and Autonomous Systems (ICAS)*, pages 84–89. IEEE, 2010.

[169] Jonathan Robie, Lars Marius Garshol, Steve Newcomb, M Fuchs, L Miller, D Brickley, V Christophides, and G Karvounarakis. The syntactic web: Syntax and semantics on the web. *Markup Languages: Theory and Practice*, 3(4):411–440, 2001.

[170] Tom Rodden, Keith Cheverst, K Davies, and Alan Dix. Exploiting context

in hci design for mobile systems. In *Workshop on human computer interaction with mobile devices*, pages 21–22, 1998.

[171] MM Roldan-Garcia, Ismael Navas-Delgado, and José F Aldana-Montes. A design methodology for semantic web database-based systems. In *Third International Conference on Information Technology and Applications (ICITA'05)*, volume 1, pages 233–237. IEEE, 2005.

[172] Oscar Romero and Alberto Abelló. Automating multidimensional design from ontologies. In *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pages 1–8. ACM, 2007.

[173] Oscar Romero and Alberto Abelló. A framework for multidimensional design of data warehouses from ontologies. *Data & Knowledge Engineering*, 69(11):1138–1157, 2010.

[174] Oscar Romero, Diego Calvanese, Alberto Abelló, and Mariano Rodríguez-Muro. Discovering functional dependencies for multidimensional design. In *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*, pages 1–8. ACM, 2009.

[175] Oscar Romero, Alkis Simitsis, and Alberto Abelló. Gem: requirement-driven generation of etl and multidimensional conceptual designs. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 80–95. Springer, 2011.

[176] Yannis Roussos, Yannis Stavrakas, and Vassia Pavlaki. Towards a context-aware relational model. In *the proceedings of the International Workshop on Context Representation and Reasoning (CRR 05)*, 2005.

[177] Nick Ryan, Jason Pascoe, and David Morse. Enhanced reality fieldwork: the context aware archaeological assistant. *Bar International Series*, 750:269–274, 1999.

[178] Aïcha BEN SALEM. *Qualité contextuelle des données : détection et nettoyage guidés par la sémantique des données*. PhD thesis, Université Paris 13, 2015.

[179] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *First Workshop on Mobile Computing Systems and Applications. WMCSA 1994*, pages 85–90. IEEE, 1994.

[180] Bill N Schilit and Marvin M Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, 1994.

[181] Andy Seaborne. Rdql-a query language for rdf. *W3C Member submission*, 9(29-21):33, 2004.

[182] Khouri Selma, Boukhari IlyèS, Bellatreche Ladjel, Sardet Eric, Jean Sté-Phane, and Baron Michael. Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. *Computers in Industry*, 63(8):799–812, 2012.

[183] Luciano Serafini and Andrei Tamilin. Local tableaux for reasoning in distributed description logics. In *Proc. of the International Workshop on Description*

Logics, DL, volume 4, pages 100–109, 2004.

[184] Alkis Simitisis, Panos Vassiliadis, Spiros Skiadopoulos, and Timos Sellis. Data. warehouse. refreshment. 2007.

[185] Alkis Simitisis, Panos Vassiliadis, Spiros Skiadopoulos, and Timos Sellis. *Data warehouse refreshment*. IRM Press, 2007.

[186] Alkis Simitsis, Dimitrios Skoutas, and Malú Castellanos. Representation of conceptual etl designs in natural language using semantic web technology. *Data & Knowledge Engineering*, 69(1):96–115, 2010.

[187] Michael Sintek and Stefan Decker. Triple - a query, inference, and transformation language for the semantic web. In *International Semantic Web Conference*, pages 364–378. Springer, 2002.

[188] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[189] Dimitrios Skoutas and Alkis Simitsis. Ontology-based conceptual design of etl processes for both structured and semi-structured data. *International Journal on Semantic Web and Information Systems*, 3(4):1–24, 2007.

[190] Dimitrios Skoutas, Alkis Simitsis, and Timos Sellis. Ontology-driven conceptual design of etl processes using graph transformations. *Journal on Data Semantics XIII*, pages 120–146, 2009.

[191] D Steer. Treehugger 1.0 introduction. *Online only*, 2003.

[192] Kostas Stefanidis, Georgia Koutrika, and Evaggelia Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Transactions on Database Systems (TODS)*, 36(3):19, 2011.

[193] Kilian Stoffel, Merwyn Taylor, and James Hendler. Efficient management of very large ontologies. In *AAAI/IAAI*, pages 442–447, 1997.

[194] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004-The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[195] Vijayan Sugumaran and Veda C. Storey. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Trans. Database Syst.*, 31(3):1064–1094, September 2006.

[196] Yannis Theoharis, Vassilis Christophides, and Grigoris Karvounarakis. Benchmarking database representations of rdf/s stores. In *The Semantic Web–ISWC 2005*, pages 685–701. Springer, 2005.

[197] K Tolle and F Wleklinski. easy rdf query language (erql). *Online only*, 2004.

[198] Dragan Đuric, Dragan Gaševic, and Vladan Devedžic. A mda-based approach to the ontology definition metamodel. In *Proceedings of the 4TH Workshop on Computational Intelligence and Information Technologies, Niš, Serbia*. Citeseer, 2003.

[199] Arthur H Van Bunningen, Ling Feng, and Peter MG Apers. Context for ubiquitous data management. In *International Workshop on Ubiquitous Data Management, 2005. UDM 2005.*, pages 17–24. IEEE, 2005.

[200] Panos Vassiliadis, Alkis Simitsis, and Eftychia Baikousi. A taxonomy of etl activities. In *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*, pages 25–32. ACM, 2009.

[201] Panos Vassiliadis, Alkis Simitsis, Panos Georgantas, Manolis Terrovitis, and Spiros Skiadopoulos. A generic and customizable framework for the design of etl scenarios. *Information Systems*, 30(7):492–525, 2005.

[202] Holger Wache, Th Scholz, Helge Stieghahn, and Brigitta Konig-Ries. An integration method for the specification of rule-oriented mediators. In *International Symposium on Database Applications in Non-Traditional Environments, 1999.(DANTE'99)*, pages 109–112. IEEE, 1999.

[203] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004.*, pages 18–22. IEEE, 2004.

[204] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *Personal Communications, IEEE*, 4(5):42–47, 1997.

[205] Stuart Weibel. The dublin core: a simple content description model for electronic resources. *Bulletin of the American Society for Information Science and Technology*, 24(1):9–11, 1997.

[206] Kevin Wilkinson, Craig Sayers, Harumi Kuno, and Dave Reynolds. Efficient rdf storage and retrieval in jena2. In *Proceedings of the First International Conference on Semantic Web and Databases*, pages 120–139. CEUR-WS. org, 2003.

[207] David Wood, Paul Gearon, and Tom Adams. Kowari: A platform for semantic web storage and analysis. In *XTech 2005 Conference*, pages 05–0402. Citeseer, 2005.

[208] Chang Xu and Shing-Chi Cheung. Inconsistency detection and resolution for context-aware middleware support. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 336–345. ACM, 2005.

[209] T. Sellis Y. Roussos. A model for context aware relational database. Technical report, National Technical University of Athens, 2008.

[210] Bazhar Youness. *Handling Behavioral Semantics in Persistent Meta-Modeling Systems*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2013.

[211] Daniel C Zilio, Jun Rao, Sam Lightstone, Guy Lohman, Adam Storm, Christian Garcia-Arellano, and Scott Fadden. Db2 design advisor: integrated automatic physical database design. In *Proceedings of the Thirtieth international conference on Very large data bases*, pages 1087–1097. VLDB Endowment, 2004.

Annexe A :

Cette annexe illustre la représentation UML de notre modèle de contexte.

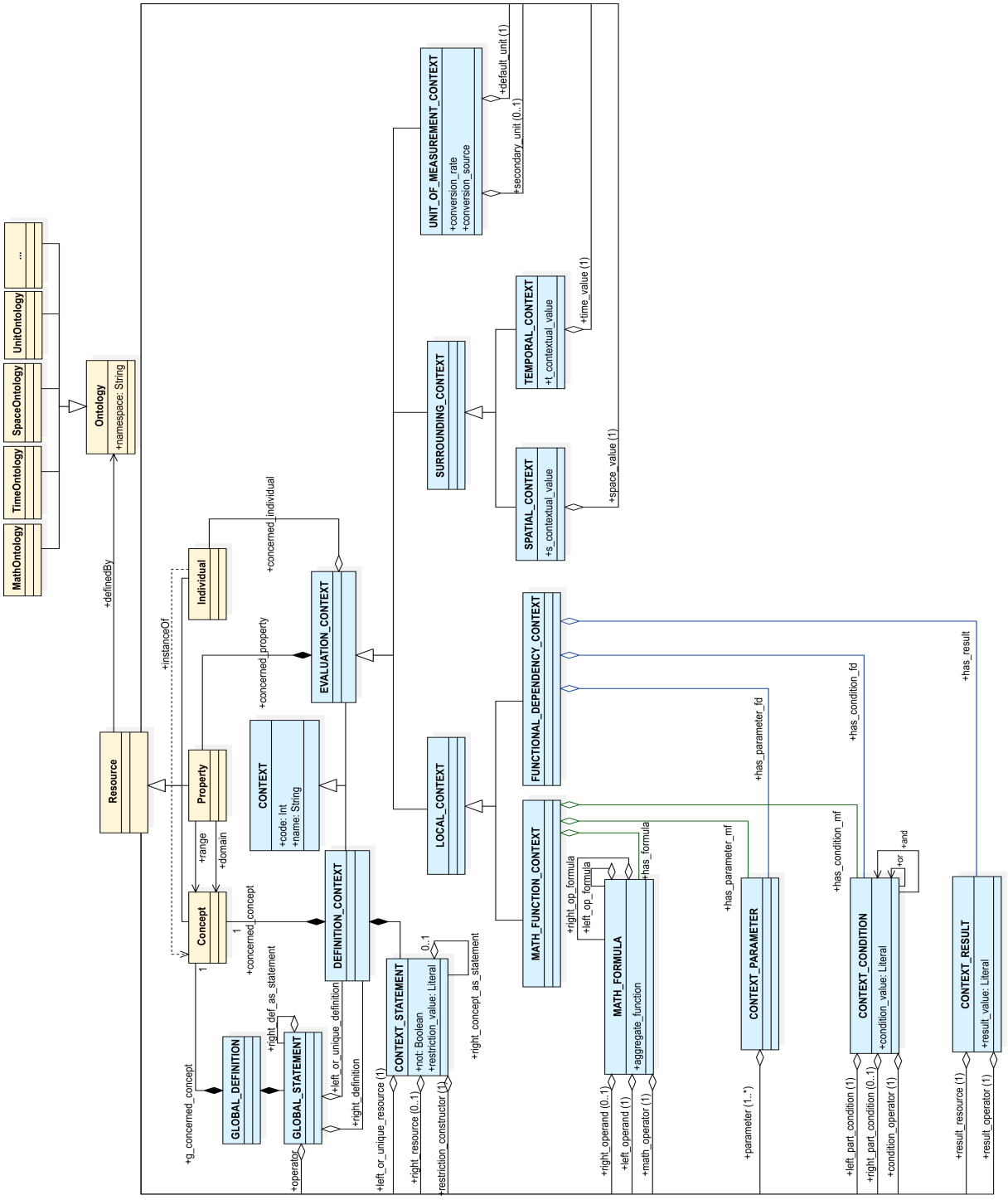


FIGURE 1 – Représentation UML de notre modèle générique de contexte

Annexe B :

Cette annexe présente les requêtes *OntoQL* correspondant à la création et au peuplement de l'ensemble des classes décrivant le fragment de l'ontologie *Healthcare* que nous avons utilisé tout au long de ce mémoire.

Requête$_1$: création de la classe DRG

```
CREATE #CLASS DRG (
PROPERTIES (codeDRG Int, labelDRG STRING));
CREATE EXTENT OF DRG (codeDRG, labelDRG);
```

Requête$_2$: création de la classe ICD

```
CREATE #CLASS ICD (
PROPERTIES (codeICD Int, labelICD STRING));
CREATE EXTENT OF ICD (codeICD, labelICD);
```

Requête$_3$: création de la classe EntryMode

```
CREATE #CLASS EntryMode (
PROPERTIES (labelEntryMode STRING));
CREATE EXTENT OF EntryMode (labelEntryMode);
```

Requête$_4$: création de la classe Time

```
CREATE #CLASS Time (
PROPERTIES (labelTime STRING));
CREATE EXTENT OF Time (labelTime);
```

Requête$_5$: création de la classe MedicalProcedure

```
CREATE #CLASS MedicalProcedure (
PROPERTIES (nameprocedure STRING, price REAL));
CREATE EXTENT OF MedicalProcedure (nameprocedure, price);
```

Requête$_6$: création de la classe HealthcareInstitution

```
CREATE #CLASS HealthcareInstitution (
PROPERTIES (nameInstitution STRING));
CREATE EXTENT OF HealthcareInstitution (nameInstitution);
```

Requête$_7$: création de la classe Person

```
CREATE #CLASS Person (
PROPERTIES (name STRING, age INTEGER, ageCategory STRING));
CREATE EXTENT OF Person (name, age, ageCategory);
```

Requête$_8$: création de la classe Patient

```
CREATE #CLASS Patient (
PROPERTIES (name STRING, age INTEGER,
ageCategory Enum('Infant','Adult','Senior')));
CREATE EXTENT OF Patient (name, age, ageCategory);
```

Requête$_9$: création de la classe MedicalStaff

```
CREATE #CLASS MedicalStaff (
PROPERTIES (idStaff INTEGER, worksFor REF (HealthcareInstitution)));
CREATE EXTENT OF MedicalStaff (idStaff, worksFor);
```

Requête$_{10}$: création de la classe Doctor

```
CREATE #CLASS Doctor (
PROPERTIES (idDoctor INTEGER));
CREATE EXTENT OF Doctor (idDoctor);
```

Requête$_{11}$: création de la classe Nurse

```
CREATE #CLASS Nurse (
PROPERTIES (idNurse INTEGER));
CREATE EXTENT OF Nurse (idNurse);
```

Requête$_{12}$: création de la classe MedicalMonitoring

```
CREATE #CLASS MedicalMonitoring (
PROPERTIES (heartRate REAL, respirationRate REAL, bloodPressure REAL,
BPCtegory Enum('Low','Normal','High'),hasMonitoringTime REF (Time),
hasProcedure REF (Procedure)));
CREATE EXTENT OF MedicalMonitoring (heartRate, respirationRate,
bloodPressure, BPCategory, hasMonitoringTime, hasProcedure);
```

Requête$_{13}$: création de la classe Ward

```
CREATE #CLASS Ward (
PROPERTIES (nameWard STRING, belngongsTo REF (HealthcareInstitution)));
CREATE EXTENT OF Ward (nameWard, belongsTo);
```

Requête$_{14}$: création de la classe Sojourn

```
CREATE #CLASS Sojourn (
PROPERTIES (bedsUtilization REAL, totalImpatientDaysOfCare INTEGER,
totalAdmission INTEGER, totalDischargeDays INTEGER,
totalDischarge INTEGER, averageLengthOfStay REAL, groupedBy REF (DRG),
codedBy REF (ICD), hasEntryMode REF (EntryMode),
hasExitMode REF (ExitMode), hasTime REF (Time), concerns REF (Patient)));
CREATE EXTENT OF Sojourn (bedsUtilization, totalImpatientDaysOfCare,
totalAdmission, totalDischargeDays, totalDischarge, averageLengthOfStay,
groupedBy, codedBy, hasEntryMode, hasExitMode, hasTime, concerns);
```

Annexe C :

Cette annexe présente les requêtes *OntoQL* correspondant à l'insertion des informations contextuelles suivant les exemples décrits dans le Chapitre 5.

Exemple 1 : Information contextuelle de type Dépendance Fonctionnelle

```
ageCategory_Rid = SELECT #rid FROM #property WHERE #code = 'ageCategory';
smoker_Rid = SELECT #rid FROM #property WHERE #code = 'smoker';
bloodPressure_Rid = SELECT #rid FROM #property WHERE #code =
'bloodPressure';
BPCategory_Rid = SELECT #rid FROM #property WHERE #code = 'BPCategory';

INSERT INTO #Context_Parameter (#code, #parameter) VALUES
('Context_parameter_1',array [" + ageCategory_Rid + ", "+smoker_Rid+",
"+bloodPressure_Rid+"]);

Context_parameter_1_Rid = SELECT #rid FROM #Context_Parameter WHERE
#code = 'Context_parameter_1';

INSERT INTO #Context_Condition (#code, #left_part_condition,
#right_part_condition, #condition_operator, #condition_value,
#and_condition, #or_condition) VALUES ('Context_condition_1',
"+blood_pressure_Rid+", Null, '>', 14, Null, Null);

Context_condition_1_Rid = SELECT #rid FROM #Context_Condition WHERE
#code = 'Context_condition_1';

INSERT INTO #Context_Condition (#code, #left_part_condition,
#right_part_condition, #condition_operator, #condition_value,
#and_condition, #or_condition) VALUES ('Context_condition_2',
"+smoker_Rid+", Null, '=', True, Null, Null);

Context_condition_2_Rid = SELECT #rid FROM #Context_Condition WHERE
#code = 'Context_condition_2';

INSERT INTO #Context_Condition (#code, #left_part_condition,
#right_part_condition, #condition_operator, #condition_value,
#and_condition, #or_condition) VALUES ('Context_condition_3',
"+ageCategory_Rid+", Null, '=', 'Adult', array [" +
Context_condition_1_Rid + ", "+Context_condition_2_Rid+"], Null);

Context_condition_3_Rid = SELECT #rid FROM #Context_Condition WHERE
#code = 'Context_condition_3';
```

```
INSERT INTO #Context_Result (#code, #result_operator, #result_value)
VALUES ('Context_result_1', '=', 'High');

Context_result_1_Rid = SELECT #rid FROM #Context_Result WHERE
#code = 'Context_result_1';

INSERT INTO #Functional_Dependency_Context (#code, #name,
#concerned_property, #concerned_individual, #has_parameter_fd,
#has_condition_fd, #has_result) VALUES ('100', 'HighBP',
"+BPCategory_Rid+", $, "+Context_parameter_1_Rid+",
"+Context_condition_3_Rid+", "+Context_result_1_Rid+");
```

Exemple 2 : Information contextuelle de type Fonction Mathématique

```
nameWard_Rid = SELECT #rid FROM #property WHERE #code = 'nameWard';
TDD_Rid = SELECT #rid FROM #property WHERE #code = 'totalDischargeDays';
TD_Rid = SELECT #rid FROM #property WHERE #code = 'totalDischarge';
ALOS_Rid = SELECT #rid FROM #property WHERE #code = 'averageLengthOfStay';

INSERT INTO #Context_Parameter (#code, #parameter) VALUES
('Context_parameter_2',array [" + nameWard_Rid + ", "+TDD_Rid+",
"+TD_Rid+"]);

Context_parameter_2_Rid = SELECT #rid FROM #Context_Parameter WHERE
#code = 'Context_parameter_2';

INSERT INTO #Context_Condition (#code, #left_part_condition,
#right_part_condition, #condition_operator, #condition_value,
#and_condition, #or_condition) VALUES ('Context_condition_4',
"+nameWard_Rid+", 'equal', 'Psychiatry');

Context_condition_4_Rid = SELECT #rid FROM #Context_Condition WHERE
#code = 'Context_condition_4';

INSERT INTO #Math_Formula (#code, #aggregate_function, #operator,
#left_operand, #right_operand, #right_operand_formula) VALUES
('Math_formula_1', $, '/', "+TDD_Rid+", "+TD_Rid+", $);

Math_formula_1_Rid = SELECT #rid FROM #Math_Formula WHERE
#code = 'Math_formula_1';

INSERT INTO #Math_Function_Context (#code, #name,
```

```
#concerned_property, #concerned_individual, #has_parameter_mf,
#has_condition_mf, #has_formula) VALUES ('200', 'LongStay',
"+ALOS_Rid+", $, "+Context_parameter_2_Rid+",
"+Context_condition_4_Rid+", "+Math_formula_1_Rid+");
```

Exemple 3 : Information contextuelle de type Contexte Temporel

```
price_Rid = SELECT #rid FROM #property WHERE #code = 'price';
consultation_Rid = SELECT #rid FROM #MedicalProcedure WHERE #code =
'General Practice Consultation';

INSERT INTO #Temporal_Context (#code, #name,
#concerned_property, #concerned_individual, #time_value,
#t_contextual_value) VALUES ('300', 'Tariff2013', "+price_Rid+",
"+consultation_Rid+", '2013', 20);
```

Exemple 4 : Information contextuelle de type Unités de Mesure

```
price_Rid = SELECT #rid FROM #property WHERE #code = 'price';

INSERT INTO #Unit_Of_Measurement_Context (#code, #name,
#concerned_property, #concerned_individual, #default_unit,
#target_unit, conversion_rate) VALUES ('400', 'USDollar', "+price_Rid+",
$, 'Euro', 'USDollar', 1.055765);
```

Table des figures

1.1 Données techniques : exemple d'une voiture 4
1.2 Cycle de vie de construction d'un entrepôt de données 6
1.3 Rôle des ontologies dans la construction et l'exploitation des systèmes de stockage de données 7
1.4 Différentes liaisons entre ontologie, contexte et base/entrepôt de données 8
1.5 Vue globale de la thèse : aperçu des contributions 9

2.1 Extrait de l'ontologie *Healthcare* 27
2.2 Représentation générique du fragment de l'ontologie *Healthcare* 28
2.3 Représentation spécifique du fragment de l'ontologie *Healthcare* 29
2.4 Représentation verticale des instances du fragment de l'ontologie *Healthcare* . 30
2.5 Représentation verticale avec ID des instances du fragment de l'ontologie *Healthcare* 30
2.6 Représentation binaire des instances du fragment de l'ontologie *Healthcare* . . 31
2.7 Représentation horizontale des instances du fragment de l'ontologie *Healthcare* 31
2.8 Architectures des Bases de Données Sémantiques 33
2.9 Cube de données à trois dimensions 36
2.10 Schémas en étoile, en flocon de neige et en constellation 37
2.11 Projection de l'ontologie sur les phases du cycle de vie d'un entrepôt de données 41

3.1 Exemple d'appariement C-OWL 57

4.1 Architecture à quatre niveaux de l'OMG (inspiré de [198]) 73
4.2 Hiérarchie des concepts de base d'ODM [198] 74

4.3 La représentation des définitions en EXPRESS-G 78
4.4 La représentation des relations en EXPRESS-G 79
4.5 Extrait de l'ontologie définissant le domaine de la santé 81
4.6 Représentation EXPRESS-G de notre modèle générique du contexte 85

5.1 Architecture quatre quarts d'OntoDB 102
5.2 Modèle d'ontologies noyau du langage *OntoQL*[99] 102
5.3 Architecture d'OntoDB étendue avec le modèle du contexte 106
5.4 Instanciation 1 : information contextuelle de type Dépendance Fonctionnelle . 116
5.5 Premier exemple d'instanciation de l'ontologie *Healthcare* 117
5.6 Deuxième exemple d'instanciation de l'ontologie *Healthcare* 118
5.7 Instanciation 2 : information contextuelle de type Fonction Mathématique . . . 119
5.8 Instanciation 3 : information contextuelle de type Contexte temporel 119
5.9 Troisième exemple d'instanciation de l'ontologie *Healthcare* 121
5.10 Instanciation 4 : information contextuelle de type Unités de Mesure 122

6.1 Modèle de besoin connecté au modèle d'ontologie [60] 133
6.2 Architecture générale de l'approche d'intégration proposée 136
6.3 Une itération de notre algorithme ETL 148
6.4 Architecture fonctionnelle de l'outil *CONTIC-DW Design Tool* 150
6.5 Modèle d'ontologie OWL enrichi avec le modèle de contexte (sous Protégé) . . 151
6.6 Définition des OLs comme des fragments de l'OP 152
6.7 Interface de l'identification et du chargement des sources de données 153
6.8 Interface du choix et du chargement de l'OP 153
6.9 Interface de définition des besoins 154
6.10 Interface de génération du modèle multidimensionnel 154
6.11 Interface de visualisation du modèle multidimensionnel 155
6.12 Définition des OLs comme des fragments de l'OP 155
6.13 Interface d'exécution du processus ETL 155
6.14 Intégration des contexte par Mapping 156
6.15 Résultats de l'exécution de la requête *R1* (Oracle 12c) 157

1 Représentation UML de notre modèle générique de contexte 183

Liste des tableaux

2.1 Constructeurs fournis par les modèles d'ontologies 25

3.1 Catégorisation du contexte 51
3.2 Comparaison entre les techniques de modélisation du contexte 54
3.3 Analyse des travaux proposant la prise en charge du contexte dans les systèmes de Bases de Données 61
3.4 Analyse des travaux intégrant le contexte dans les systèmes des Entrepôts de Données 64

4.1 Glossaire des termes médicaux 82
4.2 Extrait de la connaissance experte 83
4.3 Spécification en EXPRESS des entités 'Context ', 'Context Statement' 87
4.4 Spécification en EXPRESS des entités 'Global Statement'et 'Global Definition' 88
4.5 Spécification en EXPRESS de l'entité 'Context Parameter' 89
4.6 Spécification en EXPRESS de l'entité 'Context Condition' 90
4.7 Spécification en EXPRESS de l'entité 'Math Formula' 91
4.8 Spécification en EXPRESS de l'entité 'Math Function Context' 91
4.9 Spécification en EXPRESS de l'entité 'Context Result' 92
4.10 Spécification en EXPRESS de l'entité 'Functional Dependency Context' 92
4.11 Spécification en EXPRESS de l'entité 'Temporal Context' 93
4.12 Spécification en EXPRESS de l'entité 'Unit Of Measurement Context' 94

5.1 Création des entités correspondant à la catégorie "Contexte de définition" . . . 107
5.2 Création des entités correspondant à la catégorie "Fonction Mathématique " . . 108

5.3 Création des entités correspondant à la catégorie “Dépendance Fonctionnelle” . 108
5.4 Création de l’entité correspondant à la catégorie ‘Contexte Temporel’ 109
5.5 Création de l’entité correspondant à la catégorie ‘Contexte Spatial’ 109
5.6 Création de l’entité correspondant à la catégorie ‘Unité de mesure’ 109
5.7 Requêtes OntoQL génériques utilisant l’opérateur USING CONTEXT 112
5.8 Interprétation de la requête Q1 112
5.9 Interprétation de la requête Q2 113
5.10 Interprétation de la requête Q3 113
5.11 Interprétation de la requête Q3 114
5.12 Création des classes *Medical_Procedure* et *Medical_Monitoring* 115
5.13 Instanciation de la classe *Medical_Procedure* 115
5.14 Requête Q_{cas1} et son interprétation 116
5.15 Exécution de la requête Q_{cas1} 117
5.16 Requête Q_{cas2} et son interprétation 118
5.17 Exécution de la requête Q_{cas2} 120
5.18 Requête Q_{cas3} et son interprétation 120
5.19 Exécution de la requête Q_{cas3} 121
5.20 Requête Q_{cas4} et son interprétation 122
5.21 Exécution de la requête Q_{cas4} 123

6.1 Échantillon de situations contextuelles définies sur les sources et l’OP 152

Glossaire

ALOS : Average Lengh Of Stay
BDD : Base de Données
BDS : Base de Données Sémantique
BUR : Beds Utilization rate
C-OWL : Context Ontology Web Language
DAML : DARPA Agent Markup Language
DRG : Diagnosis Related Groups
DSA : Data Staging Area
ED : Entrepôt de Données
EDS : Entrepôt de Données Sémantique
EDSC : Entrepôt de Données Sémantique Contextuel
ETL : Extract-Transform-Load
GaV : Global as View
GlaV : Generalized Local as View
GPS : Global Positioning System
GUI : Globally Unique Identifier
HOLAP : Hybrid On-Line Analytical Processing
HR : Heart rate
ICD : International Classification Of Diseases
LAV : Local as View
LDD : Logique de Descriptions Distribuée
LIAS : Laboratoire d'Informatique et d'Automatique pour les Systèmes
MDA : Model Driven Architecture
MOF : Meta Object Facility
MOLAP : Multidimensional On-Line Analytical Processing
OC : Ontologie Conceptuelle
OCC : Ontologie Conceptuelle Canonique
OCNC : Ontologie Conceptuelle Non Canonique
ODM : Ontology Definition Meta-Model

OID : Object IDentifier
OIL : Ontology Inference Layer
OLAP : On-Line Analytical Processing
OL : Ontologie Linguistique
OMG : Object Management Group
OMR : Ontologie de multi-représentation
ORM : Object Role Modelling
OWL : Ontology Web Language
PLIB : Parts LIBrary
RDF : Ressource Description Framework
RDFS : Ressource Description Framework Schema
ROLAP : Relational On-Line Analytical Processing
RR : Respiration rate
SGBD : Système de Gestion de Bases de Données
SI : Système d'Information
SID : Systèmes d'Intégration de Données
SQL : Structured Query Language
TA : Total Admissions
TD : Total Discharge
TDD : Total Discharge Days
TIDC : Total Inpatient Days Of Care
UML : Unified Modelling Language
URI : Uniform Resource Identifier
URL : Uniform Resource Locator
W3C : World Wide Web Consortium
XML : eXtensible Markup Language

Résumé

Nous assistons à une époque où toute entreprise (ou organisme), dans le but d'augmenter son pouvoir décisionnel, est fortement intéressée par la collecte et l'analyse des données provenant de multiples sources hétérogènes et variées. Ces sources présentent également une autre spécificité à savoir la sensibilité au contexte. Cette situation nous met face à un enjeu scientifique crucial, du fait qu'elle réunit trois problématiques complémentaires : (i) la résolution de l'hétérogénéité qui peut exister entre les sources, (ii) la construction d'un système d'intégration décisionnel, et (iii) la prise en compte du contexte dans cette intégration. Afin de répondre à ces problématiques, nous nous intéressons dans cette thèse à la conception des applications contextuelles basées sur une ontologie de domaine, supposée existante. Pour ce faire, nous proposons d'abord un modèle de contexte qui intègre les dimensions principales identifiées dans la littérature. Une fois construit, il est lié au modèle de l'ontologie. Cette façon de procéder augmente la flexibilité dans la conception des applications avancées. Ensuite, nous proposons deux cas d'étude : (1) la contextualisation de sources de données sémantiques où nous étendons le système OntoBD/OntoQL afin qu'il prenne en compte le contexte, et (2) la conception d'un entrepôt de données contextuel où le modèle de contexte est projeté sur les différentes phases de conception du cycle de vie. Afin de valider notre proposition, nous présentons un prototype d'outil d'aide à la conception implémentant les différentes étapes de l'approche de conception proposée.

Mots clés : Contexte, Ontologie, Modélisation, Base de Données Sémantiques, Entrepôt de Données Sémantiques.

Abstract

We are witnessing an era when any company is strongly interested in collecting and analyzing data from heterogeneous and varied sources. These sources also have another specificity, namely context awareness. Three complementary problems are identified: the resolution of the heterogeneity of the sources, (ii) the construction of a decisional integrating system, and (iii) taking into account the context in this integration. To solve these problems, we are interested in this thesis in the design of contextual applications based on a domain ontology.

To do this, we first propose a context model that integrates the main dimensions identified in the literature. Once built, it is linked to the ontology model. This approach increases flexibility in the design of advanced applications. Then, we propose two case studies: (1) the contextualization of semantic data sources where we extend the OntoBD/OntoQL system to take the context into account, and (2) the design of a contextual data warehouse where the context model is projected on the different phases of the life cycle design. To validate our proposal, we present a tool implementing the different phases of the proposed design approach.

Key words : Context, Ontology, Modeling, Semantic Database, Semantic Data Warehouse.