La sémantique : une valeur ajoutée pour la conception et l'exploitation des données

Stéphane Jean

LIAS : Laboratoire d'Informatique et d'Automatique
pour les Systèmes
ISAE-ENSMA : Ecole Nationale Supérieure de
Mécanique et d'Aérotechnique
Université de Poitiers

Mémoire

pour l'obtention du diplôme

D'HABILITATION A DIRIGER DES RECHERCHES

(Faculté des Sciences Fondamentales et Appliquées)

(Diplôme National — Arrêté du 7 août 2006)

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

Présenté par :

Stéphane JEAN

****

La sémantique : une valeur ajoutée pour la conception et l'exploitation des données

****

Soutenu le 1er juillet 2021
devant la Comission d'Examen

****

JURY

Rapporteurs :	**Nadine CULLOT**	Professeure, Université de Bourgogne
	Mohand-Said HACID	Professeur, LIRIS / Université Claude Bernard Lyon 1
	Oscar PASTOR	Professeur, Université Polytechnique de Valence
Examinateurs :	**Yamine AIT-AMEUR**	Professeur, IRIT / INPT-ENSEEIHT
	Ladjel BELLATRECHE	Professeur, LIAS / ISAE-ENSMA
	Laure BERTI-EQUILLE	Directrice de Recherche, IRD
	Georg GOTTLOB	Professeur, Université d'Oxford
	Patrick VALDURIEZ	Directeur de Recherche, INRIA

Table des matières

Chapitre 1

Introduction

Nos travaux de recherche sont menés dans l'équipe *Ingénierie des Données et des moDèles (IDD)* du *Laboratoire d'Informatique et d'Automatique pour les Systèmes (LIAS)* qui fait partie de l'*École Nationale Supérieure de Mécanique et d'Aérotechnique (ISAE-ENSMA)*. Vu ce contexte, l'équipe IDD s'intéresse en particulier aux données manipulées dans des domaines de l'*ingénierie* tels que l'aéronautique, le transport, la mécanique ou l'énergie. Dans la plupart de ces domaines, les produits à concevoir sont des assemblages de composants. Une partie importante de la connaissance métier concerne ainsi les composants manipulés. Elle inclut en particulier les critères utilisés pour sélectionner des composants, leur comportement, les conditions de leur utilisation et la représentation pertinente de ces composants pour chaque discipline spécifique [1].

Prenons, comme exemple, un roulement à billes. Les propriétés essentielles d'un tel composant sont présentées dans la figure 1.1. Elles incluent des *propriétés caractéristiques* telles que la largeur ou les diamètres interne et externe dont les valeurs dépendent de l'unité de mesure utilisée. Le *comportement* de ce composant est également caractérisé par des propriétés comme la durée de vie qui définit la période de temps pendant laquelle le composant devrait fonctionner correctement. La valeur de cette propriété dépend de celles d'autres propriétés, définissant des *conditions d'utilisation* du composant, telles que

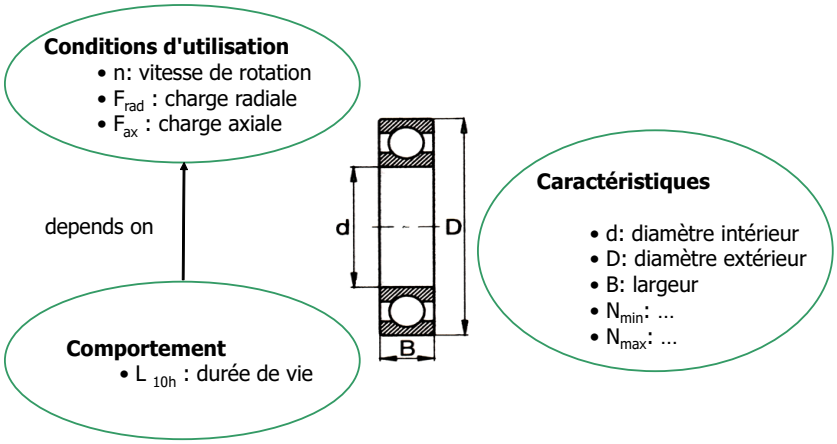


FIGURE 1.1 – Propriétés d'un roulement à billes

la vitesse de rotation ou les charges radiale et axiale. La figure 1.1 présente une représentation 2D d'un roulement à billes. Cependant, d'autres *points de vue* sur ce composant sont utilisés en ingénierie comme, par exemple, une représentation 3D ou schématique (voir figure 1.2).

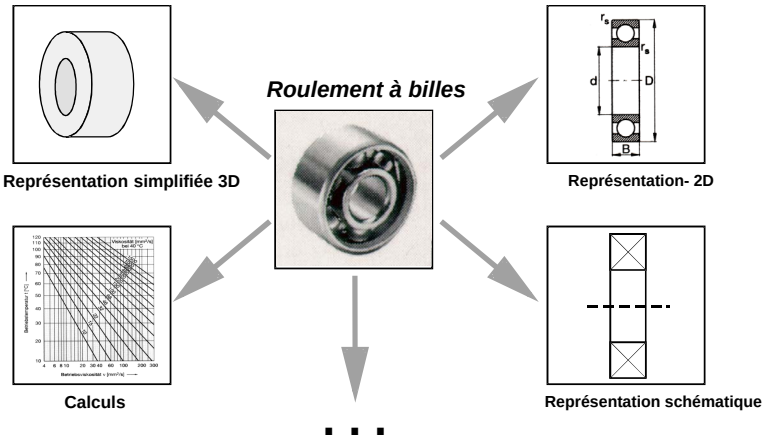


FIGURE 1.2 – Différents points de vue sur un roulement à billes

La gestion des données liées à des composants, comme des roulements à billes, vise à répondre à différents besoins tels que produire des catalogues de composants industriels, faciliter la sélection d'un composant répondant à des critères spécifiques, échanger des composants ou fournir une interface unique pour accéder à des composants produits par différents fournisseurs [1]. Ces besoins nécessitent de définir précisément la *connaissance métier* liée aux composants. Cela sous-entend, comme nous l'avons vu pour les roulements à billes, d'expliciter les unités de mesure, de définir des propriétés dont la valeur dépend d'autres propriétés et de donner plusieurs points de vue sur un même composant.

Suite à ce constat, l'équipe IDD s'est intéressée à la modélisation explicite de la sémantique des données sous la forme d'*ontologies* et ce, notamment pour les données produites dans des domaines de l'ingénierie. Ces travaux ont tout d'abord mené à la standardisation du *modèle d'ontologies PLIB (Parts LIBrary, ISO 13584)* [1] qui permet de définir formellement des ontologies ayant des caractéristiques spécifiques liées au contexte de l'ingénierie. Puis, les travaux de thèse d'Hondjack Dehainsala [2] ont proposé une *Base de Données (BD)* [1], nommée *OntoDB*, qui permet la persistance d'ontologies PLIB. Nous qualifions ce type de BD, spécialisées pour la persistance d'ontologies, de *Bases de Données Sémantiques (BDS)*.

Dans nos travaux de thèse, nous avons défini un langage d'exploitation, nommé *OntoQL*, pour une BDS telle qu'OntoDB. L'explicitation de la sémantique des données étant également une thématique étudiée dans le contexte du *Web sémantique* [3], nous avons étudié les liens entre le modèle d'ontologies PLIB et ceux conçus dans ce contexte. Cela nous a permis de développer le langage OntoQL de manière à ce qu'il ne soit pas spécifique au modèle PLIB et, qu'au contraire, il puisse s'adapter, autant que possible, à d'autres modèles. Nos travaux sur la conception de la *plateforme OntoDB/OntoQL* sont le point de départ des activités présentées dans ce mémoire. Dans leur prolongement, nous nous sommes

1. Il conviendrait ici d'utiliser le terme *système de gestion de base de données* au lieu de *base de données*. Pour simplifier, nous utiliserons uniquement ce dernier en faisant en sorte que le contexte de la phrase explicite la notion désignée.

intéressés à la conception, à l'exploitation et à la généralisation de ce nouveau type de BD. Dans ce qui suit, nous introduisons et définissons les notions importantes liées à ces travaux.

1.1 Sémantique implicite des données

Une *information* est une connaissance sur un fait, un procédé ou une personne. La représentation de cette information sous forme numérique conduit à la production d'une *donnée*. Celle-ci peut être de différentes natures telles qu'une ligne d'une table dans une *Base de Données Relationnelles (BDR)* ou du texte sur une page Web. La structure d'une donnée peut être définie via un *modèle de données*.

Comme exemple, considérons la ligne $\langle Paris, 17, 11, 84, 11\rangle$ appartenant à une table d'une BDR utilisée par une application météorologique. Son modèle de données est le schéma de sa table qui comprend des colonnes indiquant la ville concernée, la température qu'il y fait avec les précipitations, l'humidité et la vitesse du vent. L'application utilisant ces données indique la météo d'une ville sélectionnée.

Lorsqu'une donnée est produite, la connaissance associée à l'information représentée est souvent partiellement définie et basée sur des règles d'interprétation externes. Cela vient des faits suivants.

- Une donnée est souvent produite dans le contexte d'une application. Ainsi, elle ne représente que ce qui est nécessaire pour le fonctionnement de cette application. Dans l'exemple précédent, seul le nom de la ville est indiqué. Une ville a pourtant de nombreuses autres caractéristiques telles que son nombre d'habitants, son code postal ou son département. Ces informations n'étant pas pertinentes pour l'application météorologique, elles ne sont pas représentées dans la BDR.
- Une donnée est produite par des concepteurs qui ont en tête un contexte pour son utilisation qui n'est généralement pas explicité. Dans l'exemple précédent, la date à laquelle les conditions météorologiques ont été enregistrées n'est pas spécifiée. Vu que l'application météorologique indique la météo actuelle d'une ville, la BDR enregistre probablement les dernières conditions météorologiques connues pour la ville spécifiée. Par ailleurs, si cette application est faite pour des utilisateurs français, la température est certainement exprimée en degrés Celsius, les précipitations et l'humidité en pourcentage et la vitesse du vent en km/h, même si ce n'est pas explicité.

1.2 Hétérogénéité

Comme nous venons de le voir, les données et leurs modèles sont souvent conçus pour répondre à un besoin particulier, ce qui fait que leur *sémantique* est partiellement définie. En conséquence, le problème d'*hétérogénéité* se pose : des données ou modèles de données conçus sur le même domaine présentent de nombreuses différences. Cette hétérogénéité est de différentes natures.

- *Hétérogénéité des formalismes de modélisation* : de nombreux *formalismes* existent pour modéliser des données. Le choix d'un formalisme particulier dépend de nombreux facteurs tels que la préférence du concepteur, l'adéquation aux données à représenter ou la facilité d'utilisation. En conséquence, des *sources de données* (par exemple, des BD) sur le même domaine peuvent utiliser des formalismes différents.
- *Hétérogénéité structurelle des données* : des données représentant la même information peuvent

être définies avec des caractéristiques différentes. La raison principale est que les données sont généralement définies pour un applicatif spécifique, nécessitant une description particulière des données. Cela peut aussi être le résultat de choix techniques. Par exemple, une donnée peut être plus ou moins décomposée dans une BDR selon le degré de normalisation choisi.

– *Hétérogénéité sémantique des données* : les objets du monde réel peuvent être sujets à différentes perceptions. Ainsi, des données représentant la même information peuvent être définies avec différents noms, unités de mesure et de monnaie, règles d'interprétation, etc. Cela vient du fait que les sources de données sont conçues indépendamment les unes des autres par des concepteurs ayant des objectifs applicatifs différents. Ainsi, un même concept peut être représenté selon des points de vue différents.

Cette hétérogénéité est problématique lorsque les données, conçues dans un contexte particulier, sont utilisées pour des besoins non prévus par leurs concepteurs. Par exemple, les utilisateurs des réseaux sociaux produisent de nombreuses données visant essentiellement à permettre une interaction sociale entre individus. Ces données sont pourtant aujourd'hui utilisées pour de nombreux autres besoins tels que la détection d'évènements (par exemple, des catastrophes naturelles ou des actes terroristes), la publicité ou la détection d'opinions sur des produits. Utiliser une donnée dans un autre contexte que celui pour lequel elle a été produite pose de nombreux problèmes bien connus.

– *L'intégration de données* : elle vise à fournir une interface unique, uniforme et transparente aux données provenant de différentes sources.
– *L'échange de données* : il nécessite un format d'échange de données permettant de définir une vue commune des informations qui doivent être partagées.

Pour traiter ces problèmes, il est nécessaire de résoudre les conflits liés à l'hétérogénéité des données, que nous avons évoqués précédemment. Cela implique d'*expliciter* la sémantique des données manipulées. Nous présentons dans ce qui suit une vue d'ensemble des approches proposées pour cela en mettant l'accent sur la solution choisie dans nos travaux, à savoir l'utilisation des ontologies.

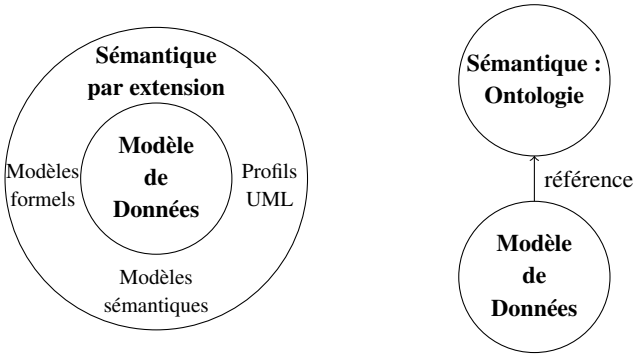
1.3 Explicitation de la sémantique des données

Comme l'illustre la figure 1.3, deux principales approches ont été suivies pour expliciter la sémantique des données : l'extension des modèles de données et la modélisation ontologique. Ces approches sont décrites dans ce qui suit.

1.3.1 Extension des modèles de données

La première approche, illustrée sur la figure 1.3 (a), consiste à étendre les modèles de données pour définir plus précisément la sémantique des données modélisées. Cette approche a été suivie dans différents domaines.

– *Les BD* : dans les années 90, de nombreuses propositions ont été faites pour définir des langages de modélisation de données plus expressifs que le modèle entité-association et relationnel. Ces propositions connues sous le nom de *modèles sémantiques (semantic models)* [4] ont pour but de modéliser plus précisément les données et leurs relations. Dans ces modèles, de nombreux



(a) Extension des modèles de données (b) Modélisation ontologique

FIGURE 1.3 – Deux principales approches d'explicitation de la sémantique des données

constructeurs ont été ajoutés par rapport à ceux du modèle entité-association. Ils visent principalement à permettre une plus grande abstraction des données en s'appuyant sur des concepts orientés-objets (par exemple, l'héritage), à définir plus précisément les relations entre les données, à supporter les informations temporelles et à définir la *sémantique comportementale* des données via des opérations. Cette modélisation peut ensuite être traduite en un modèle d'implémentation, aussi appelé *modèle logique*, qui est généralement un modèle relationnel.

Ce processus fait que la sémantique exprimée par le modèle initial, qualifié de *modèle conceptuel*, est perdue puisque seul le modèle logique est utilisé par les applications. Celui-ci résultant de la normalisation et de l'adaptation au système support est, en général, différent du modèle conceptuel. Ainsi, les propositions de *modèles sémantiques* ont permis de définir des modèles conceptuels plus expressifs. Mais, l'absence de représentation de ces modèles dans les BD fait que l'interprétation des données modélisées est toujours aussi difficile.

– *Les méthodes formelles* : les langages dits formels permettent une modélisation précise des données. C'est par exemple le cas du langage B [5] qui repose sur des machines abstraites et encode un système état-transition. Dans ce dernier, les variables représentent l'état et les opérations permettent la transition d'un état vers un autre. Ce langage se base sur une modélisation incrémentale en commençant par un modèle abstrait qui est peu à peu raffiné pour arriver au niveau de détail attendu dans l'application. Utiliser ce type de langage permet ainsi de définir précisément la sémantique des données. Cependant, cette approche souffre des mêmes inconvénients que les modèles sémantiques, dans le sens où ils permettent une modélisation conceptuelle plus riche des données, mais pas des données représentées au sein d'une BD.
– *Le génie logiciel* : le langage UML est une référence pour la modélisation de données et d'applicatifs. Ce langage contient un mécanisme d'extension via des *profils UML*. Par exemple, il est possible de créer un profil UML pour associer une unité de mesure à une propriété et, ainsi, définir plus précisément la sémantique des données modélisées. Cependant, ce profil est *ad-hoc* et non standardisé. Les outils supportant le langage UML ne sont donc pas conçus pour interpréter un tel profil. Cette interprétation demanderait de modifier et de recompiler ces outils.

Comme nous venons de le voir, augmenter l'expressivité des modèles de données pour expliciter la sémantique des données présente plusieurs inconvénients lorsque celles-ci sont stockées en BD : (i) le

modèle de données implémenté est différent du modèle conceptuel qui n'est pas conservé dans la BD, (ii) le modèle de données se complexifie, ce qui rend plus difficile son appréhension par les experts du domaine ainsi que sa transformation en un modèle d'implémentation, (iii) cette approche modélise la sémantique du domaine dans le modèle conceptuel alors que celle-ci pourrait être réutilisée dans d'autres applicatifs et (iv) les modèles ainsi obtenus ne sont pas forcément consensuels, ce qui rend difficile leur interprétation dans différents applicatifs.

1.3.2 Modélisation de la sémantique du domaine

Pour répondre aux limitations précédentes, une autre approche, illustrée sur la figure 1.3 (b), a été proposée. Elle consiste à modéliser la sémantique d'un domaine sous la forme d'une *ontologie* (ou *base de connaissances*). Cette notion a été initialement définie par Gruber comme « une spécification explicite d'une conceptualisation » [6]. De notre point de vue, une ontologie est « un dictionnaire formel et consensuel des catégories et propriétés des entités d'un domaine ainsi que des relations qui les unissent » [7]. Cette définition met en avant trois principales caractéristiques d'une ontologie.

- *Formelle* : une ontologie définit un ensemble de concepts sur un domaine donné. Ces définitions reposent sur un *modèle d'ontologies* qui propose différents constructeurs tels que ceux de classes et de propriétés. Défini formellement, il permet de vérifier la consistance de l'ontologie conçue et de réaliser des raisonnements automatiques sur ses concepts.
- *Consensuelle* : une ontologie est acceptée et partagée par une communauté. Ainsi, contrairement à un modèle conceptuel, une ontologie n'est pas conçue pour une application spécifique. Elle décrit les connaissances d'un domaine pour satisfaire les besoins des membres d'une communauté.
- *Identification universelle* : chaque concept d'une ontologie a un identifiant universel (par exemple, une *URI, Uniform Resource Identifier)*. Il permet de référencer un concept d'une ontologie et la sémantique qu'il représente à partir de n'importe quel environnement.

Deux catégories d'ontologies sont distinguées [1] : (i) les *ontologies de haut niveau* qui définissent des concepts généraux et (ii) des ontologies plus spécifiques dites *de domaine* qui sont liées à un univers du discours particulier. Elles décrivent et représentent la connaissance existant dans le domaine correspondant à cet univers. Dans nos travaux, nous nous sommes focalisés sur les ontologies de domaine. Dans ce mémoire, le mot ontologie sera utilisé pour désigner une ontologie de domaine.

Comme autre typologie [1, 8], nous distinguons les *Ontologies Linguistiques (OL)*, dont l'objectif est la représentation de la signification des termes utilisés dans un univers du discours particulier pour une langue naturelle donnée, des *Ontologies Conceptuelles (OC)*, dont le but est la représentation de catégories et propriétés d'objets présents dans un domaine particulier. Dans une OC, deux types de concepts existent [9].

- *Les concepts primitifs* « pour lesquels nous ne sommes pas capables de donner une définition axiomatique complète » [9]. La définition de ces concepts s'appuie sur une documentation textuelle et un savoir partagé avec les utilisateurs.
- *Les concepts définis* « pour lesquels une ontologie fournit une définition axiomatique complète au moyen de conditions nécessaires et suffisantes exprimées en termes d'autres concepts primitifs ou eux-mêmes définis » [9]. Ces définitions introduisent ainsi un nouveau concept pour désigner

quelque chose qui est déjà défini par un autre moyen dans l'ontologie en cours de conception.

Cette distinction amène à diviser la catégorie des OC en deux sous-catégories [7] : (i) les *Ontologies Conceptuelles Canoniques (OCC)* qui ne contiennent que des concepts primitifs. Elles définissent ainsi un vocabulaire canonique dans lequel chaque information du domaine d'étude est définie de manière unique et (ii) les *Ontologies Conceptuelles Non Canoniques (OCNC)* qui incluent des concepts primitifs et définis. Elles introduisent ainsi de nouvelles capacités de raisonnement et des opérateurs d'équivalence qui peuvent être utilisés pour définir des correspondances entre ontologies.

Les trois catégories d'ontologies introduites précédemment peuvent être liées dans un modèle en couches illustré dans la figure 1.4 [7]. Dans ce modèle, le noyau est constitué d'une OCC qui fournit les définitions nécessaires à la représentation et à l'échange de connaissances sur un domaine d'étude. Une couche OCNC étend le vocabulaire canonique avec des équivalences de concepts (par exemple, des expressions de classe venant de la logique de description) afin de définir tous les concepts utilisés dans le domaine. Enfin, une couche OL définit les termes associés aux concepts OCC et OCNC éventuellement dans différentes langues naturelles.

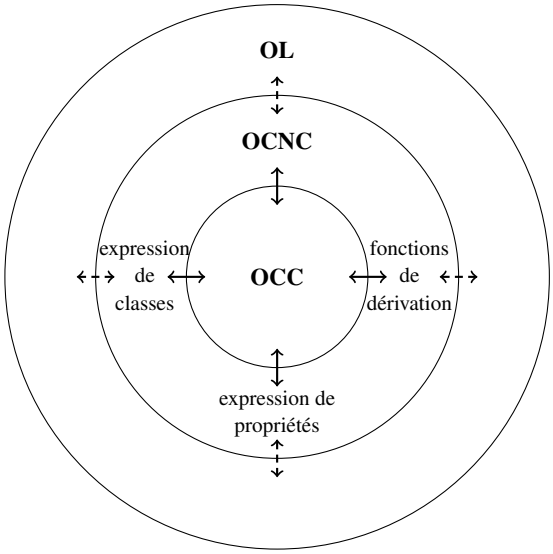


FIGURE 1.4 – Modèle en couches pour les ontologies de domaine

Les ontologies sont utilisées pour de nombreux problèmes de recherche et en particulier pour l'intégration de données qui est une problématique importante en ingénierie.

1.3.3 Intégration de données basées sur des ontologies et hypothèses

Lors de l'intégration de sources de données, les ontologies peuvent servir de référentiels communs. Pour que les sources de données puissent avoir une certaine autonomie, elles peuvent se baser sur des ontologies dites *locales*. Celles-ci sont généralement construites à partir d'une ontologie dite *globale*, si une telle ontologie existe au moment de la création des sources de données. On parle alors d'*intégration*

à priori [10]. Dans le cas contraire, une réconciliation sémantique doit être effectuée entre les ontologies locales et l'ontologie globale. On parle d'*intégration à postériori* [10].

Dans les scénarios d'intégration évoqués précédemment, le domaine d'application joue un rôle important. En effet, dans des domaines d'étude *contrôlés*, des ontologies standardisées existent et un certain degré de complétude et d'exactitude peut être garanti. C'est par exemple le cas de domaines de l'ingénierie, pour lesquels de nombreuses ontologies standardisées existent comme *Electronic Components* (IEC 61360-4), *Machining Tools* (ISO 13399) ou *Mechanical Fasteners* (ISO 13584-511). A l'opposé, dans des domaines d'étude plus ouverts, tels que le Web sémantique, de nombreuses ontologies couvrant le même domaine peuvent exister sans garantie de complétude. En conséquence, ces ontologies sont basées sur des hypothèses différentes de celles conçues dans des domaines de l'ingénierie.

- *Hypothèses du monde ouvert et du monde clos.* Sous l'hypothèse du monde clos (*Close World Assumption, CWA*), n'importe quel fait qui n'est pas connu comme étant vrai est considéré comme faux. Au contraire, sous l'hypothèse du monde ouvert (*Open World Assumption, OWA*), n'importe quel fait qui n'est pas connu peut être vrai. Ces hypothèses ont des conséquences importantes sur l'interprétation des contraintes exprimées dans une ontologie. En effet, sous l'hypothèse CWA, les contraintes sont vérifiées alors que sous OWA, elles sont utilisées pour faire de l'inférence. L'hypothèse OWA est souvent faite dans des domaines ouverts tels que le Web sémantique alors que CWA est plus adaptée dans des environnements contrôlés, comme ceux évoqués précédemment dans les domaines de l'ingénierie.
- *Hypothèse d'unicité des noms.* Sous l'hypothèse d'unicité des noms (*Unique Name Assumption, UNA*), si deux objets ont des identifiants différents, ils représentent nécessairement deux objets distincts. Cette hypothèse est souvent faite en complément de CWA dans les environnements contrôlés où un certain degré de complétude peut être garanti. À l'opposé, elle est rarement faite si OWA est considérée. En effet, sans l'hypothèse UNA, si deux objets ont des identifiants différents, un processus d'inférence peut déduire que ce sont, malgré tout, le même objet.
- *Hypothèses de typage*. Sous l'hypothèse OWA, la classification des instances (c'est-à-dire, la découverte des classes auxquelles appartient une instance) est une capacité de raisonnement importante. Ainsi, cette hypothèse est souvent associée à celle de *typage faible* (*Weak Typing Assumption, WTA*) : (i) une instance peut appartenir à plusieurs classes sans lien dans l'ontologie, (ii) une propriété peut être définie sans un domaine associé et (iii) une instance peut avoir une valeur pour n'importe quelle propriété de l'ontologie. Cette caractéristique offre une grande flexibilité pour la définition des instances qui peuvent être *structurées* (c'est-à-dire que toutes les instances d'une même classe ont tendance à avoir une valeur pour les mêmes propriétés) ou, au contraire, non structurées. Dans les domaines de l'ingénierie, la représentation de composants conduit généralement à des données structurées qui respectent une hypothèse de *typage fort* (*Strong Typing Assumption, STA*) : (i) chaque instance a une *classe de base* qui est celle la plus basse dans la hiérarchie parmi les classes auxquelles appartient l'instance, (ii) chaque propriété est définie sur une classe qui définit son contexte d'application et est associée à un codomaine et (iii) seules les propriétés applicables à une classe peuvent être utilisées pour décrire ses instances.

Comme nous le voyons dans cette section, les ontologies peuvent être utilisées dans différents contextes qui ne nécessitent pas les mêmes hypothèses. En conséquence, différents langages de définition d'ontologies, que nous appelons *modèle d'ontologies*, ont été proposés.

1.3.4 Modèles d'ontologies

Dans cette section, nous décrivons succinctement des modèles d'ontologies liés à nos travaux en précisant leurs domaines d'application et le type d'ontologies qu'ils permettent de définir.

RDF-Schema. *RDF-Schema* ou *RDFS* [11] est basé sur le langage *RDF (Resource Description Framework)*. Ce dernier est utilisé pour définir des assertions sous la forme de triplets *(sujet, prédicat, objet)*. Le *sujet* est une URI qui dénote une ressource, le *prédicat* est une propriété qui la caractérise et l'*objet* est la valeur de la propriété pour le sujet. Celle-ci peut être soit une valeur littérale, soit l'URI d'une autre ressource. RDFS étend RDF par un ensemble de constructeurs afin de pouvoir concevoir un *schéma d'ontologie*. Ainsi, il permet de définir des classes et propriétés ainsi que leurs hiérarchies (`subClassOf` et `subPropertyOf`). Il fournit aussi des constructeurs pour spécifier le domaine et codomaine des propriétés en s'appuyant sur des types de données. Enfin, il se base sur des triplets RDF pour la définition des instances des classes et de leurs valeurs de propriétés. Ainsi, RDFS ne fournit pas de constructeur de concepts définis. Il est donc orienté vers la construction d'OCC. Ce langage a été conçu dans le contexte du Web sémantique et fait donc l'hypothèse OWA mais pas UNA.

OWL. *OWL (Web Ontology Language)* [12] étend le pouvoir d'expression de RDFS. Trois versions de ce langage ont été définies : OWL Lite, OWL DL et OWL Full qui ont un pouvoir d'expression croissant (OWL Lite $\subset$ OWL DL $\subset$ OWL Full) mais, en contrepartie, des raisonnements de plus en plus coûteux. Ces langages introduisent des opérateurs pour définir des concepts définis et donc des ontologies OCNC. Ils permettent ainsi de créer une classe comme étant l'union, l'intersection ou le complément d'autres classes, c'est-à-dire en utilisant des *opérateurs ensemblistes*. Une classe peut également être une *restriction* d'une autre classe. Par exemple, la classe `Parent` peut être définie comme étant les instances de la classe `Personne` qui ont au moins une valeur pour la propriété `enfant`. Tout comme RDFS, OWL a été conçu dans le contexte du Web sémantique et fait donc l'hypothèse OWA mais pas UNA. OWL se focalise sur les ontologies OCNC et leurs capacités de raisonnement.

OWL Flight. Considérant que les hypothèses sous-jacentes au langage OWL ne sont pas adaptées à tous les domaines, De Bruijn *et al.* ont proposé le langage OWL Flight [13]. Ce dernier est basé sur un sous-ensemble du langage OWL DL. En plus de ces constructeurs, il introduit un système de typage plus élaboré que celui de OWL, des capacités de métamodélisation et la possibilité de définir des contraintes d'intégrité. Ce langage fait l'hypothèse UNA et les contraintes sont basées sur CWA.

PLIB. Le modèle d'ontologies PLIB [1] est un standard international (ISO 13584) défini initialement pour échanger et intégrer automatiquement des catalogues de composants industriels. Compte tenu de la complexité des concepts primitifs des domaines de l'ingénierie, le but de ce modèle est de définir de tels concepts de la manière la plus précise possible. Dans PLIB, chaque concept est identifié de manière universelle par un *BSU (Basic Semantic Unit)*. Deux opérateurs de subsomption sont disponibles pour hiérarchiser les classes : *is_a*, l'opérateur d'héritage simple classique, et *case_of* qui permet un héritage partiel des propriétés d'une super-classe. Pour définir précisément les concepts des domaines de l'ingénierie, PLIB permet de spécifier différents points de vue sur un même concept, d'utiliser des unités de mesure ou de monnaie et de définir des propriétés dont les valeurs dépendent d'autres propriétés. Il permet aussi d'attacher une description linguistique, composée de noms et synonymes éventuellement définis dans plusieurs langues naturelles, à chaque concept d'une ontologie. Même s'il permet de définir des propriétés calculées à l'aide de fonctions de dérivation, le modèle PLIB se focalise sur la définition

d'OCC pour les domaines de l'ingénierie. Il fait donc les hypothèses CWA et UNA qui sont généralement considérées comme les plus plausibles dans ces domaines [1].

Les modèles d'ontologies présentés précédemment sont comparés sur différents critères dans le tableau 1.1. Comme nous pouvons le voir, ces modèles diffèrent par les constructeurs fournis et les hypothèses faites selon les domaines ciblés. Ainsi, le modèle PLIB propose des constructeurs pour définir les informations techniques liées aux composants manipulés dans des domaines de l'ingénierie en faisant les hypothèses qui leur sont généralement associées. A l'opposé, le modèle OWL propose des constructeurs pour définir des OCNC et ainsi offrir des capacités de raisonnement adaptées à des domaines ouverts liés au contexte du Web sémantique. RDFS et OWL Flight présentent des caractéristiques intermédiaires entre ces deux modèles. RDFS permet la définition d'OCC pour le Web sémantique. OWL Flight se base sur OWL mais en faisant des hypothèses proches de celles du modèle PLIB et en proposant des constructeurs pour définir plus précisément les concepts primitifs des ontologies.

Critère	**RDFS**	**OWL**	**OWL Flight**	**PLIB**
Identifiant universel	URI	URI	URI	BSU
Subsomption	subClass/Prop.	subClass/Prop.	subClass/Prop.	is_a, case_of
Constructeurs OCC [2]			typage élaboré	unités, contexte, etc.
Constructeurs OCNC		restriction	restriction	fonctions de
		op. ensemblistes	op. ensemblistes	dérivation
Constructeurs OL	label/comment	label/comment	label/comment	nom/synonyme
Hypothèses	OWA,WTA	OWA,WTA	CWA [3],UNA,WTA	CWA,UNA,STA

TABLE 1.1 – Comparaison de plusieurs modèles d'ontologies

Dans nos travaux de recherche, que nous positionnons dans la section suivante, nous nous sommes intéressés à la persistance des ontologies qui peuvent être définies avec ces différents modèles d'ontologies. À la différence de nombreux travaux de l'état de l'art, notre objectif est de concevoir des solutions extensibles qui prennent en compte cette diversité de modèles d'ontologies. Ainsi, le constat que les ontologies peuvent avoir des caractéristiques différentes, qui viennent des modèles ayant permis de les définir et de leurs domaines d'application, est resté le fil conducteur des travaux présentés dans ce mémoire.

1.4 Positionnement de nos recherches

Nos travaux de recherche portent sur les *Bases de Données Sémantiques (BDS)* qui permettent la persistance d'ontologies. Ils sont organisés autour de trois axes de recherche qui sont introduits dans ce qui suit.

2. Ces modèles possédant tous les constructeurs de classes et propriétés, nous n'indiquons ici que ceux qui sont spécifiques.
3. Uniquement pour les contraintes.

1.4.1 Extension et généralisation des BDS

Les ontologies sont utilisées dans de nombreux domaines pour annoter des données ou modèles afin d'en expliciter la sémantique. Concernant la persistance de ces informations, la plupart des BDS ne permettent que le stockage d'ontologies définies selon un modèle spécifique. Ainsi, d'une part, elles ne considèrent pas le fait que différents modèles d'ontologies pourraient être utilisés. D'autre part, elles ne fournissent pas de solutions pour représenter également les données ou modèles annotés par ces ontologies. Stocker l'ensemble de ces informations dans le même environnement présenterait cependant l'avantage de bénéficier des capacités de passage à l'échelle de cet environnement et de pouvoir manipuler l'ensemble de ces informations avec le même langage de requêtes. Notre premier axe de recherche vise à répondre à cette problématique en proposant une BDS extensible.

Le point de départ de cet axe est la BDS OntoDB et son langage OntoQL, qui ont été développés au cours des thèses d'Hondjack Dehainsala [2] et de moi-même [7]. Dans le prolongement de ces travaux, nous avons étendu cette BDS afin de pouvoir persister et annoter différents types de modèles (par exemple, des modèles métiers ou de préférences utilisateur) par des ontologies. Cela nous a conduits à identifier une limitation de cette BDS qui est de ne permettre de définir que la *sémantique structurelle* d'un modèle et non pas sa *sémantique comportementale*. Nous l'avons donc généralisée pour qu'elle permette de définir cette sémantique de manière flexible, c'est-à-dire en faisant en sorte qu'elle puisse être codée dans différents environnements de programmation et qu'elle soit intégrée directement à la BDS sans besoin de redémarrage. Ainsi, les objectifs auxquels nous répondons dans cet axe de recherche sont les suivants :

- définir une BDS permettant de persister différents types de modèles et de les lier à des ontologies ;
- concevoir un langage d'exploitation des données et des modèles contenus dans cette BDS ;
- fournir des mécanismes permettant de définir la sémantique structurelle et comportementale des modèles persistés.

Les travaux menés dans cet axe de recherche nous ont ainsi conduits à concevoir une BDS originale qui permet d'expliciter la sémantique de données ou de modèles via des ontologies au sein d'un environnement persistant. Elle est construite en suivant l'architecture de métamodélisaiton du *MOF (Meta Object Facility)* [14], utilisée dans l'*Ingénierie Dirigée par les Modèles*, afin d'en faciliter l'extension. De nombreuses autres BDS ayant été proposées dans la littérature, avec des caractéristiques différentes de celles d'OntoDB, ce nouveau type de BD présente une forte diversité. Aussi, nous sommes intéressés au processus de conception des BDS avec comme objectif de prendre en compte cette caractéristique.

1.4.2 Ingénierie des BDS

Le développement des BD traditionnelles suit un processus composé de différentes étapes : (i) la collecte et l'analyse des besoins, (ii) la modélisation de la BD lors des phases conceptuelles et logiques, et (iii) l'implémentation de la BD et son optimisation lors de la phase physique. Les BDS étant un nouveau type de BD, elles apportent de nouvelles caractéristiques qui peuvent avoir des conséquences sur ce processus. En particulier, elle présente une diversité qu'il convient de prendre en compte. Aussi, nous avons étudié les impacts des caractéristiques des BDS sur le processus de conception traditionnel des BD.

Dans la phase de collecte et d'analyse des besoins, nous avons constaté que les applications basées sur des ontologies sont souvent définies par plusieurs partenaires qui définissent leurs besoins applicatifs avec différents formalismes et vocabulaires. Par ailleurs, ces besoins ne sont pas utilisés dans les autres phases du processus de conception des BD. Les BDS permettant l'explicitation de la sémantique via des ontologies, nous nous sommes appuyés sur celles-ci pour proposer une démarche permettant d'intégrer des besoins définis avec différents formalismes et vocabulaires. Elle permet également d'utiliser les besoins ainsi intégrés pour l'optimisation de la BDS en cours de conception.

Concernant les autres phases de ce processus, nous avons voulu prendre en compte la diversité des BDS. La conséquence de cette caractéristique est que différents choix sont possibles dans les phases de conception d'une BDS. Les méthodes actuelles n'explicitant pas ces choix et les liens qu'ils ont entre eux, nous avons proposé une démarche générale de conception des BDS qui s'appuie sur les méthodes et outils des *Lignes de Produits Logiciels*. Pour implémenter cette démarche, nous avons ensuite considéré le problème de sélection des vues matérialisées, qui est traité pendant la phase physique. Nous avons constaté que la plupart des approches traitant ce problème pour les BDS ne prenaient pas en compte leur diversité. En conséquence, nous avons proposé de nouvelles approches basées sur un modèle de coût *générique*, dans le sens où il peut s'appliquer à différentes BDS.

Ainsi, les objectifs auxquels nous répondons dans cet axe de recherche sont les suivants :

- permettre d'intégrer des besoins définis avec différents formalismes et vocabulaires et les utiliser dans d'autres phases du processus de conception des BDS ;
- proposer une démarche de conception d'une BDS qui explicite les différents choix possibles et les liens qu'ils ont entre eux ;
- définir des approches de sélection de vues matérialisées pour les BDS qui prennent en compte leur diversité.

Dans cet axe de recherche, nous nous sommes intéressés à la conception des BDS. Une fois celle-ci réalisée, l'exploitation de la BDS conçue est une activité essentielle qui implique fortement l'utilisateur. Nous nous sommes donc intéressés à des problèmes que peuvent rencontrer les utilisateurs lorsqu'ils interrogent une BDS.

1.4.3 Techniques coopératives pour les BDS

Quand un utilisateur exprime une requête sur une BD, il espère obtenir un résultat de qualité, c'est-à-dire qui soit utile pour trouver l'information recherchée. Ce n'est généralement pas le cas lorsqu'une réponse vide est retournée à l'utilisateur. Cette problématique, qualifiée de *problème des réponses vides*, a été largement étudiée dans différents contextes tels que les BDR ou les systèmes de recommandation. Ce problème est important dans le contexte des ontologies parce que celles-ci sont incomplètes et que leurs structures et contenus sont souvent méconnus de l'utilisateur [15]. Ainsi, une *requête sémantique* peut échouer pour des raisons habituelles (par exemple, parce que la requête est trop restrictive ou qu'elle est mal formulée) mais aussi pour d'autres raisons, propres au contexte des ontologies (par exemple, parce que le résultat attendu n'est pas dans l'ontologie interrogée ou qu'une propriété impliquée dans la requête utilisateur n'est pas utilisée pour décrire les instances de l'ontologie). Ces raisons font que le problème des réponses vides est fréquent [16] dans le contexte des ontologies.

Pour répondre à ce problème, des approches ont été proposées pour fournir des *réponses coopératives* comme résultats d'une requête. Celles-ci sont des réponses indirectes, plus utiles que celles retournées par la requête. Ainsi, plusieurs travaux se sont intéressés à la proposition de techniques de *relaxation* pour les requêtes sémantiques [17–20]. Elles visent à élargir une requête pour retourner des réponses alternatives au lieu d'un résultat vide. Ces travaux présentent plusieurs limitations et notamment le fait qu'ils ne cherchent pas à identifier *les causes d'échec* de la requête utilisateur. Des approches ont été proposées pour cela dans le contexte des BDR et des systèmes de recommandation. Cependant, vu la différence de modèles de données et de langages de requêtes, l'adaptation de ces techniques au BDS n'est pas triviale. Aussi, les objectifs que nous nous sommes fixés dans cet axe de recherche sont les suivants :

- proposer des approches identifiant les causes d'échec d'une requête sémantique retournant un résultat vide ;
- prendre en compte les spécificités des ontologies dans ces approches ;
- proposer des approches de relaxation de requêtes sémantiques plus efficaces que celles de la littérature en s'appuyant sur la notion de cause d'échec.

1.5 Structure du mémoire

La structure de ce mémoire suit les trois axes de recherche introduits précédemment.

Le chapitre 2 aborde l'extension et la généralisation de la BDS OntoDB et de son langage OntoQL. Nous présentons d'abord cette BDS en nous focalisant sur son originalité. Puis, nous abordons des extensions que nous avons réalisées pour deux problématiques particulières : l'annotation de modèles métiers utilisés en ingénierie et la prise en compte des préférences utilisateur. Enfin, nous expliquons les limites de cette plateforme qui nous ont conduits à la généraliser en un système permettant la persistance et la manipulation de différents types de modèles.

Le chapitre 3 présente nos activités liées au processus de conception des BDS. Nous y présentons d'abord une approche d'intégration de besoins hétérogènes qui s'appuie sur des ontologies et permet d'utiliser les besoins intégrés pour sélectionner des structures d'optimisation sur la BDS en cours de développement. Ensuite, nous présentons notre démarche de conception d'une BDS comme une *Ligne de Produits Logiciels*. L'implémentation de cette démarche nécessite de prendre en compte la diversité des BDS dans les différentes phases de conception. Aussi, nous considérons la phase physique en proposant deux approches de sélection de vues matérialisées qui prennent en compte cette caractéristique.

Le chapitre 4 est consacré à l'exploitation des BDS. Nous y abordons le problème des réponses vides en proposant deux approches permettant de déterminer les raisons pour lesquelles une requête sémantique a retourné ce résultat insatisfaisant. Nous étendons ensuite ces approches au contexte des *ontologies incertaines*, c'est-à-dire à celles dont les faits peuvent ne pas être vrais. Enfin, pour faciliter la tâche de l'utilisateur, nous proposons des approches qui permettent de relaxer automatiquement une requête sémantique qui retourne une réponse vide.

Enfin, nous concluons dans le chapitre 5 en synthétisant nos contributions. Nous y développons des perspectives à courts termes, concrétisées par des travaux en cours, ainsi que des perspectives à moyen ou long terme qui visent à développer de nouveaux axes de recherche autour des ontologies.

Chapitre 2

Extension et généralisation des Bases de Données Sémantiques

2.1 Introduction

Avec la disponibilité de modèles d'ontologies standards tels que PLIB [1], RDFS [11] ou OWL [12], de nombreuses ontologies volumineuses ont été conçues. Dans le domaine du Web sémantique, nous pouvons citer des exemples d'ontologies résultant de projets académiques comme *YAGO* (1,2 milliard de triplets RDF) [21] et *DBPedia* (3 milliards de triplets RDF) [22] ou de projets commerciaux comme celles conçues par Google (*Knowledge Vault*) [23] ou Wallmart [24]. Dans des domaines de l'ingénierie, de nombreuses ontologies ont également été conçues et standardisées comme *Mechanical Fasteners* (ISO 13584-511, son schéma est composé de 250 classes et 410 propriétés) et *Cutting Tools* (ISO 13399, 500 classes et 360 propriétés). L'exploitation de ces ontologies volumineuses nécessite l'utilisation de systèmes persistants associés à des langages de définition, manipulation et interrogation d'ontologies. Cette problématique est apparue au début des années 2000 [25, 26] et continue d'être un domaine de recherche actif [27–31].

De nombreuses approches ont été proposées pour assurer la persistance des ontologies dans des *Bases de Données (BD)*. Nous les qualifions de *Bases de Données Sémantiques (BDS)*. Dans le contexte du Web sémantique, le terme *triplestore* est utilisé car ces BDS se focalisent sur la persistance de triplets RDF. Vu les hypothèses de typage faible associées à ce langage, les triplestores s'appuient généralement sur des schémas de représentation des données *éclatés*, dans lesquels chaque instance est décomposée en une multitude d'enregistrements. Lorsque les instances d'une même classe présentent une *structure régulière*, c'est-à-dire qu'elles ont tendance à avoir une valeur pour les mêmes propriétés, ce type de représentation pose des problèmes de performance [2, 32]. Par ailleurs, les triplestores ne proposent pas de mécanisme d'extension permettant de gérer les évolutions du modèle d'ontologies utilisé ou de lier ce dernier à d'autres modèles (par exemple, un modèle de préférences utilisateur).

Ce chapitre présente nos propositions pour répondre à ces problématiques. Dans la section 2.2, nous donnons une vue d'ensemble de l'état de l'art sur la persistance d'ontologies. Nous y définissons également nos objectifs de recherche. Dans la section 2.3, nous présentons brièvement la plateforme OntoDB/OntoQL que nous avons conçue pour répondre à nos objectifs. Nous décrivons ensuite deux exten-

sions de cette plateforme que nous avons réalisées.

- La première extension vise à faciliter l'intégration de données définies par des *modèles métiers* issus de l'ingénierie. Elle permet la persistance de ces modèles et leur annotation par des ontologies. Cette extension est décrite dans la section 2.4.
- La seconde extension vise à permettre l'interrogation d'ontologies en prenant en compte les préférences utilisateur. Elle permet la persistance de différents types de préférences et de les définir à l'aide d'ontologies. Cette extension est décrite dans la section 2.5.

Ces extensions de la plateforme OntoDB/OntoQL nous ont fait prendre conscience que celle-ci pourrait être utilisée plus généralement pour la persistance et l'exploitation de tout type de métamodèles, modèles et instances. Cependant, nous avons identifié plusieurs limitations de cette plateforme pour supporter des opérations usuelles en métamodélisation (par exemple, faire des transformations de modèles). Aussi nous présentons dans la section 2.6, une généralisation d'OntoDB/OntoQL pour qu'elle soit une *plateforme de métamodélisation persistante*. Nous établissons ensuite un bilan de cet axe de recherche dans la section 2.7 et concluons dans la section 2.8.

2.2 Objectifs de recherche

Un état de l'art exhaustif sur la persistance d'ontologies dépasse le cadre de ce mémoire. Nous nous contentons, dans cette section, de positionner nos travaux par rapport aux principales approches de cet état de l'art. La figure 2.1 en donne une vue d'ensemble. Nous distinguons d'abord les approches permettant un *stockage centralisé* d'ontologies, de celles dédiées à leur *stockage distribué*. Ces dernières visent la manipulation de données sémantiques massives [27]. Nous nous intéressons, dans nos travaux, aux approches centralisées.

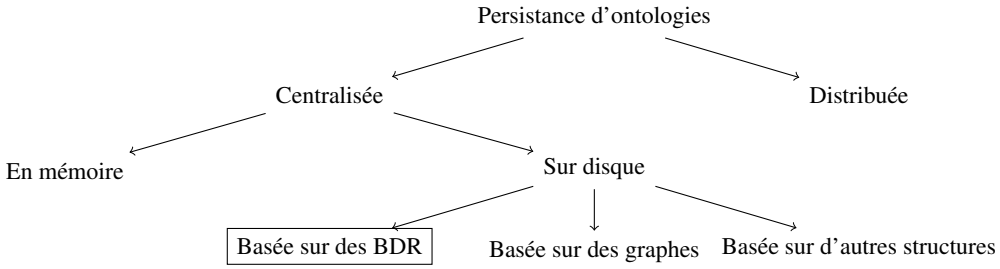


FIGURE 2.1 – Vue d'ensemble des approches permettant la persistance d'ontologies (nos travaux suivent celle encadrée)

Dans ce contexte, les premières approches qui sont apparues permettaient la persistance d'ontologies en mémoire centrale (par exemple, *OWLJessKB* [33] ou *Corese* [34]). Pour répondre à des besoins de performance et de fiabilité, de nombreuses approches permettant la persistance d'ontologies sur disque dur sont ensuite apparues. Elles peuvent être regroupées en trois principales catégories.

- Les approches basées sur les *Bases De Données Relationnelles (BDR)* comme, *3Store* [25], *SWStore* [35], *Jena2* [26], *Sesame* [36] ou *OntoMS* [37].
- Les approches basées sur la structure de graphe des données RDF qui proposent un stockage natif

de ces données sous cette forme (par exemple, *gStore* [38]).

– Les approches basées sur d'autres structures de données comme les index B^+ pour *RDF-3X* [39] et *Hexastore* [40] ou une matrice binaire pour *TripleBit* [41].

Notre approche étant basée sur les BDR, nous détaillons cette catégorie dans ce qui suit. Nous distinguons les BDS de cette catégorie en utilisant deux dimensions orthogonales : le *modèle de stockage* et l'*architecture* utilisés. Le modèle de stockage définit les tables dont se sert la BDS pour la persistance des données. Deux principaux modèles de stockage avaient été proposés au moment de la conception de la plateforme OntoDB/OntoQL (voir figure 2.2).

– *Le modèle de stockage vertical* [25]. Il s'appuie sur la structure des données RDF et consiste ainsi, principalement, en une table de triplets à trois colonnes *(sujet, prédicat, objet)*. Il s'accompagne généralement d'autres tables pour gérer les URI des ressources RDF.

– *Le modèle de stockage binaire* [35,36]. Il consiste à décomposer la table de triplets en un ensemble de tables à deux colonnes *(sujet, objet)*, une pour chaque prédicat. Le but de ce modèle est d'éviter que les requêtes nécessitent de nombreuses auto-jointures sur la table de triplets, comme dans l'approche verticale.

TRIPLES

Subject	Predicate	Object
ID1	type	GraduateStudent
ID1	memberOf	ID2
ID1	Undergraduate DegreeFrom	ID3
ID2	type	Department
ID2	subOrganizationOf	ID3
ID3	type	University

(a) Modèle de stockage vertical

TYPE

Subject	Object
ID1	Graduate Student
ID2	Department
ID3	University

UNDERGRADUATE DEGREEFROM

Subject	Object
ID1	ID3

MEMBEROF

Subject	Object
ID1	ID2

SUBORGANIZATIONOF

Subject	Object
ID2	ID3

(b) Modèle de stockage binaire

FIGURE 2.2 – Modèles de stockage des BDS existants lors de la conception d'OntoDB/OntoQL

L'architecture d'une BDS indique le nombre de parties utilisées pour la persistance des données. Les architectures des BDS proposées au moment de la conception de la plateforme OntoDB/OntoQL sont illustrées dans la figure 2.3. Certaines BDS utilisent une seule partie pour stocker les instances de l'ontologie et éventuellement son schéma. Puisque ces BDS contiennent également la partie *métabase*, présente dans tous les BDR, nous qualifions leur architecture comme étant *deux parties*. D'autres BDS séparent la représentation des instances de l'ontologie de celle de son schéma (par exemple, *Sesame* [36] ou *OntoMS* [37]), conduisant ainsi à une architecture *trois parties*.

Nous avons identifié plusieurs limitations aux modèles de stockage et architectures des BDS présentés précédemment. D'une part, lorsque les données sémantiques présentent une *structure régulière*, les modèles de stockage vertical et binaire impliquent de nombreuses jointures pour obtenir les instances d'une classe avec leurs valeurs de propriétés. Ces jointures pourraient être évitées en groupant les propriétés utilisées ensemble en une seule table. D'autre part, les architectures de BDS existantes ne représentent pas explicitement le modèle d'ontologies utilisé. Ainsi, ces BDS ne proposent pas de

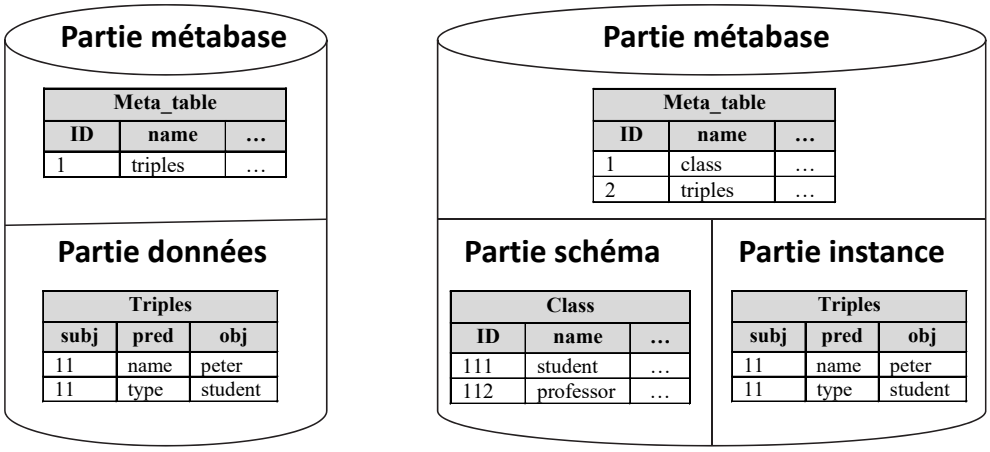


FIGURE 2.3 – Architectures des BDS existantes lors de la conception d'OntoDB/OntoQL

mécanismes pour gérer les évolutions du modèle d'ontologies utilisé ou le lier à d'autres modèles.

Afin de répondre à ces limitations, nous avons conçu la plateforme OntoDB/OntoQL. Nous avons ensuite proposé des extensions et une généralisation de cette plateforme. Les objectifs de ces travaux sont les suivants :

- définir une BDS permettant des évolutions ou changements du modèle d'ontologies utilisé et tirant profit des hypothèses de typage faites, par exemple, dans des domaines de l'ingénierie ;
- concevoir un langage d'exploitation de BDS, proche de SQL, tirant profit des caractéristiques de la BDS définie ;
- exploiter la BDS et le langage associé pour répondre à des besoins tels que la caractérisation sémantique de modèles métiers issus de l'ingénierie ou la modélisation de préférences utilisateur ;
- généraliser la BDS pour pouvoir supporter tout type de modèles et métamodèles ainsi que leur sémantique comportementale.

Nous avons répondu à ces objectifs par les contributions suivantes :

- contribution 1 : la définition de la plateforme OntoDB/OntoQL (voir section 2.3) ;
- contribution 2 et 3 : l'extension de la plateforme OntoDB/OntoQL pour l'annotation sémantique de modèles métiers issus de l'ingénierie (voir section 2.4) et la prise en compte de préférences utilisateur (voir section 2.5) ;
- contribution 4 : la généralisation de la plateforme OntoDB/OntoQL pour en faire un système de métamodélisation persistant (voir section 2.6).

2.3 Contribution 1 : la plateforme OntoDB/OntoQL

La plateforme OntoDB/OntoQL vise à apporter des solutions aux problématiques évoquées dans la section précédente. Nous les détaillons dans ce qui suit, en montrant pourquoi elles sont importantes pour nos domaines d'application.

2.3.1 Contexte et motivation : la persistance d'ontologies

La disponibilité de modèles d'ontologies standardisés comme PLIB [1], RDFS [11] ou OWL [12] a permis la création de nombreuses ontologies [21–24]. Pour leur persistance, des BDS ont été conçues [25, 26, 35–37] et continuent de l'être [27–31]. Généralement, ces propositions se focalisent sur la persistance d'ontologies définies avec un modèle d'ontologies spécifique. Elles ont pour objectif principal la performance qui peut être évaluée avec un banc d'essai comme LUBM [42] ou WatDiv [43]. À la différence de ces approches, nos travaux visent à concevoir une plateforme *extensible* pour qu'elle puisse s'adapter aux éléments suivants.

- *Évolutions du modèle d'ontologies utilisé*. Cette caractéristique permet de prendre en compte des modèles d'ontologies qui évoluent fréquemment. C'était le cas du modèle d'ontologies PLIB, utilisé dans nos travaux, qui subissait des évolutions en moyenne tous les six mois.
- *Changement de modèles d'ontologies*. Cette caractéristique permet de représenter et d'utiliser des ontologies basées sur divers modèles d'ontologies tant que celles-ci respectent les hypothèses sous-jacentes à la plateforme.
- *Persistance de différents types de modèles*. Cette caractéristique permet la persistance et manipulation dans un même environnement de modèles et des ontologies qui en décrivent le sens. Nous l'avons utilisée pour définir la sémantique de modèles métiers issus de l'ingénierie qui, par nature, se focalisent sur la structuration des données pour une application particulière (voir section 2.4).

La manière de représenter les instances des ontologies dans les BDS existantes est également inadaptée à nos domaines d'application. Comme nous l'avons vu, ces BDS utilisent le modèle de stockage vertical ou binaire. Ces modèles décomposent les instances des ontologies en de multiple enregistrements. Ils sont ainsi adaptés à des domaines où chaque instance est porteuse de sa propre structure (c'est-à-dire, qu'elle peut avoir des valeurs pour des propriétés appartenant à plusieurs classes qui seront différentes de celles des autres instances de sa ou ses classes d'appartenance).

Ce n'est pas le cas dans les ontologies PLIB, qui sont liées aux domaines de l'ingénierie, où des hypothèses de typage fort sont faites. Ainsi, les objets du domaine sont caractérisés par leur appartenance à une seule classe de base. Cette dernière définit généralement de nombreuses propriétés qui peuvent être utilisées pour décrire ses instances. Parmi celles-ci, il est possible d' identifier un sous-ensemble de propriétés pour lesquelles les instances de cette classe ont, pour la plupart, une valeur. Elles ont ainsi une structure régulière. Cette caractéristique n'est pas spécifique aux ontologies PLIB. L'étude menée par Duan *et al.* [44] a montré que de nombreux jeux de données RDF n'utilisent pas toute la flexibilité offerte par ce langage et présente ainsi une telle structuration.

Lorsque les instances des classes d'une ontologie ont une structure régulière, les modèles de stockage vertical et binaire posent un double problème. D'une part, les requêtes vont nécessiter de nombreuses jointures pour reconstruire la structure des instances, ce qui nuit à leurs performances. D'autre part, dans les BDS utilisant un de ces modèles, la notion de schéma de données (c'est-à-dire, l'identification explicite des propriétés qui caractérisent les instances), n'est pas présente. En conséquence, les utilisateurs peuvent éprouver des difficultés pour exprimer leurs requêtes sur les ontologies [15]. En effet, ils ne peuvent pas savoir quelles sont les propriétés utilisées pour décrire les instances des classes parmi toutes celles définies dans l'ontologie. En conséquence, ils risquent d'exprimer des requêtes retournant un résultat vide, car utilisant une propriété qui ne décrit pas les instances recherchées.

Pour répondre à ces problématiques, nous avons conçu une BDS extensible basée sur un modèle de stockage adapté aux données ayant une structure régulière. Elle est associée à un langage d'exploitation tirant profit de ces caractéristiques.

2.3.2 Principe général de la plateforme OntoDB/OntoQL

Dans cette section, nous présentons succinctement la plateforme OntoDB/OntoQL, conçue au cours des travaux de thèse de Hondjack Dehainsala [2] et de moi-même [7], qui est le point de départ de nos contributions post-thèse. Nous nous focalisons sur l'originalité de cette plateforme.

2.3.2.1 Architecture quatre parties

Pour permettre des évolutions du modèle d'ontologies utilisé, la BDS OntoDB propose une architecture *quatre parties* présentée dans la figure 2.4. Cette architecture étend celle trois parties par l'ajout du *métaschéma*. Cette nouvelle partie permet la persistance du modèle d'ontologies utilisé par la BDS sous la forme d'un ensemble d'*entités* et d'*attributs*. Elle contient ainsi des tables (`Entity`, `Attribute`) contenant les constructeurs du modèle d'ontologies supporté (par exemple, `Class` ou `Property`). Elle est illustrée sur la figure 2.4 (2).

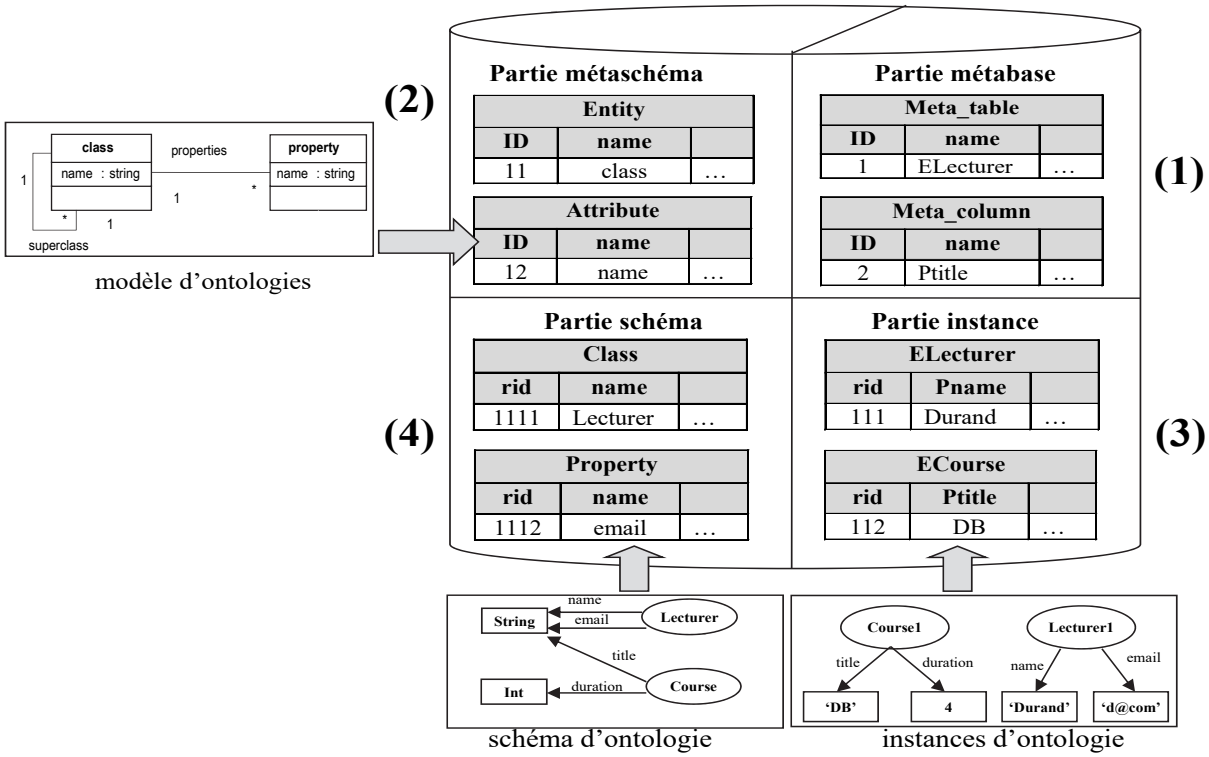


FIGURE 2.4 – Architecture *quatre parties* de la BDS OntoDB

À l'aide du langage OntoQL, le modèle d'ontologies contenu dans le métaschéma d'OntoDB peut être étendu.

Exemple. L'instruction suivante permet de décrire les classes des ontologies stockées dans OntoDB avec une remarque qui peut être définie dans différentes langues naturelles.

```
ALTER ENTITY #Class ADD ATTRIBUTE #remark MULTILINGUAL STRING
```

Explication. Le préfixe # est utilisé pour référencer un élément du modèle d'ontologies (par exemple, le constructeur de classes `#Class`). Il évite de coder en dur les éléments du modèle d'ontologies vu que celui-ci est extensible.

Une fois l'instruction précédente exécutée, toutes les classes stockées dans OntoDB seront décrites par une remarque. Des requêtes pourront être exprimées pour retrouver cette caractéristique.

Exemple. La requête suivante recherche les remarques définies en français sur les classes des ontologies.

```
SELECT #remark[FR] FROM #Class
```

Explication. Cette requête va rechercher toutes les classes définies avec le constructeur `#Class` mais aussi avec les constructeurs qui héritent de `#Class`. Cette caractéristique est utile pour les modèles d'ontologies qui permettent de définir plusieurs types de classe, comme PLIB ou OWL.

Le langage OntoQL repose sur un *modèle d'ontologies noyau* composé des principaux constructeurs des modèles d'ontologies usuels (voir [7] pour plus de détails sur ce modèle). Ce modèle d'ontologies noyau peut être étendu mais ne peut pas être modifié. Lorsqu'une extension du modèle noyau est faite par spécialisation, les nouvelles entités héritent du comportement de leurs super-entités. Ce comportement est défini dans la sémantique opérationnelle du modèle noyau. Ainsi, toute spécialisation de l'entité `Class` définit une nouvelle catégorie de classes. Ces dernières conservent, par héritage, le comportement usuel d'une classe (c'est-à-dire, un conteneur capable d'avoir des instances).

Nous présenterons dans les sections 2.4 et 2.5 des cas d'étude utilisant les capacités d'extension du modèle d'ontologies. Ils montrent l'intérêt de l'architecture quatre parties. Nous verrons ensuite, dans la section 2.6, une extension d'OntoDB/OntoQL permettant d'associer une sémantique comportementale à un élément ajouté au modèle noyau.

2.3.2.2 Modèle de persistance horizontal

Une seconde originalité de la plateforme OntoDB/OntoQL est le modèle de stockage utilisé pour persister les instances de l'ontologie (voir partie (3) de la figure 2.4). OntoDB utilise en effet le *modèle de stockage horizontal*. Ce dernier conserve une représentation proche du modèle relationnel en groupant, dans une même table, les propriétés qui sont définies ensemble. Le but de ce groupement est de faire en sorte que les requêtes nécessitent moins de jointures que dans l'approche binaire ou verticale.

La proposition de ce modèle de stockage repose sur le constat que le schéma d'une ontologie ne fait que *décrire* des propriétés qui peuvent être utilisées pour caractériser les instances d'une classe pour différentes applications. Il ne définit donc pas la *structure* de ces instances comme le fait un schéma de BD classique. En conséquence, nous avons choisi d'associer une *extension* à chaque classe. Elle précise l'ensemble des propriétés utilisées pour décrire les instances de cette classe dans l'application en cours de conception. Pour différentes applications, la même classe pourra ainsi avoir des extensions différentes

et l'intégration de données restera possible grâce au fait que ces extensions sont liées à la même classe. Cette approche repose cependant sur le fait qu'une instance n'ait qu'une seule classe de base (c'est-à-dire, l'hypothèse de typage fort).

Exemple. L'instruction OntoQL suivante crée la classe `Lecturer` avec différentes propriétés.

```
CREATE #Class Lecturer (
     DESCRIPTOR (#name[EN]='Lecturer', #remark[EN]='several ranks in UK')
     #Property  (name String, email String, phone String
                 worksFor REF(Department), teacherOf REF(Publication) ARRAY)
```

Explication. Cette instruction utilise le constructeur `#Class` pour définir une classe dont la description est donnée dans la clause `DESCRIPTOR` et les propriétés après le constructeur `#Property`. Comme le montre cet exemple, le langage OntoQL peut utiliser des types introduits avec les BD relationnelles-objets (type `REF` et `ARRAY`).

Une extension peut être associée à chaque classe créée avec le constructeur `#Class` ou un constructeur qui en hérite.

Exemple. L'instruction OntoQL suivante associe une extension à la classe `Lecturer`.

```
CREATE EXTENT OF Lecturer (name, email, worksFor)
```

Explication. Une extension est créée en spécifiant la liste des propriétés utilisées pour décrire les instances de la classe dans un contexte donné. Ici, l'extension de la classe `Lecturer` ne comprend que trois des cinq propriétés de cette classe.

Une requête peut être exprimée pour rechercher les instances d'une classe et ses valeurs de propriétés.

Exemple. La requête suivante retourne les instances de `Lecturer` avec leurs valeurs de propriétés.

```
SELECT name, email, phone, worksFor, teacherOf FROM Lecturer
```

Explication. Cette requête retournera la valeur `NULL` pour les propriétés `phone` et `teacherOf` qui ne sont pas utilisées dans l'extension de la classe `Lecturer`.

En supposant que les noms des classes et des propriétés soient également définis en français, cette requête pourrait aussi être écrite dans cette langue.

```
SELECT nom, email, téléphone, travaillePour, enseignantDe FROM Enseignant
 USING LANGUAGE FR
```

Enfin, pour un utilisateur qui connaîtrait les noms des tables et colonnes stockant ces informations dans OntoDB, il serait possible d'écrire une requête SQL. Celle-ci est considérée comme une instruction valide du langage OntoQL.

Nous nous sommes focalisés dans cette section sur les spécificités de la plateforme OntoDB/OntoQL que nous avons exploitées dans nos travaux post-thèse. Le lecteur intéressé par une présentation complète de cette plateforme pourra consulter [2, 7, 45].

2.3.3 Résultats

Au niveau conception, à notre connaissance, OntoDB est la seule BDS offrant des possibilités d'extension grâce à la présence du métaschéma. Ces capacités permettent de modifier le modèle d'ontologies utilisé par la BDS, ce qui était nécessaire pour la gestion des évolutions du modèle PLIB. Elles permettent aussi le stockage d'ontologies définies dans d'autres modèles d'ontologies. Ces capacités ne sont cependant pas sans limite. D'une part, les données et requêtes sont traitées selon les hypothèses faites dans le modèle PLIB (c'est-à-dire, celles du monde clos et d'unicité des noms). D'autre part, seule la sémantique structurelle des extensions faites peut être définie. Nous verrons dans la section 2.6 comment nous avons répondu à cette limitation.

Au niveau implémentation, la BDS OntoDB a été codée comme surcouche à PostgreSQL. Le langage OntoQL est lui codé en Java et chaque instruction est traduite en SQL pour pouvoir être exécutée sur OntoDB. Cette implémentation est disponible sur `https://forge.lias-lab.fr/projects/ontodb`. Nous l'avons utilisée pour mener des expérimentations détaillées dans [2]. Elles consistent à comparer le modèle de stockage horizontal avec les modèles vertical et binaire. Les résultats de ces expérimentations montrent que ce modèle de stockage fournit des meilleures performances pour les requêtes qui spécifient la ou les classes des instances recherchées. De plus, cette amélioration de performance, permise par le modèle horizontal, augmente avec le nombre de propriétés présentes dans la requête. Dans ces expérimentations, le seul cas où le modèle horizontal obtient des performances moindres que les deux autres modèles concerne les requêtes qui ne précisent pas les classes des instances recherchées et utilisent peu de propriétés.

Nous présentons, dans ce qui suit, des approches qui utilisent les capacités d'extension de la plateforme OntoDB/OntoQL pour répondre à différentes problématiques. Dans chacune de ces approches, le principe suivi est toujours le même : chaque extension d'OntoDB est accompagnée de celle du langage OntoQL.

2.4 Contribution 2 : annotation de modèles métiers en ingénierie

Dans cette section, nous abordons le problème de l'intégration de données dans le contexte de l'ingénierie. Nous commençons par décrire cette problématique.

2.4.1 Contexte et motivation : l'intégration de données en ingénierie

Les domaines de l'ingénierie s'intéressent principalement au cycle de vie d'un objet tel qu'un avion, une voiture ou un réservoir pétrolier. Ils se basent sur de nombreux *modèles métiers* pour gérer les différents aspects du cycle de vie de l'objet d'intérêt. Ces modèles sont générés par des logiciels et utilisés pour faire des simulations sur le domaine d'intérêt. Ils peuvent aussi servir à faire des vérifications ou validations sur l'objet étudié. Ils sont de différents types : schéma UML, tables de BD, structures de données définies dans un langage de programmation, modèle numérique, visualisation graphique, etc. Étant issus de différents champs d'expertise, ces modèles présentent une forte hétérogénéité en termes de formats, vocabulaire utilisé et données manipulées. Cela pose des problèmes aux ingénieurs pour échanger des données sur l'objet étudié ou utiliser les connaissances de divers champs d'expertise sur ce

même objet [46].

La solution habituelle pour répondre à ces problèmes est d'*intégrer* les données manipulées. Cela consiste à fournir une *interface unifiée* à ces sources de données afin de donner l'illusion d'un système d'information homogène et centralisé [47]. Cette approche est adaptée aux domaines où un modèle conceptuel unique peut être défini à partir de sources de données qui sont similaires les unes aux autres et qui évoluent peu. C'est par exemple le cas si l'on souhaite intégrer les données de différentes bibliothèques qui décrivent des livres avec des attributs similaires (par exemple, un titre, des auteurs ou un éditeur) mais dont la représentation peut être différente dans chaque source.

Le problème est plus complexe dans les domaines de l'ingénierie. En effet, les experts ont besoin d'intégrer différentes sources de données selon le problème qu'ils ont à traiter. De plus, les sources de données contiennent des informations qui sont disjointes du point de vue structurelle, du fait qu'elles proviennent de différents champs d'expertise. Prenons par exemple le cas d'un ingénieur pétrolier souhaitant étudier la répartition de roches plutoniques dans une région donnée. Cet ingénieur pourrait se poser des questions sur la géométrie 3D de ces roches et vouloir comprendre leur relation avec les couches de terrain sous lesquelles elles sont installées. Pour répondre à de telles questions, cet ingénieur devra utiliser des informations variées telles que des cartes géologiques de la région ou des BD issues de la géophysique et de la géochimie. Dans de telles sources de données, établir des liens entre ces informations ne peut souvent être faits que par des ingénieurs ou scientifiques à partir de connaissances qui ne sont pas explicitées dans les sources de données [48]. L'intégration des modèles métiers utilisés en ingénierie nécessite donc de rendre explicite la connaissance qui permet d'établir des liens entre ces modèles. Pour pouvoir faciliter l'échange et le partage d'information sur ces modèles, nous avons donc proposé de les lier à des ontologies. Cette approche repose donc sur la possibilité d'*annoter* des modèles métiers avec des ontologies du domaine.

Ces travaux ont été menés dans le cadre du projet ANR eWokHub. Celui-ci portait sur le domaine de l'ingénierie pétrolière et avait pour objectif de pouvoir caractériser des réservoirs pétroliers. Le but de cette activité est de construire un modèle précis du réservoir permettant de calculer la quantité de pétrole ou de gaz qu'il contient. Pour faire cela, des ingénieurs de différents champs des géosciences et de l'industrie pétrolière sont amenés à interpréter des données acquises sur les champs pétroliers. Ces dernières sont conformes à différents modèles métiers et représentent plusieurs aspects du réservoir. Lors de l'étude des réservoirs, les experts sont amenés à interpréter et à commenter les données liées à ces modèles métiers. Vu la diversité des vocabulaires employés dans les différents champs des géosciences, ce processus conduit à la production de données hétérogènes. En conséquence, il est difficile pour les experts de partager leurs interprétations, qui sont pourtant faites sur le même réservoir.

Nous avons développé plusieurs cas d'étude dans le cadre de ce projet. Ils nous ont permis de constater que les principales approches existantes pour annoter des modèles par des ontologies [49–51] ne répondent pas conjointement aux besoins suivants :

- disposer d'un environnement pour la persistance des modèles métiers et des ontologies utilisées pour les annoter ;
- permettre d'annoter un élément d'un modèle métier éventuellement par différents utilisateurs en conservant des informations sur ces annotations (par exemple, la date ou l'auteur de l'annotation) ;
- pouvoir retrouver, à l'aide de requêtes, les annotations faites sur un modèle métier ou inversement

les éléments de ce modèle qu'un concept d'une ontologie annote.

Pour répondre à ces objectifs, nous avons proposé une extension de la plateforme OntoDB/OntoQL qui est présentée dans la section suivante.

2.4.2 Principe général de l'extension de la plateforme OntoDB/OntoQL proposée

La figure 2.5 donne une vue simplifiée de notre proposition d'extension du métaschéma d'OntoDB. À la base, celui-ci ne contient qu'un modèle d'ontologies noyau, présenté partiellement dans la figure 2.5 (a). Il est constitué des constructeurs de classes (`Class`) et de propriétés (`Property`) auxquelles sont rattachés des types de données (`Datatype`). Afin de pouvoir stocker des modèles métiers issus de l'ingénierie, nous avons conçu un métamodèle permettant de définir ce type de modèles. Un extrait de ce métamodèle est présenté dans la figure 2.5 (c). Celui-ci est proche du métamodèle utilisé dans l'architecture MOF [14]. Il permet ainsi de créer des classes, des attributs et des relations. Il comporte aussi des informations techniques comme le format dans lequel les instances d'une classe d'un modèle métier sont représentées.

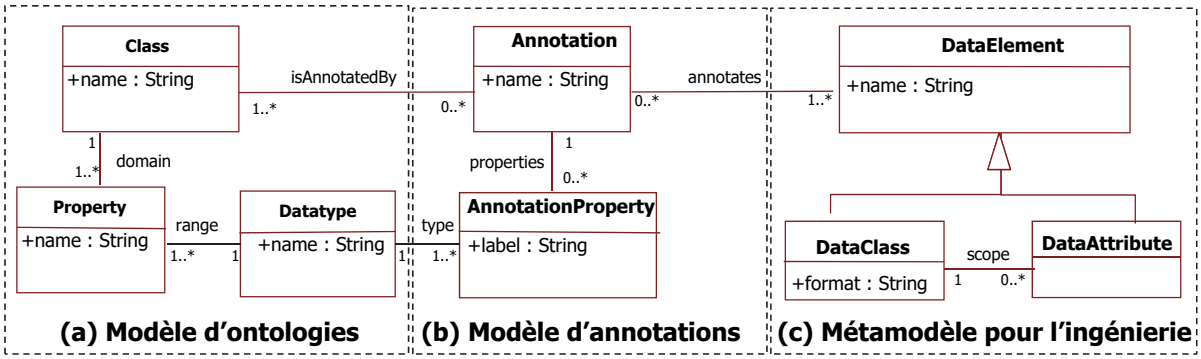


FIGURE 2.5 – Extrait de l'extension du métaschéma d'OntoDB proposée

Afin de faire le lien entre les modèles métiers et les ontologies du domaine des géosciences, nous avons défini un modèle d'annotations, présenté dans la figure 2.5 (b). Une annotation consiste à relier une ou plusieurs classes d'une ontologie à des éléments d'un modèle métier. Ces annotations sont décrites par des `AnnotationProperty`. Ces dernières sont définies par un libellé et un type de données. Ainsi, il est possible d'utiliser différents types de métadonnées, comme celles du Dublin Core [4] (par exemple, le créateur de l'annotation ou la date à laquelle elle a été faite). Cette extension peut être créée dans le métaschéma d'OntoDB à l'aide d'instructions OntoQL [46].

Exemple. L'instruction suivante permet de créer une version simplifiée de l'entité `Annotation`. Nous considérons, pour cet exemple et ceux de cette section, qu'une annotation relie une seule classe d'une ontologie à un seul élément d'un modèle métier et que cette annotation n'a qu'une propriété.

```
CREATE ENTITY #Annotation (
      #name STRING,
      #annotates REF (#DataElement),
```

4. http://dublincore.org

```
        #isAnnotatedBy REF (#Class),
        #property REF (#AnnotationProperty))
```

Explication. L'entité `Annotation` est créée avec un nom et des références vers un élément d'un modèle métier annoté, vers la classe qui sert à faire l'annotation et vers des propriétés qui la caractérisent.

Une fois l'extension complète réalisée, des annotations de modèles métiers peuvent être ajoutées en insérant des données dans cette extension.

Exemple. L'instruction suivante permet d'annoter l'élément `GridTile` d'un modèle métier avec la classe `Atmosphere` d'une ontologie.

```
INSERT INTO #Annotation (#name, #annotates, #isAnnotatedBy)
     VALUES ('ClimateAnnotation',
            (SELECT #oid from #DataClass WHERE #name = 'GridTile'),
            (SELECT #oid from #Class WHERE #name = 'Atmosphere'))
```

Explication. Cette instruction crée une annotation nommée `ClimateAnnotation`. L'élément annoté `GridTile` ainsi que la classe `Atmosphere`, qui l'annote, sont recherchés via des sous-requêtes.

Des requêtes peuvent ensuite être exprimées pour retrouver les annotations faites sur un élément d'un modèle métier ou inversement.

Exemple. La requête suivante retourne les annotations faites sur l'élément `GridTile`.

```
SELECT #Annotation.#isAnnotatedBy.#name FROM #Annotation
 WHERE #Annotation.#annotates.#name = 'GridTile'
```

Explication. Cette requête utilise des *expressions de chemin* (par exemple, `#isAnnotatedBy.#name`) pour retrouver le nom de la classe qui annote l'élément `GridTile`.

2.4.3 Résultats

Nous avons réalisé l'extension de la BDS OntoDB pour l'annotation de modèles métiers avec des instructions OntoQL. Cette extension comprend (i) la définition d'un métamodèle pour représenter des modèles métiers dans OntoDB et (ii) un modèle d'annotations qui permet de lier ces modèles métiers à des ontologies stockées grâce aux capacités natives de la plateforme. Par ailleurs, nous avons constaté que l'annotation de modèles métiers issus de l'ingénierie nécessite l'utilisation de plusieurs ontologies. Aussi, nous avons proposé une extension du langage OntoQL avec des opérateurs permettant d'établir des correspondances entre ontologies [46]. L'interpréteur de requêtes OntoQL a également été étendu pour pouvoir prendre en compte ces correspondances. L'implémentation complète du langage OntoQL est disponible sur https://forge.lias-lab.fr/projects/ontoql.

Pour valider notre approche, nous avons développé plusieurs cas d'étude dans le contexte des géosciences et du projet ANR EWokHub avec la collaboration de l'Institut Français du Pétrole (IFP) et du Bureau de Recherches Géologiques et Minières (BRGM). Ces cas d'étude nous ont d'abord amenés à définir, avec l'aide d'experts, différentes ontologies sur le domaine des géosciences. Nous avons établi des

liens entre ces ontologies grâce aux opérateurs de correspondance ajoutés au langage OntoQL [46]. Puis, nous avons représenté dans la plateforme OntoDB/OntoQL des modèles métiers utilisés pour la caractérisation de réservoirs pétroliers. Ces modèles ont ensuite été annotés avec les ontologies précédemment construites. Enfin, nous avons montré qu'un ensemble de requêtes OntoQL pouvaient être écrites pour répondre à des besoins exprimés par des géologues afin d'exploiter les modèles métiers annotés [46]. Lors de l'écriture de ces requêtes, nous avons constaté l'importance de pouvoir les personnaliser en fonction des préférences de chaque utilisateur. Nous abordons cette problématique dans la section suivante.

2.5 Contribution 3 : modélisation de préférences utilisateur

Nous présentons dans ce qui suit l'approche que nous avons suivie pour pouvoir attacher des préférences à des ontologies et les utiliser dans des requêtes. Cette approche repose à nouveau sur les capacités d'extension de la plateforme OntoDB/OntoQL.

2.5.1 Contexte et motivation : prise en compte de préférences utilisateur

Pour éviter que les utilisateurs soient submergés par de trop nombreux résultats fournis en réponse à leurs requêtes, ceux-ci doivent être triés et filtrés pour être utilisables. Pour répondre à cette problématique, de nombreuses approches ont proposé de prendre en compte les préférences des utilisateurs [52–56]. Elles permettent non seulement de présenter les résultats d'une requête de manière plus pertinente, mais aussi de faire des recommandations pour des groupes d'utilisateurs [57]. Concernant leur prise en compte dans les BD, des travaux ont été menés pour les intégrer aux BDR [52] ou aux entrepôts de données [53]. Cependant, dans ces approches, le traitement des préférences utilisateur est fortement couplé aux applications soit au niveau du code ou de l'interface graphique de l'application, soit au niveau de la BD en fonction de son modèle logique. Ce fort couplage rend le partage et la réutilisation de préférences entre applications difficiles.

Pour limiter le couplage entre les préférences utilisateur et les applications, plusieurs approches ont proposé de lier ces préférences à des ontologies qui sont conçues indépendamment d'une application particulière [54–56]. Cependant, nous avons identifié plusieurs limitations à ces approches [58]. En particulier, elles ne permettent pas de répondre conjointement aux trois besoins suivants :

- reposer sur un modèle de préférences défini formellement et incluant différents types de préférence proposés dans la littérature ;
- permettre la persistance des préférences définies selon le modèle précédent, autorisant ainsi une modification et consultation des préférences définies ;
- intégrer les préférences utilisateur à un langage d'interrogation dédié, personnalisant ainsi l'accès aux données de l'application.

Nous montrons dans la section suivante comment nous avons répondu à ces besoins en étendant la plateforme OntoDB/OntoQL.

2.5.2 Principe général de l'extension de la plateforme OntoDB/OntoQL proposée

Notre proposition [58] repose sur un modèle de préférences, dont un extrait simplifié est présenté dans la figure 2.6 (b). Celui-ci est relié au modèle d'ontologies initialement stocké dans OntoDB. Ce dernier est présenté, en partie, dans la figure 2.6 (a). Les préférences définies via le modèle proposé sont rattachées aux ontologies à l'aide de liens (`pref_link`). Ils associent une préférence à un ensemble de classes ou propriétés de l'ontologie. Prenons comme exemple une préférence `cheap`, indiquant ce qu'un utilisateur considère comme étant un hôtel abordable. Cette préférence pourrait être reliée à la propriété `price` d'une classe `Hotel` définie dans une ontologie portant sur le domaine du tourisme.

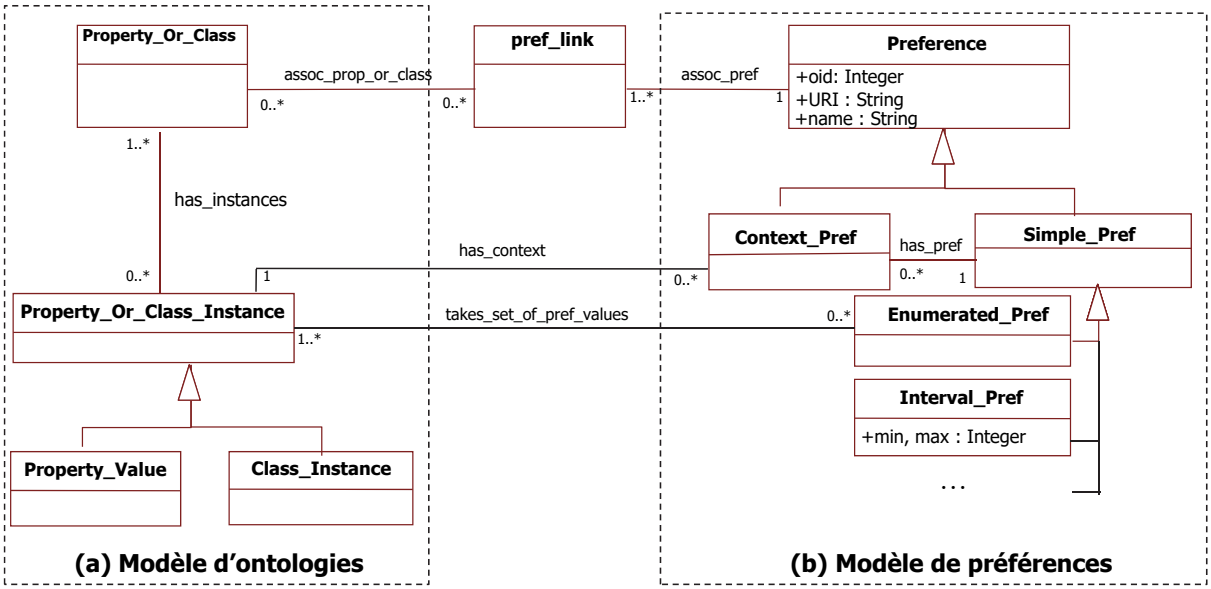


FIGURE 2.6 – Modèle de préférences proposé et ses liens avec le modèle d'ontologies

Dans le modèle de préférences proposé, nous avons distingué les préférences génériques (`Simple_Pref`) de celles qui dépendent du contexte dans lequel elles sont définies (`Contexte_Pref`). Par exemple, la préférence `cheap` évoquée précédemment peut être interprétée différemment selon que l'hôtel concerné soit situé à Paris, Londres ou Poitiers. Le contexte d'une préférence est ainsi défini grâce à un lien vers des instances de classes (`Class_Instance`) ou des valeurs de propriétés (`Property_Value`).

Le modèle de préférences que nous proposons inclut différents types de préférences définies dans la littérature et notamment celles issues des communautés BD et Web sémantique. Ainsi, une préférence énumérée (`Enumerated_Pref`) peut être définie en indiquant les instances de classes ou valeurs de propriétés préférées par l'utilisateur (par exemple, une préférence pour les hôtels `Ibis` ou `Novotel`). Une préférence de type intervalle (`Interval_Pref`) indique l'intervalle de valeurs préféré pour une propriété donnée (par exemple, la préférence `cheap` peut être définie pour les hôtels dont le prix est entre 80 et 90). Le modèle inclut d'autres types de préférences comme les préférences floues ou booléennes. La définition formelle du modèle de préférences complet que nous avons proposé est disponible dans [58].

Le modèle de préférence présenté précédemment est créé dans la plateforme OntoDB/OntoQL grâce aux capacités d'extension du métaschéma [59].

Exemple. L'instruction suivante crée l'entité `#Interval_Pref` pour pouvoir définir des préférences de type intervalle.

```
CREATE ENTITY #Interval_Pref UNDER #Simple_Pref(
      #min_value INT,
      #max_value INT)
```

Explication. L'entité `#Interval_Pref` est créée comme sous-entité de `#Simple_Pref` en utilisant le mot clé `UNDER`. Cela lui permet d'hériter des attributs d'une préférence (c'est-à-dire, un identifiant, une URI et un nom) et d'un lien vers des classes ou propriétés de l'ontologie (`#pref_link`). Toutes les préférences de ce type ont une valeur minimale et maximale définissant l'intervalle préféré.

Une fois l'extension du métaschéma faite, des préférences peuvent être définies et associées à des classes ou propriété de l'ontologie grâce au langage de manipulation de données d'OntoQL.

Exemple. Les deux instructions suivantes définissent la préférence `cheap` évoquée précédemment et l'associe à la propriété `price` de la classe `Hotel`. Pour simplifier, cet exemple considère qu'un `#pref_link` associe une seule préférence à une seule classe ou propriété d'une ontologie.

```
INSERT INTO #Interval_Pref (#name, #min_value, #max_value)
                    VALUES ('cheap', 80, 90)

INSERT INTO #Pref_Link (#assoc_pref, #assoc_prop_or_class)
     VALUES ((SELECT #oid from #Preference WHERE #name = 'cheap'),
             (SELECT #oid from #Property WHERE #name = 'price'
                                          AND #scope.#name='Hotel'))
```

Explication. La première instruction crée la préférence `cheap` en indiquant que l'intervalle de valeurs préféré est `[80, 90]`. La seconde fait le lien entre cette préférence et la propriété `price` de la classe `Hotel` en recherchant ces éléments via des sous-requêtes.

Nous avons ensuite étendu le langage OntoQL avec une clause `PREFERRING` afin de pouvoir exploiter les préférences définies.

Exemple. La requête suivante recherche des noms et prix d'hôtels sachant que l'on préfère ceux qui sont `cheap`.

```
SELECT name, price FROM Hotel PREFERRING 'cheap'
```

Explication. Selon le paramétrage effectué, la préférence `cheap` sera interprétée soit en réécrivant la requête avec la clause `WHERE price between 80 and 90`, soit en ordonnant les résultats pour que les hôtels dont le prix est entre 80 et 90 apparaissent en premier.

La clause `PREFERRING` permet également de combiner plusieurs préférences avec différents opérateurs. La définition complète de cette clause, à la fois sur les aspects syntaxiques et sémantiques, est disponible dans [58].

2.5.3 Résultats

Nous avons réalisé l'extension de la BDS OntoDB pour la prise en compte des préférences utilisateur avec des instructions OntoQL. Cette extension comprend le métamodèle de préférences que nous avons défini et qui regroupe différents types de préférences définis dans la littérature. Elle comprend également la possibilité d'établir des liens entre les préférences définies et les ontologies stockées dans OntoDB. Concernant le langage OntoQL, nous avons étendu sa syntaxe avec une nouvelle clause `PREFERRING` pour pouvoir inclure des préférences dans une requête et en combiner plusieurs. Du point de vue sémantique, nous avons modifié l'algèbre du langage OntoQL pour pouvoir interpréter cette clause de manière à ordonner ou filtrer les résultats en fonctions des préférences exprimées. Le script permettant l'extension de la BDS OntoDB pour la prise en compte de préférences utilisateurs est défini dans [58]. L'implémentation du langage OntoQL qui prend en compte la clause `PREFERRING` est disponible sur https://forge.lias-lab.fr/projects/ontoql.

Pour montrer l'intérêt de notre proposition, nous avons développé un cas d'étude complet dans [58]. Celui-ci consiste à définir différents types de préférence sur une ontologie réelle portant sur le domaine du tourisme et à montrer l'intérêt de ces préférences pour faciliter la recherche d'information dans cette ontologie. Ce cas d'étude a été valorisé dans le cadre d'une prestation avec la société Good Morning Planet [5]. Il montre, en effet, l'intérêt de prendre en compte les préférences utilisateur afin d'organiser des voyages personnalisés pour chaque client.

2.6 Contribution 4 : prise en compte de la sémantique comportementale

Dans les travaux présentés précédemment, nous avons utilisé les capacités d'extension de la plateforme d'OntoDB/OntoQL. Cependant, celles-ci ne permettent de définir que la sémantique structurelle de ces extensions. Nous avons dû introduire leur sémantique comportementale en modifiant le langage OntoQL. Cette section présente notre approche pour répondre à cette limitation.

2.6.1 Contexte et motivation : les systèmes de métamodélisation persistants

Comme nous l'avons vu dans la présentation de la plateforme OntoDB/OntoQL, celle-ci présente une architecture permettant plusieurs niveaux de modélisation. Elle est conforme à l'architecture MOF [14] puisqu'elle permet de stocker (i) des instances de l'ontologie (niveau M0 du MOF), (ii) le schéma des ontologies (niveau M1) et (iii) le modèle d'ontologies utilisé (niveau M2). Même si cette plateforme a été conçue pour la persistance d'ontologies, grâce à ses capacités d'extension permises par la présence de la partie métaschéma, elle peut être utilisée pour stocker tout type de modèles. Elle rentre ainsi dans la catégorie des *systèmes de métamodélisation persistants (Persistent MedaModeling Systems, PMMS)*, c'est-à-dire des systèmes qui permettent le stockage persistant de métamodèles, modèles et instance et qui sont équipés d'un langage d'exploitation pour la manipulation de ces différents niveaux de modélisation [60].

Dans la catégorie des PMMS, nous trouvons des systèmes comme ConceptBase [61], Rondo [62],

5. https://www.goodmorningplanet.com

DB-Main [63] ou Clio [64]. Cette catégorie s'oppose aux systèmes de métamodélisation classiques (par exemple, CDO [65] ou EMFStore [66]) dont le principe consiste à charger l'ensemble des données manipulées en mémoire centrale pour pouvoir ensuite les traiter. L'intérêt des PMMS par rapport à ces systèmes est de bénéficier des avantages des BD (par exemple, le passage à l'échelle, la sécurité, la gestion de la concurrence d'accès ou l'existence d'un langage d'exploitation déclaratif) sur lesquelles s'appuie le stockage persistant des données manipulées. Par contre, au moment de nos travaux, ces PMMS ne présentaient pas l'ensemble des fonctionnalités offertes par les systèmes de métamodélisation classiques.

Dans [60], nous avons défini un ensemble d'exigences pour qu'un PMMS aient les fonctionnalités attendues d'un système de métamodélisation. En étudiant les PMMS existants, dont OntoDB/OntoQL, nous avons identifié que les exigences suivantes n'étaient pas satisfaites, ou seulement partiellement.

- Le langage d'exploitation du PMMS devrait permettre l'introduction d'opérations pouvant s'appliquer sur les modèles et les instances stockés. Dans le cas d'OntoDB/OntoQL, des procédures stockées peuvent être définies et utilisées en s'appuyant sur PostgreSQL. Cependant, ces procédures stockées ne peuvent manipuler que les tables sous-jacentes à la plateforme OntoDB/OntoQL, elles ne permettent pas d'utiliser directement des concepts de plus hauts niveaux tels que les classes et propriétés des modèles stockés.
- Le langage d'exploitation du PMMS devrait permettre d'utiliser des opérations implémentées avec différents langages de programmation, qui soient internes ou externes au PMMS, ou en s'appuyant sur la disponibilité d'un service Web. Cette possibilité, non offerte dans OntoDB/OntoQL, permettrait de faciliter la réutilisation de code existant et d'offrir plus de flexibilité pour l'implémentation des opérations définies.
- Un PMMS devrait permettre d'utiliser une implémentation externe ou distante d'une opération *à la volée*, c'est-à-dire sans nécessité un redémarrage du PMMS. Cette caractéristique permettrait d'améliorer la disponibilité du PMMS.

Nous présentons dans la section suivante l'extension de la plateforme OntoDB/OntoQL que nous avons proposée pour répondre à ces exigences.

2.6.2 Principe général de la plateforme BeMoRe (Behavioral Model Repository)

La plateforme *BeMoRe (Behavioral Model Repository)* consiste en une extension de la plateforme OntoDB/OntoQL pour répondre aux exigences définies précédemment. Nous commençons par décrire l'extension de la BDS OntoDB.

2.6.2.1 Extension d'OntoDB : la plateforme BeMoRe

Pour pouvoir définir des opérations s'appliquant aux modèles et instances stockés dans OntoDB, nous avons proposé une extension qui introduit ces notions aux niveaux métamodèle et modèle d'OntoDB [60]. Par soucis de concision, nous ne décrivons que l'extension de la partie métamodèle (c'est-à-dire, le métaschéma). Nous présentons dans la partie encadrée de la figure 2.7 une vue simplifiée de l'extension du modèle conceptuel de cette partie. Elle permet la définition d'opérations ayant une liste de paramètres d'entrée et de sortie. Ceux-ci sont définis par des types de données qui peuvent être des types simples (par exemple, les types entier ou chaîne de caractères), des références vers des entités ou attributs

du métamodèle ou des collections. Chaque opération peut être associée à une ou plusieurs implémentations qui sont caractérisées par un ensemble de descripteurs (c'est-à-dire, des couples *clé-valeur*). Grâce à ces descripteurs génériques, différents types d'implémentation peuvent être décrits.

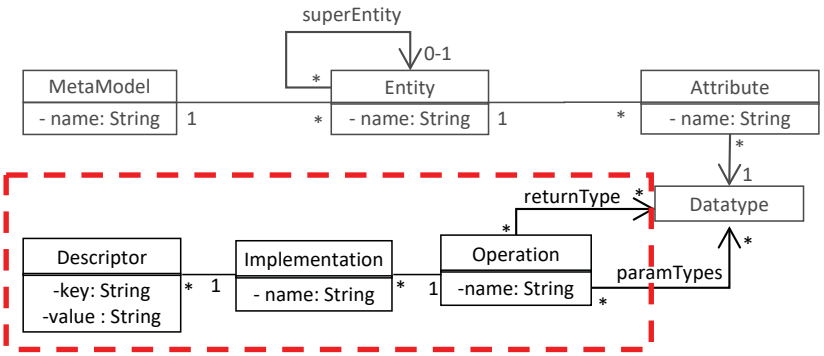


FIGURE 2.7 – Extension du modèle conceptuel de la partie métaschéma d'OntoDB

Nous avons défini formellement cette extension dans [67] et spécifié l'ensemble des contraintes qui s'appliquent à ce modèle. Le modèle logique qui correspond à cette extension est un ensemble de tables ajoutées à la partie métaschéma d'OntoDB. Un exemple simplifié du contenu de ces tables est présenté dans la figure 2.8. Dans cette figure, le nom des tables ajoutées est en rouge. Dans cet exemple, une opération `OWL2PLIB` est définie. Elle permet de convertir une classe OWL en une classe PLIB. Elle est associée à une unique implémentation faite en Java (`javaO2P`). Cette dernière est décrite avec différents descripteurs : la localisation de l'archive jar contenant le code Java à exécuter, le nom de la méthode qui doit être exécutée et le nom de la classe dans laquelle celle-ci est définie.

Operation

name	input	output
OWL2PLIB	#OWLClass	#PLIBClass

Attribute

...	...	...
...	...	...

Entity

...	...
...	...

Implementation

name	operation
javaO2P	OWL2PLIB

Descriptor

name	value	implem
Location	D:\	javaO2P
Class	JavaOp	javaO2P
Package	fr..lias	javaO2P
method	convert	javaO2P

FIGURE 2.8 – Extension du modèle logique de la partie métaschéma d'OntoDB

2.6.2.2 Extension du langage OntoQL

Afin d'exploiter l'extension d'OntoDB décrite précédemment, nous avons modifié le langage OntoQL pour définir et utiliser des opérations.

Exemple. L'instruction suivante permet de créer l'opération `OWL2PLIB` décrite dans la section précé-

dente. Cette instruction est basée sur le fait qu'il existe des entités `#OWLClass` et `#PLIBClass` pour créer des classes correspondant à ces modèles d'ontologies.

```
CREATE OPERATION #OWL2PLIB
           INPUT REF(#OWLClass) OUTPUT REF(#PLIBClass)
```

Explication. En suivant la logique de la conception du langage OntoQL, le préfixe # indique que l'opération `#OWL2PLIB` s'applique sur des modèles. Une opération concernant les instances ne possède pas ce préfixe.

Une fois qu'une opération est créée, une ou plusieurs implémentations peuvent lui être associées.

Exemple. L'instruction suivante crée une implémentation Java pour l'opération `#OWL2PLIB`.

```
CREATE IMPLEMENTATION #JavaO2P
  DESCRIPTORS (
    Type='Java', Location='/usr/lib/JavaOp.jar',
    Class='fr.ensma.lias.javaOp', Method='convert')
IMPLEMENTS #OWL2PLIB
```

Explication. Une implémentation est décrite par un ensemble de descripteurs. Pour le langage Java, ils précisent le fichier JAR contenant le code, la classe à exécuter, la méthode à appeler, etc.

Lorsqu'une opération a au moins une implémentation, elle peut être appelée dans une requête OntoQL.

Exemple. L'instruction suivante convertit l'ensemble des classes OWL stockées dans OntoDB en des classes PLIB en utilisant l'implémentation Java de l'opération `OWL2PLIB`.

```
SELECT #OWL2PLIB(c).#remark[en] FROM #Class AS c
 USING IMPLEMENTATION #JavaO2P->#OWL2PLIB
```

Explication. Cette requête retourne les remarques en anglais des classes PLIB obtenues. Elles correspondent aux commentaires (`comment`) des classes OWL converties.

2.6.3 Résultats

Nous avons proposé le PMMS BeMoRe qui est une généralisation d'OntoDB/OntoQL. Il permet de définir la sémantique comportementale des extensions permises par cette plateforme. Il repose sur la définition d'un métamodèle d'opérations qui peuvent s'appliquer aux niveaux modèle et instance. Il comprend également la modification de la syntaxe du langage OntoQL et de sa sémantique pour pouvoir définir de telles opérations, leur associer différentes implémentations et les utiliser dans des requêtes.

Le PMMS BeMoRe a été implémenté comme extension de la plateforme OntoDB/OntoQL. Son code est disponible à l'adresse https://forge.lias-lab.fr/projects/bemore. L'architecture de ce prototype est présentée dans la figure 2.9. Elle repose sur la plateforme OntoDB/OntoQL, elle-même implémentée sur PostgreSQL. Elle inclut une interface de programmation, appelée *Behavior API*, qui fait le pont entre la plateforme OntoDB/OntoQL et des environnements de programmation externes (par exemple, des services Web ou des programmes externes). En particulier, elle établit une correspondance

entre les types de données utilisés dans la plateforme OntoDB/OntoQL et ceux d'environnements externes. Comme indiqué sur la figure 2.9, notre prototype inclut actuellement deux implémentations de la Behavior API, une pour pouvoir appeler des services Web et l'autre pour exécuter des programmes Java.

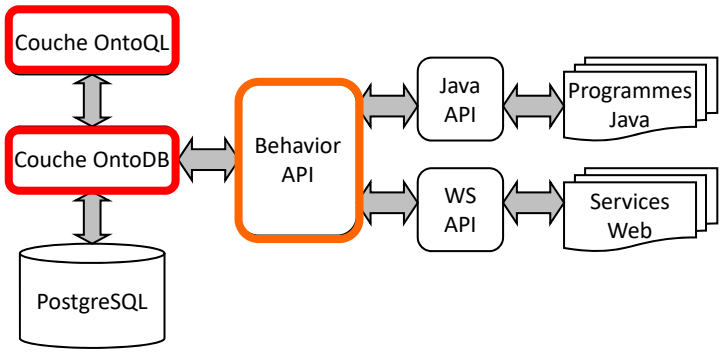


FIGURE 2.9 – Architecture du prototype BeMoRe

Nous avons proposé une évaluation expérimentale du prototype BeMoRe dans [67]. Nous avons comparé les temps d'exécution de requêtes impliquant une opération possédant trois implémentations : avec des procédures stockées codées sur PostgreSQL, avec un programme Java externe et en utilisant des services Web locaux. Nous avons réalisé ces évaluations expérimentales sur différentes tailles de données. Ces expérimentations nous ont permis de constater qu'une implémentation basée sur des procédures stockées est 4 à 5 fois plus rapide que celle basée sur un programme Java qui est elle-même bien plus rapide qu'une implémentation basée sur des services Web locaux. Pour optimiser les implémentations basées sur des programmes externes et services Web, nous avons identifié qu'il serait préférable de concevoir les opérations de manière à minimiser le nombre d'appels à l'implémentation externe. Il s'agit donc de favoriser les opérations qui prennent en paramètres des ensembles de données.

Enfin, nous avons montré l'intérêt du prototype BeMoRe en réalisant les cas d'études suivants.

- *La prise en compte de constructeurs du langage OWL dans la plateforme BeMoRe*. Nous avons ainsi montré dans [68] que des constructeurs comme les opérateurs ensemblistes (par exemple, `unionOf` ou `intersectionOf`) ou de restriction (par exemple, `hasValue` ou `allValuesFrom`) peuvent être introduits dans la plateforme BeMoRe. Cet ajout concerne non seulement la sémantique structurelle de ces opérateurs, grâce aux capacités d'extension de la plateforme OntoDB/OntoQL, mais aussi leur sémantique comportementale, grâce à la définition d'opérations permises par BeMoRe.
- *La conception d'une BDR en 3ème forme normale à partir d'une ontologie*. La conversion directe d'une ontologie en une BDR conduit généralement à des tables non normalisées. Cela est dû au fait qu'une ontologie peut contenir des concepts *définis* qui introduisent de la redondance (voir section 1.3.2). Aussi notre approche repose sur l'identification de dépendances entre classes et propriétés de l'ontologie [69]. Ces dépendances sont stockées dans la plateforme BeMoRe et des opérations permettent de les exploiter pour ne générer que des tables en 3ème forme normale.
- *L'analyse des systèmes temps réel*. La conception de ces systèmes reposent sur plusieurs modèles conçus avec différents formalismes tels que *SysML (Systems Modeling Language)* ou *AADL (Architecture Analysis and Design Language)*. Chacun de ces formalismes permet de définir un point

de vue particulier sur ces systèmes (par exemple, leur architecture ou les aspects matériels et logiciels). Nous avons montré dans [70] que ces différents formalismes peuvent être introduits dans la plateforme BeMoRe et que des opérations peuvent être définies pour réaliser des transformations entre ces modèles. Par ailleurs, la validation temporelle de ces systèmes peut être également réalisée avec une opération éventuellement implémentée dans différents environnements de programmation.

2.7 Bilan sur nos travaux liés à la persistance des données sémantiques

Nous dressons à présent un bilan sur les travaux présentés dans ce chapitre en commençant par synthétiser notre contribution à l'état de l'art.

2.7.1 Contributions à l'état de l'art

Les travaux menés dans le cadre de l'axe de recherche décrit dans ce chapitre nous ont permis d'apporter notre contribution à l'état de l'art sur différents aspects détaillés dans ce qui suit.

2.7.1.1 Contributions à la persistance d'ontologies

La figure 2.10 situe la plateforme OntoDB/OntoQL, en termes d'architecture et de modèle de stockage, parmi différentes BDS implémentées avec une BDR. Elle présente deux principales originalités. D'une part, elle repose sur l'architecture quatre parties qui permet de faire évoluer le modèle d'ontologies utilisé et de persister différents types de modèles. D'autre part, en s'appuyant sur l'hypothèse de typage fort, elle utilise le modèle de stockage horizontal. Celui-ci fournit de meilleures performances pour les requêtes qui spécifient la classe des instances recherchées et utilisent plusieurs propriétés.

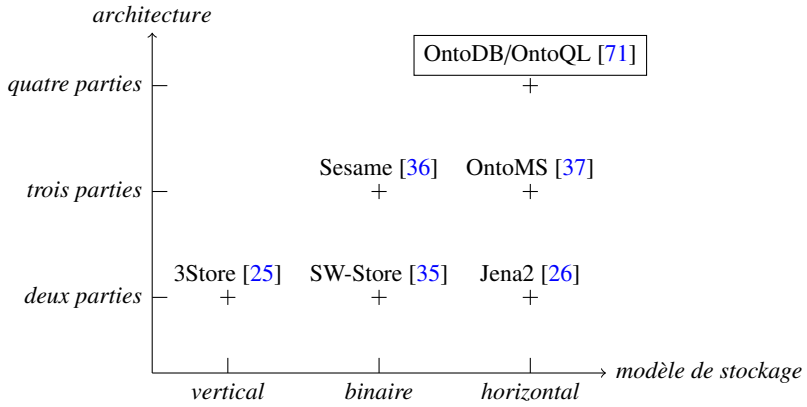


FIGURE 2.10 – Approches de persistance d'ontologies basées sur les BDR (notre approche est encadrée)

Notons que la plateforme OntoDB/OntoQL n'est pas la seule à avoir émis l'idée de garder une représentation proche du modèle relationnel en groupant, dans une table, les propriétés qui sont définies ensemble [26, 32]. Cependant, ce groupement n'est pas fait de la même façon que dans OntoDB. Dans

Jena2 [26], ce groupement doit être fait manuellement par l'administrateur de la BDS grâce à sa connaissance métier. Dans l'approche de *Levandoski et al.* [32], il est fait automatiquement en spécifiant des critères pour le définir (par exemple, le pourcentage maximum de valeurs nulles tolérées dans les tables ainsi formées). Dans ces deux approches [26, 32], le modèle de stockage horizontal est utilisé conjointement avec un autre modèle pour les données dont la structure est irrégulière.

Nous remarquons également que la distinction entre le schéma d'une ontologie (descriptif) et le schéma d'une BD (structurel), sur laquelle repose le modèle de stockage d'OntoDB, a été mise en avant par Pham *et al.* [15]. Contrairement à la plateforme OntoDB/OntoQL qui définit la structure d'une classe *a priori* via la notion d'extension, cette approche vise à identifier la structure des données sémantiques *à postériori*, en identifiant les propriétés définies ensemble dans un jeu de données RDF.

Enfin, à notre connaissance, OntoDB est la seule BDS proposant des capacités d'extension exploitées par le langage OntoQL. Nous avons montré l'intérêt de cette caractéristique pour deux problématiques : l'annotation de modèles métiers utilisés dans des domaines de l'ingénierie et la prise en compte des préférences utilisateur au sein d'une BDS.

2.7.1.2 Contributions aux langages d'exploitation d'ontologies

Nous avons conçu le langage OntoQL en parallèle d'autres langages orientés vers les technologies du Web sémantique et notamment le langage SPARQL [72] qui est maintenant le standard dans ce domaine. D'autres propositions avaient été faites dont le langage RQL [73] qui se rapproche le plus des fonctionnalités offertes par OntoQL. Le tableau 2.1 compare ces trois langages sur différents critères.

Langage	Modèle	Interrogation	Manipulation	Linguistique	SQL [6]
SPARQL [72]	RDF & entailment	schéma/instances	DML	fonctions	non
RQL [73]	RDFS	schéma/instances	DML	non	non
OntoQL [7]	noyau extensible	schéma/instances	DDL et DML	multilingue	oui

TABLE 2.1 – Comparaison d'OntoQL avec deux langages d'exploitation d'ontologies

Comme le montre ce tableau , ces trois langages permettent d'interroger le schéma d'ontologies, les instances ou conjointement le schéma et les instances. En plus d'un langage d'interrogation, ils peuvent proposer un *langage de définition de données (DDL)* ou de *manipulation de données (DML)*. Le langage OntoQL se démarque des deux autres langages sur les trois aspects suivants.

– *Le modèle d'ontologies supporté*. OntoQL repose sur un modèle d'ontologies noyau, composé des principaux constructeurs des modèles d'ontologies usuels. Il peut être étendu par des instructions du langage. A l'opposé, RQL repose sur une modèle d'ontologies figé qui se base sur RDFS. SPARQL considère l'ensemble des données (c'est-à-dire, les schémas et les instances des ontologies) comme des triplets RDF. La sémantique de différents modèles d'ontologies, tels que RDFS ou OWL, peut être pris en compte dans des requêtes SPARQL via différents modes de raisonnement (*entailment regime*).

6. Ce critère indique si le langage est compatible avec SQL.

– *L'exploitation des informations linguistiques*. Dans une ontologie, les noms des classes et propriétés ainsi que les commentaires faits sur ces éléments peuvent être définis dans plusieurs langues naturelles. Le langage OntoQL exploite ces informations pour permettre d'écrire des requêtes en plusieurs langues naturelles. SPARQL se base uniquement sur des fonctions pour permettre à l'utilisateur d'exploiter ces informations (par exemple, `lang` qui retourne la langue naturelle dans laquelle est défini un littéral). RQL ne les exploite pas.
– *La compatibilité SQL*. Même si les langages SPARQL et RQL reposent sur une syntaxe inspirée de SQL, ces langages ne sont pas compatibles avec ce langage. C'est par contre le cas d'OntoQL qui considère une instruction SQL (c'est-à-dire, l'interrogation des tables et de leurs colonnes) comme une instruction valide mais permet également d'interroger des ontologies à la fois sur son schéma et sur ses instances.

2.7.1.3 Contribution à l'Ingénierie Dirigée par les Modèles

Nous avons conçu la plateforme OntoDB/OntoQL pour qu'elle ne soit pas spécifique à un modèle d'ontologies donné. Dans la continuité de cette démarche, nous avons généralisé cette plateforme pour qu'elle permette de manipuler n'importe quel type de métamodèles, modèles et instances. Nous avons ainsi conçu le PMMS BeMoRe comme une surcouche à la plateforme OntoDB/OntoQL. Il offre un environnement persistant pour manipuler différents types de modèles via un langage déclaratif.

Comme le montre le tableau 2.2, l'originalité de BeMoRe par rapport à d'autres PMMS est double. D'une part, il permet d'introduire des opérations utilisateur *dynamiquement*, c'est-à-dire sans besoin de redémarrage. D'autre part, il offre la possibilité d'implanter ces opérations dans différents environnements de programmation interne à la BD sous-jacente ou externe (par exemple, à l'aide d'un programme Java ou de services Web). Grâce à ces capacités, il est possible de définir non seulement la sémantique structurelle des modèles stockés dans ce PMMS mais aussi leur sémantique comportementale .

PMMS	Persistance	Métamodèle extensible	Opérations	Implémentation	Disponibilité
BeMoRe [60]	BDR	Oui	Utilisateurs	Flexible	Immédiate
ConceptBase [61]	BD déductive	Oui	Utilisateurs	Prolog	Redémarrage
DB-Main [63]	BDR	Non	Utilisateurs	C++	Redémarrage
Rondo [62]	BDR	Non	Prédéfinis		

TABLE 2.2 – Comparaison de BeMoRe avec d'autres PMMS

2.7.2 Production scientifique

Le tableau 2.3 synthétise notre production scientifique liée aux travaux de recherche présentés dans ce chapitre et, plus généralement, à cet axe de recherche. Ce dernier a débuté avec nos travaux de doctorat et a été mené lors de la thèse de Youness Bazhar (co-encadrement avec Yamine Aït-Ameur) [60] et de celles de Laura Silveira Mastella [74], Dilek Tapucu [58], Nabil Belaïd [75] et Kevin Royer [76] pour

lesquelles je n'étais pas encadrant officiel mais dans lesquelles je me suis fortement impliqué [7].

Production	Description
Publications	4 RI, 2 RN, 19 CI, 7 CN, 3 CHAP, 6 API
Encadrements	Youness Bazhar (2013), Laura Silveira Mastella (2010), Dilek Tapucu (2010)
doctoraux	Nabil Belaïd (2011), Kevin Royer (2015)
Projets et contrats	EWokHub, DAFOE 4APP (ANR), CFCA (prestations), PatriMAR (projet PRES)
Logiciels	OntoDB/OntoQL, BeMoRe

TABLE 2.3 – Production scientifique liée à nos travaux sur la persistance des données sémantiques

Cet axe de recherche a donné lieu à 4 revues internationales (RI), 2 revues nationales (RN), 19 conférences internationales (CI), 7 conférences nationales (CN), 3 chapitres de livres d'audience internationale (CHAP) et 6 autres publications internationales (API) (poster, démonstration). Parmi ces publications, les principales sont les suivantes.

- Un article dans la revue internationale *Soft Computing* [71] qui synthétise nos travaux et, plus généralement, ceux de l'équipe IDD sur la conception, persistance et exploitation d'ontologies dans des domaines de l'ingénierie.
- Un article dans la revue internationale *International Journal of Semantic Computing (IJSC)* [77] qui présente le langage OntoQL et son utilisation dans différents contextes applicatifs.
- Deux articles dans les conférences internationales *International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2012)* [69] et *International Conference on Software Engineering & Knowledge Engineering (SEKE 2013)* [67] qui présentent les travaux réalisés dans le cadre de la thèse de Youness Bazhar sur la prise en compte de la sémantique comportementale dans la plateforme OntoDB/OntoQL.
- Un chapitre du livre *Shared Earth Modeling: Knowledge Driven Solutions for Building and Managing Subsurface 3D Geological models* [45] qui présente les travaux faits dans le contexte du projet ANR EWokHub et des thèses de Laura Silveira Mastella [74] et Nabil Belaid [75].

Ces travaux ont impliqué le développement des logiciels OntoDB/OntoQL [8] (50K lignes de code pour la partie OntoQL, environ 1500 téléchargements), BeMoRe [9] (66K lignes de code, environ 100 téléchargements). Ces logiciels ont été utilisés dans le cadre de plusieurs thèses de doctorat (Dilek Tapucu, Laura Silveira Mastella, Nabil Belaid, Youness Bazhar, Ilyès Boukhari, Chédlia Chakroun, Chimène Fankam, Bery Mbaiossoum, Kevin Royer, Henri Valéry Teguiak et Okba Barkat), projets académiques (ANR EWok-Hub, DAFOE 4APP, projet PRES PatriMAR) et applications industrielles (avec l'Institut Français du Pétrole *IFP*, le Bureau de Recherches Géologiques et Minières *BRGM*, l'Électricité de France *EDF*, la Compagnie Française de Câblage *CFCA*, la société Geosiris et le CRITT Informatique).

7. Références : Yamine Aït-Ameur, yamine@enseeiht.fr et Ladjel Bellatreche, bellatreche@ensma.fr
8. https://forge.lias-lab.fr/projects/ontodb
9. https://forge.lias-lab.fr/projects/bemore

2.8 Conclusion et perspectives de recherche

Dans ce chapitre, nous avons présenté notre premier axe de recherche sur les BDS. La base de ces travaux est la plateforme OntoDB/OntoQL. Elle présente deux principales originalités.

- Proposer un modèle de stockage adapté à des données sémantiques dont la structure est régulière. Au niveau du langage OntoQL, cela se traduit par la possibilité d'associer une *extension* à une classe qui explicite cette structure.
- Proposer une architecture *quatre parties* contenant la partie *métaschéma* qui permet de persister le modèle d'ontologies utilisé dans la BDS. Dans le langage OntoQL, cette partie contient un *modèle d'ontologies noyau* composé des principaux constructeurs des modèles d'ontologies usuels. Ce modèle peut être étendu avec des instructions du langage.

Par la suite, nos travaux se sont basés sur ces capacités d'extension pour répondre à différentes problématiques telles que l'annotation sémantique de modèles métiers issus de l'ingénierie ou la prise en compte de préférences utilisateur. Dans ces approches, nous avons suivi un principe général qui est de proposer une extension du métaschéma d'OntoDB et d'étendre le langage OntoQL pour prendre en compte la sémantique comportementale de cette extension.

Pour éviter d'avoir à systématiquement modifier le langage OntoQL dès qu'une extension du métaschéma est réalisée, nous avons proposé une généralisation de la plateforme OntoDB/OntoQL : le PMMS BeMoRe. Comparé aux autres PMMS de la littérature, BeMoRe permet l'introduction d'opérations aux niveaux modèles et instances. Elles peuvent être implantées avec différentes technologies, et ceci, sans nécessiter le redémarrage du PMMS.

Les travaux de cet axe de recherche ouvrent plusieurs perspectives sur les domaines des BDS et PMMS. Nous décrivons les deux pistes principales identifiées.

Modularité de la plateforme OntoDB/OntoQL. Concernant les BDS, la principale évolution possible de la plateforme OntoDB/OntoQL réside dans l'amélioration de sa *flexibilité*. En effet, cette plateforme nécessite actuellement que les ontologies et éventuels modèles qu'elles annotent soient stockés dans celle-ci. Or, ces différents éléments peuvent exister en dehors de la plateforme. Par exemple, le schéma d'une ontologie peut être disponible en ligne et accessible via un service Web. Les instances d'une ontologie peuvent également provenir d'une BD existante. Comme autre exemple, les modèles annotés par des ontologies peuvent être disponibles dans des systèmes d'informations externes.

Actuellement, la plateforme OntoDB/OntoQL nécessite d'*importer* ces éléments, ce qui est problématique s'ils évoluent fréquemment. À la place, notre idée serait de rendre la plateforme OntoDB/OntoQL plus *modulaire*. Elle permettrait ainsi la persistance et l'interrogation des différentes parties (c'est-à-dire, celles stockant le modèle d'ontologies, les schémas et les instances des ontologies) en interne dans la plateforme ou en externe lorsque ces données ou ontologies sont disponibles ailleurs. Cela pose plusieurs challenges, notamment pour établir des liens entre ces éléments externes et la plateforme ainsi que pour maintenir des performances acceptables.

Composition d'opérations hétérogènes au sein du PMMS BeMoRe. Concernant les PMMS, la principale piste d'évolution de BeMoRe consiste à permettre de définir une opération en composant un ensemble d'autres opérations. Cela autoriserait ainsi la composition d'opérations implémentées dans différentes technologies (par exemple, par un service Web ou en Java). Cette capacité serait utile dans

le contexte de l'ingénierie où, pour répondre à une tâche donnée, les ingénieurs ont besoin de réaliser une chaîne de traitements sur des données hétérogènes (problématique détaillée en section 2.4). Pour répondre à cette perspective, nous pensons qu'il serait intéressant de se baser sur les travaux menés sur l'orchestration de services Web. Dans notre contexte, puisque nous souhaitons composer des opérations dont les implémentations sont hétérogènes, un challenge sera de gérer le flot de données entre les différentes opérations.

Chapitre 3

Ingénierie des Bases de Données Sémantiques

3.1 Introduction

Dans le chapitre précédent, nous avons présenté nos travaux concernant l'extension et la généralisation des *Bases de Données Sémantiques (BDS)*. Dans ce chapitre, nous nous intéressons à un autre aspect de celles-ci, à savoir leur *processus de conception*. Celui-ci a lieu lors du développement d'applications utilisant des ontologies. Pour la persistance de celles-ci, une BDS est nécessaire. Nous considérons le processus qui mène à la définition de cette BDS et à son utilisation pour stocker des ontologies. Ce processus peut s'appuyer sur une BDS existante telle que OntoDB/OntoQL [2] ou Jena TDB [26]. La difficulté dans ce cas est de choisir celle qui répond aux exigences applicatives (par exemple, en termes de performance). Ce processus peut aussi consister à concevoir une nouvelle BDS. C'est par exemple le cas si l'on souhaite stocker les ontologies dans une BDR et y accéder via le langage SQL. Nous avons rencontré ce cas lors de projets industriels qui nécessitaient d'utiliser des technologies éprouvées et de ne pas multiplier les systèmes utilisés.

Comme dans les *Systèmes de Gestion de Données (SGD)* traditionnels tels que les Bases de Données Relationnelles (BDR) ou les entrepôts de données, le processus de conception des BDS suit les principales étapes suivantes [78].

- *La phase de collecte et d'analyse des besoins* consiste à représenter ces derniers selon un formalisme donné (par exemple, UML) et à en vérifier la consistance.
- *La phase conceptuelle* vise à définir l'ontologie sur laquelle se basera l'application en cours de conception. Une ontologie existante peut être utilisée. Si aucune ne convient, une ontologie peut aussi être conçue en choisissant un modèle d'ontologies.
- *La phase logique* consiste à définir l'ensemble des tables de la BDS [10] qui seront utilisées pour le stockage de l'ontologie définie dans l'étape précédente. Si l'on souhaite utiliser une BDS existante, cette phase consiste à en choisir une qui répond aux exigences applicatives, puisque le modèle logique utilisé découlera de ce choix.

10. Comme dans le chapitre 2, nous nous focalisons sur les BDS implémentées avec une BDR.

– *La phase physique* fournit une implémentation de la BDS sur la BDR choisie. Cette phase intègre également la sélection de structures d'optimisation telles que des index ou des vues matérialisées pour optimiser les temps de traitement.

Les BDS présentent des spécificités qui ont été peu prises en compte dans ces différentes phases. Ainsi, dans la phase de recueil des besoins utilisateurs, les applications basées sur des ontologies impliquent souvent différents *partenaires* qui peuvent définir des besoins *hétérogènes*. À notre connaissance, peu d'approches ont été proposées pour permettre l'intégration de ces besoins. Concernant les phases conceptuelles et logiques, comme nous l'avons vu dans le chapitre 2, les BDS présentent une *diversité* sur différents aspects tels que le modèle d'ontologies, l'architecture ou le modèle de stockage utilisés. Les méthodes [79–81] et outils [63, 82] existants pour concevoir des *Bases de Données (BD)* ne prennent pas en compte cette caractéristique. C'est également le cas des approches proposées pour la sélection de vues matérialisées lors de la phase physique [83–85]. Celles-ci sont généralement conçues pour un type particulier de BDS.

Ce chapitre présente nos propositions pour répondre à ces problématiques. Nous commençons par préciser nos objectifs de recherche par rapport aux travaux de l'état de l'art dans la section 3.2. Puis, nous présentons, dans la section 3.3, une extension du processus de conception des BDS avec une phase d'intégration de besoins hétérogènes. Pour expliciter la variété de choix possibles dans ce processus, nous proposons ensuite, dans la section 3.4, une démarche de conception des BDS qui s'appuie sur les méthodes et outils des *Lignes de Produits Logiciels*. Cette démarche nécessitant de prendre en compte la diversité des BDS dans les différentes phases de conception des BDS, nous considérons ensuite la phase physique. Ainsi, nous proposons, dans la section 3.5, deux approches de sélection des vues matérialisées qui peuvent s'appliquer à différentes BDS et prennent en compte leurs spécificités. Nous faisons ensuite un bilan de cet axe de recherche dans la section 3.6 et concluons dans la section 3.7.

3.2 Objectifs de recherche

Le processus de conception des SGD en quatre phases (recueil des besoins, conceptuelle, logique et physique) est aujourd'hui bien établi. Historiquement, l'apparition de nouveaux modèles de données (par exemple, le modèle relationnel) et de nouveaux types de SGD (par exemple, les entrepôts de données) a entrainé deux types d'évolution dans ce processus [86].

– *Une évolution verticale*, c'est-à-dire l'ajout de nouvelles phases à ce processus. Par exemple, dans le contexte des BD, le processus de conception était initialement uniquement constitué de la phase physique, basée sur le modèle hiérarchique ou en réseau. Avec la définition du modèle relationnel puis du schéma entité-association, les phases logiques et conceptuelles sont apparues pour garantir une indépendance avec l'implémentation physique. Dans le contexte des entrepôts de données, la phase *ETL (Extract Transform Load)* a été ajoutée à ce processus pour permettre l'intégration des données dans ce type de SGD.
– *Une évolution horizontale*, c'est-à-dire l'enrichissement de chaque phase avec de nouveaux modèles et outils. Par exemple, dans le contexte des BD, de nombreux langages de modélisation de données plus expressifs que le modèle entité-association ont été proposés pour la phase conceptuelle (par exemple, les modèles sémantiques [4]). De nouveaux modèles logiques sont également

apparus avec le développement des BD orientées objets, spatiales, XML, etc.

Ce constat nous a amené à étudier l'impact de l'apparition des BDS sur ce processus. La figure 3.1 présente notre point de vue sur les évolutions verticales et horizontales du processus de conception des BDS. Concernant l'évolution verticale, les applications basées sur des ontologies impliquent souvent différents *partenaires* $(p_1, \ldots, p_n)$ qui vont définir des besoins *hétérogènes*, c'est-à-dire basés sur des formalismes et vocabulaires différents. Des approches ont été proposées dans le contexte de l'*Ingénierie des Besoins* pour combiner plusieurs formalismes de besoins [87–91]. Cependant, à notre connaissance, elles ne prennent pas en compte l'hétérogénéité des besoins à la fois sur les formalismes et vocabulaires utilisés pour les définir. De plus, elles ne proposent pas d'utiliser les besoins dans d'autres phases de conception d'une BD.

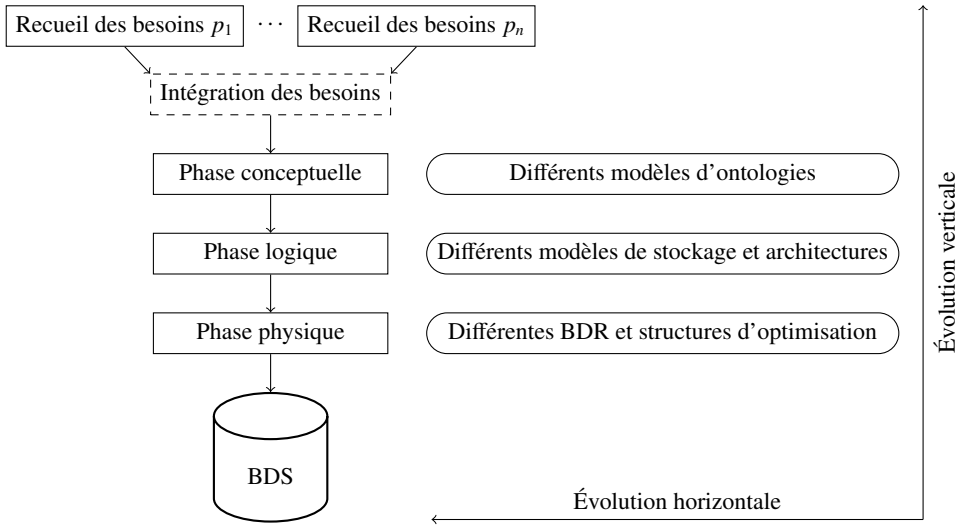


FIGURE 3.1 – Évolutions verticales et horizontales du processus de conception des BDS

Concernant l'évolution horizontale, les BDS présentent une *diversité* sur différents aspects tels que le modèle d'ontologies supporté ou l'architecture et le modèle de stockage utilisés. Cette diversité n'est pas prise en compte dans les approches actuelles de conception des SGD. Celles-ci sont en effet basées sur des outils (par exemple, les outils CASE [11] [63] ou advisors [82]) et méthodes (par exemple, des patrons de conception [79–81]) spécialisés pour un sous-ensemble des phases de conception (par exemple, les advisors se focalisent sur la phase physique). Elles n'explicitent pas les différents choix possibles lors du processus de conception des BDS et ne montrent pas l'impact d'un choix sur le reste du processus. La diversité des BDS n'est pas non plus prise en compte dans les approches de sélection de vues matérialisées [83–85], qui se focalisent généralement sur un type particulier de BDS.

Ces limitations nous ont conduits à développer un axe de recherche sur la conception des BDS. Les objectifs des travaux menés dans cet axe sont les suivants :

- définir une phase d'intégration de besoins hétérogènes liée aux autres phases de conception des BDS ;
- élaborer une démarche générale de conception des BDS prenant en compte leur diversité ;
- étudier l'impact de la diversité des BDS sur la phase physique.

11. Computer-Aided Software Engineering

Nous avons répondu à ces objectifs par les contributions suivantes :

- contribution 1 : la proposition d'une approche d'intégration de besoins hétérogènes permettant de définir des structures d'optimisation sur la BDS en cours de conception (voir section 3.3) ;
- contribution 2 : la définition d'une démarche de conception de BDS basée sur les méthodes et outils des *Lignes de Produits Logiciels* (voir section 3.4) ;
- contribution 3 : la proposition de deux approches de sélection de vues matérialisées qui prend en compte la diversité des BDS (voir section 3.5).

3.3 Contribution 1 : intégration des besoins hétérogènes

Nous proposons dans cette section une approche d'intégration de besoins hétérogènes. Nous commençons par expliquer les raisons pour lesquelles nous avons mené ces travaux.

3.3.1 Contexte et motivation : Ingénierie des Besoins

Les applications basées sur des ontologies impliquent souvent différents acteurs qui utilisent ces dernières comme modèles communs. C'est le cas dans des domaines de l'ingénierie, tels que l'aéronautique ou l'automobile, qui comprend de nombreuses *entreprises étendues*. Celles-ci sont composées de différents *partenaires*, généralement situés sur des sites géographiques distants, qui collaborent pour la réalisation d'un projet commun. Dans ce contexte, des ontologies standardisées sont souvent utilisées comme modèles de référence [1]. Comme exemple d'entreprise étendue, nous pouvons citer la société Airbus qui a fait intervenir quatre partenaires situés dans différents pays pour la réalisation de l'Airbus A380.

Les partenaires des entreprises étendues étant généralement autonomes et ayant des compétences et habitudes différentes, les systèmes informatiques qu'ils définissent présentent une *hétérogénéité* notamment en termes de modèles de données utilisés. Et, puisque ces partenaires doivent collaborer pour arriver à un résultat commun, il est nécessaire d'*intégrer* les données utilisées dans ces systèmes. De nombreux travaux se sont intéressés à cette problématique [10, 47, 92, 93]. Constatant que le problème d'intégration vient du fait que différents partenaires peuvent avoir leurs propres points de vue et vocabulaires pour la même notion, certains utilisent des ontologies et les BDS associées pour traiter ce problème [10, 93]. Par contre, peu d'approches se sont intéressées au processus qui conduit à cette hétérogénéité des données et plus particulièrement à la phase de recueil des besoins, c'est-à-dire à celle dans laquelle le cahier des charges est transformé en un ensemble de *besoins* plus ou moins formalisés. Ces derniers étant eux-mêmes définis par différents partenaires, ils présentent une hétérogénéité sur les vocabulaires et formalismes utilisés pour les définir. Aussi, nous avons étudié le problème de l'intégration de besoins hétérogènes afin d'introduire cette étape dans le processus de conception des BDS.

La définition des besoins a été largement étudiée dans la communauté de l'*Ingénierie des Besoins*. Un *besoin* est défini comme une nécessité ou un désir éprouvé par un utilisateur [94]. La définition des besoins suit les étapes suivantes [95] :

- l'*élicitation* visant à identifier les partenaires et leurs besoins ;
- la *modélisation* et l'*analyse* visant respectivement à représenter les besoins selon un langage par-

ticulier et à détecter les conflits et les anomalies entre les besoins identifiés ;

- la *validation* visant à vérifier de manière automatique ou manuelle que chaque besoin correspond bien à une attente sur le système ;
- la phase de *gestion* visant à gérer l'évolution et la traçabilité des besoins.

Dans nos travaux, nous nous sommes intéressés aux phases de modélisation et d'analyse des besoins. Constatant que les principaux travaux de l'*Ingénierie des Besoins* liés à notre problématique [87–91] ne prenaient pas en compte le fait que les besoins puissent être hétérogènes et que la phase de leur intégration pourrait être liée aux autres étapes de conception d'un SGD, nous nous sommes fixés les principaux objectifs suivants.

- Permettre l'intégration des besoins au niveau du vocabulaire utilisé. Cette hétérogénéité porte, d'une part, sur les différentes langues naturelles qui peuvent être utilisées pour définir des besoins. Et d'autre part, sur le fait que, pour une même langue, des ambiguïtés peuvent se poser sur les termes utilisés (par exemple, les problèmes de synonymes et d'homonymes).
- Unifier les formalismes utilisés par les différents partenaires d'une entreprise étendue pour définir les besoins. De nombreux formalismes existant, nous avons choisi de nous focaliser sur trois formalismes souvent utilisés (cas d'utilisation d'UML, MCT Merise [96] et Orienté buts [97]).
- Analyser les besoins ainsi intégrés. Cet objectif vise en particulier à identifier des liens entres les besoins intégrés afin de trouver ceux qui sont contradictoires ou redondants.
- Montrer l'intérêt de l'intégration des besoins sur les autres étapes du processus de conception d'un SGD tel qu'une BDS. Cet objectif étant large, nous nous sommes focalisés sur la phase physique, c'est-à-dire à l'optimisation du SGD à partir des besoins intégrés.

3.3.2 Principe général de notre approche d'intégration des besoins

Pour des raisons de concision, nous ne présentons que le principe général de notre approche d'intégration des besoins, ce qui couvre les deux premiers objectifs définis précédemment (nos autres propositions sont décrites dans [98]). L'origine de ces travaux est la thèse de Selma Khouri [86]. Nous commençons par illustrer le problème de l'hétérogénéité des besoins.

3.3.2.1 Exemples de besoins hétérogènes et vue d'ensemble de notre approche d'intégration

La figure 3.2 présente un exemple de besoins hétérogènes. Dans cet exemple, nous considérons la conception d'un système de gestion de notes basé sur une ontologie du domaine universitaire, comme celle proposée dans le banc d'essai *LUBM* [42]. En supposant que ce système soit défini par plusieurs partenaires situés dans divers pays, les besoins pourraient être exprimés dans différentes langues naturelles et avec plusieurs formalismes. Ainsi, la situation illustrée sur la figure 3.2 pourrait arriver : trois besoins représentant la même exigence (supprimer les informations de tous les étudiants de plus de 40 ans) sont définis avec des langues et formalismes différents (cas d'utilisation d'UML, MCT Merise et Orienté buts).

Pour régler ces problèmes d'hétérogénéité, nous avons conçu une approche permettant d'intégrer ces besoins. Une vue d'ensemble de cette approche est présentée dans la figure 3.3. Elle repose sur les

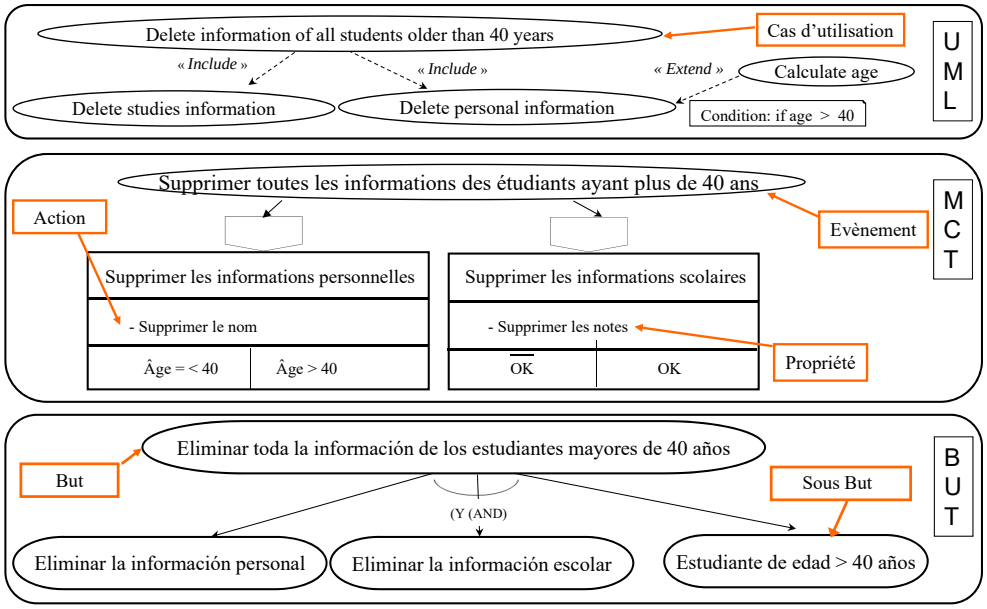


FIGURE 3.2 – Exemple de besoins hétérogènes

éléments suivants qui sont décrits succinctement dans ce qui suit.

1. Des *Ontologie Locales (OL)* sont définies par chaque partenaire à partir d'une *Ontologie Globale (OG)* pour gérer l'hétérogénéité des vocabulaires utilisés (voir section 3.3.2.2).
2. Un formalisme générique, nommé *modèle pivot*, permet d'intégrer des besoins exprimés avec l'un des trois formalismes considérés dans notre exemple (voir section 3.3.2.3).
3. Des règles de transformation convertissent les besoins des différents partenaires en instances du modèle pivot et les lient à l'OG (voir section 3.3.2.4).

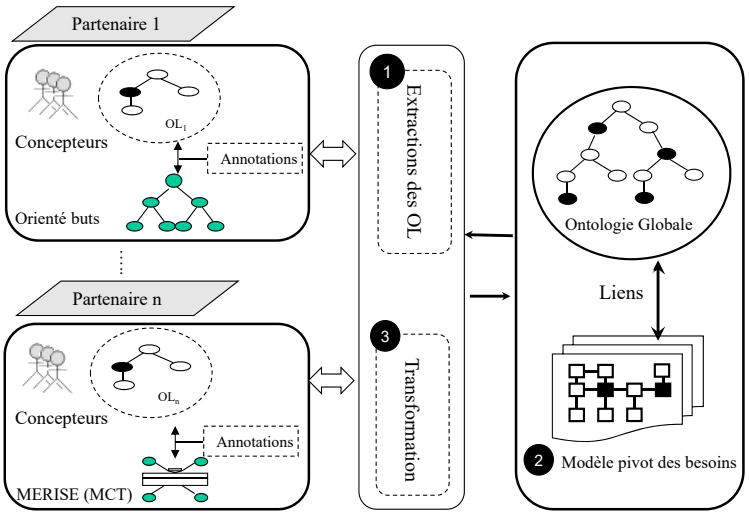


FIGURE 3.3 – Vue d'ensemble de notre approche d'intégration de besoins hétérogènes

3.3.2.2 Utilisation d'ontologies pour gérer l'hétérogénéité des vocabulaires utilisés

Une ontologie permet de définir des concepts d'un domaine d'étude et les termes qui leur sont associés, éventuellement dans plusieurs langues naturelles. Aussi, nous avons proposé d'utiliser cette notion pour gérer l'hétérogénéité des vocabulaires utilisés dans la description des besoins. D'autres approches ont également proposé d'utiliser cette notion pour expliciter les connaissances sur lesquelles s'appuient les concepteurs pour définir leurs besoins (par exemple, [99]). À la différence de notre approche, celles-ci ont pour objectif de permettre d'identifier des liens et dépendances entre des composants logiciels. Notre approche repose sur les deux hypothèses suivantes.

- Une OG existe sur le domaine ciblé par l'application en cours de construction.
- Chaque partenaire extrait une OL à partir de l'OG. Nous avons envisagé trois scénarios d'extraction : (i) l'OL est l'OG, (ii) l'OL est incluse dans l'OG et, pour donner plus d'autonomie aux partenaires, (iii) l'OL étend l'OG. Dans ce dernier scénario, notre approche laisse la possibilité aux partenaires de définir de nouvelles classes dans leurs OL tant qu'elles sont liées à celles de l'OG par une relation d'héritage. De nouvelles propriétés peuvent aussi être ajoutées à ces classes.

Exemple. La figure 3.4 présente un exemple d'OL (partie basse de la figure) construite à partir d'une OG (partie haute). Lorsqu'un partenaire définit son OL, il importe de l'OG les classes et propriétés qui couvrent les notions utilisées pour exprimer ses besoins. Dans cet exemple, les classes `Person`, `Student` et `Employee` sont ainsi importées avec leurs propriétés. Si ces éléments ne sont pas suffisants, un partenaire peut spécialiser les classes importées. C'est le cas, dans cet exemple, où les classes `UnderGraduateStudent` et `GraduateStudent` ont été ajoutées pour pourvoir utiliser les notions correspondantes. Des propriétés peuvent également être ajoutées à ces classes (par exemple, `Degree` sur la classe `GraduateStudent`).

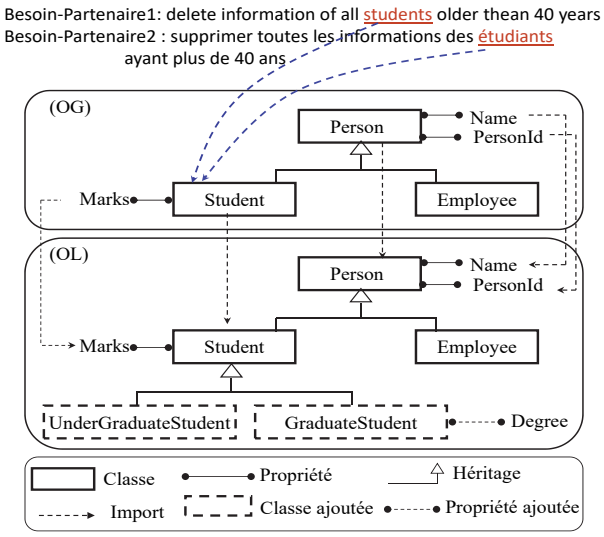


FIGURE 3.4 – Exemple de besoins annotés par une ontologie

Une fois que les OL ont été définies, les besoins de chaque partenaire sont annotés avec les classes et propriétés de ces différentes ontologies. Ce processus est semi-automatique. Notre approche cherche

d'abord des correspondances entre les termes utilisés pour décrire les besoins et ceux associés aux différents concepts des ontologies. Pour chacune de ces correspondances, une annotation est créée qui doit ensuite être validée par le concepteur du besoin annoté. Celui-ci a aussi la charge d'ajouter les annotations manquantes.

Exemple. Dans la figure 3.4 deux besoins (`Besoin-Partenaire1` et `Besoin-Partenaire2`) sont définis de manière informelle (dans notre approche, ces besoins sont exprimés dans l'un des formalismes considérés). Ils représentent la même exigence mais sont écrits dans des langues différentes. Grâce aux annotations entre les termes « students », « étudiants » et la classe `Student`, notre approche identifie que ces deux termes correspondent à la même notion.

Cet exemple illustre l'idée principale de notre approche pour gérer l'hétérogénéité des vocabulaires utilisés dans la description des besoins. Nous considérons maintenant l'hétérogénéité des formalismes utilisés pour leur définition.

3.3.2.3 Proposition d'un modèle pivot pour gérer l'hétérogénéité des formalismes utilisés

Comme de nombreux formalismes existent pour la définition des besoins, nous avons considéré les trois formalismes de notre exemple qui sont souvent utilisés en pratique (cas d'utilisation d'UML, MCT Merise et Orienté buts). Nous les avons étudiés pour identifier les principales notions qu'ils ont en commun. Ainsi, ces formalismes contiennent les notions d'*acteurs* créant des *besoins*. Ceux-ci sont définis par un ensemble d'*actions*, s'appliquent quand des *critères* sont remplis et aboutissent à des *résultats*. Enfin, des *relations* peuvent être établies entre les besoins (par exemple, une relation d'inclusion).

En nous basant sur ces notions principales, nous avons défini un formalisme générique pour la définition des besoins, nommé le *modèle pivot*. Techniquement, celui-ci consiste en la fusion des métamodèles des trois formalismes considérés. En le définissant, notre objectif était que chaque besoin, défini par un partenaire dans l'un de ces formalismes, puisse être représenté comme une instance de ce modèle. La figure 3.5 (b) présente un extrait de ce modèle de pivot. Chaque besoin (`Requirement`) est décrit par un identifiant (`IdReq`), un nom (`NameReq`), une description textuelle (`DescriptionReq`), un objectif (`PurposeReq`), un contexte (`ContextReq`) et une priorité (`PriorityReq`). Comme expliqué précédemment, ils sont définis par des acteurs (`Actor`) et décrits par des actions (`Action`), critères (`Criteria`), résultats (`Result`) et leur liens avec les autres besoins (`Relationship`). La définition formelle de ce modèle est disponible dans [100].

3.3.2.4 Transformation des besoins selon le modèle pivot et liens avec l'OG

En nous basant sur notre étude des trois formalismes considérés, nous avons défini un ensemble de règles pour transformer automatiquement des besoins exprimés dans l'un de ces formalismes en instances du modèle pivot. Comme il existe de nombreux formats et outils pour ces formalismes, nous avons d'abord défini un métamodèle pour chacun d'eux. Puis, nous avons défini nos règles de transformation sur ces métamodèles. Elles prennent en entrée une instance d'un de ces métamodèles et produisent en sortie une instance du modèle pivot.

Nous avons vu précédemment que notre approche pour gérer l'hétérogénéité des vocabulaires repose

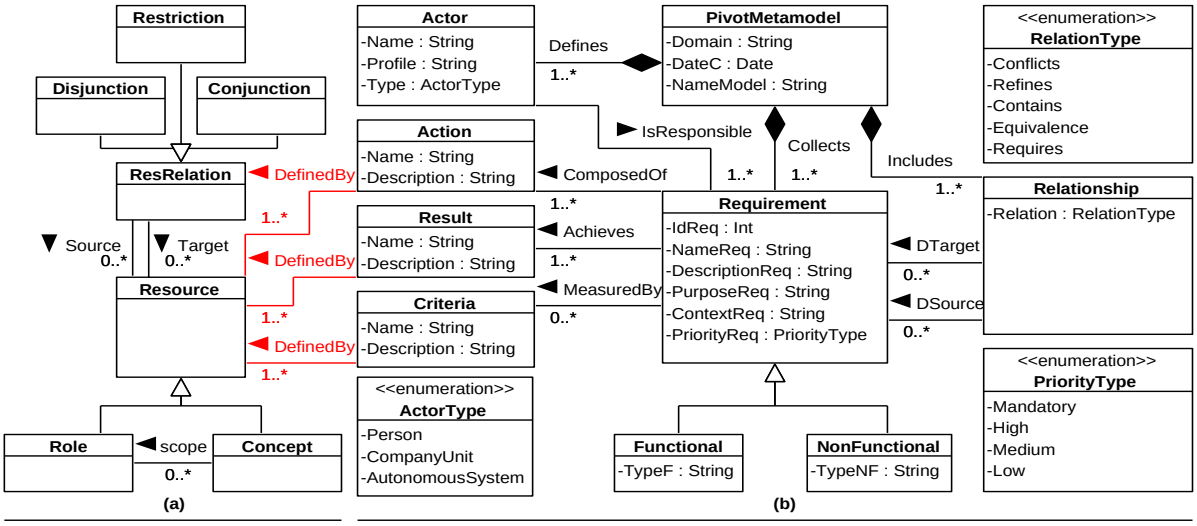


FIGURE 3.5 – Lien entre le modèle d'ontologies et le modèle pivot

sur l'annotation des besoins par les classes et propriétés des OL. Aussi, nous avons défini des règles pour exporter ces annotations en termes du modèle pivot et de l'OG. Cela nécessitait d'étendre le modèle pivot pour que ses instances puissent être annotées par des classes et des propriétés d'une ontologie. Pour cela, nous avons établi des liens entre le modèle d'ontologies utilisé et le modèle pivot. La figure 3.5 illustre ces liens. Le modèle d'ontologies est présenté dans la partie gauche (a) de la figure tandis que le modèle pivot est dans la partie droite (b). Ces liens consistent à associer des caractéristiques des besoins (`Action`, `Result` et `Criteria`) avec le constructeur `Resource` du modèle d'ontologies qui permet de créer des classes (`Concept`) ou propriétés (`Role`). Le modèle complet est nommé *OntoReq*. Ses instances sont des besoins intégrés dont les caractéristiques (`Action`, `Result` et `Criteria`) sont annotées avec des classes et des propriétés de l'OG pour unifier les vocabulaires utilisés dans leur définition.

3.3.3 Résultats

Inspiré par les approches d'intégration de données basées sur les ontologies, nous avons proposé une démarche pour intégrer des besoins hétérogènes définis par différents partenaires, ce qui est le cas dans le contexte des entreprises étendues. Ainsi, pour la gestion de l'hétérogénéité des vocabulaires utilisés pour décrire les besoins, notre approche repose sur la présence d'une OG sur le domaine ciblé par l'application en cours de conception. Cette hypothèse peut sembler forte. Cependant, elle est réaliste dans des domaines de l'ingénierie tels que l'aéronautique ou l'automobile qui comprennent de nombreuses entreprises étendues ainsi que des ontologies standardisées comme *Electronic Components* (IEC 61360-4), *Machining Tools* (ISO 13399) ou *Mechanical Fasteners* (ISO 13584-511).

Concernant l'hétérogénéité des formalismes utilisés pour la définition des besoins, notre approche repose sur un modèle pivot qui contient les notions communes à trois principaux formalismes ainsi que leurs spécificités. Nous avons défini des règles pour pouvoir automatiquement transformer les besoins de chaque concepteur en instances de ce modèle. La formalisation de cette approche est décrite dans [98]. Nous y avons également présenté les deux contributions suivantes.

- Deux approches pour *raisonner* sur les besoins intégrés afin, d'une part, de pouvoir en vérifier la consistance et, d'autre part, de déduire des relations entre ces besoins (par exemple, l'équivalence ou l'inclusion). Dans la première approche, nommée *ship whole*, chaque partenaire envoie l'ensemble de ses besoins au *partenaire intégrateur* qui réalise le raisonnement global sur l'ensemble de tous les besoins. Nous avons montré expérimentalement que cette approche passe difficilement à l'échelle quand le nombre de besoins est important. Aussi, nous avons développé une deuxième approche qui s'avère plus efficace, appelée *reason as needed*. Dans celle-ci, le raisonnement est d'abord fait sur les besoins de chaque partenaire pour éliminer des besoins redondants ou contradictoires. Puis, il est fait globalement sur les besoins restants.
- Une démarche de sélection de structures d'optimisation basée sur les besoins. L'intuition sous-jacente est que de nombreux besoins sont proches des requêtes qui seront exécutées sur le SGD en cours de conception. Aussi, plutôt que d'attendre qu'un administrateur choisisse des structures d'optimisation pour le SGD à partir des requêtes fréquentes exécutées, notre démarche propose des structures d'optimisation dès que la phase d'intégration des besoins est terminée. Le but de cette démarche n'est pas de remplacer la phase physique, qui est toujours nécessaire, mais d'alléger et d'anticiper cette phase dès le recueil des besoins. Nous avons montré expérimentalement l'intérêt de notre approche pour la sélection de différentes structures d'optimisation [100].

Concernant la mise en œuvre de nos approches, nous avons utilisé la BDS *OntoDB/OntoQL* pour persister le modèle *OntoReq* présenté dans la figure 3.5. Cette BDS contenant déjà les principaux constructeurs des modèles d'ontologies, nous avons dû créer le modèle pivot dans le métaschéma d'OntoDB et établir des liens entre ce modèle et les constructeurs de classes et de propriétés d'OntoDB. Les autres fonctionnalités de notre approche, comme l'annotation semi-automatique des besoins par les OL ou le raisonnement sur des besoins, ont été codées en Java dans un outil appelé *OntoReqTool*. La description de cet outil est disponible dans [98].

3.4 Contribution 2 : conception des BDS comme une Ligne de Produits

Dans la section précédente, nous nous sommes intéressés à l'évolution *verticale* du processus de conception des BDS en proposant une phase d'intégration de besoins. La diversité des BDS entraîne également une évolution *horizontale* de ce processus. Cela signifie que, pour chaque phase, plusieurs choix de conception sont possibles. Aussi, nous avons proposé une approche de conception des BDS qui prend en compte cette caractéristique. Nous commençons par détailler la motivation de ces travaux.

3.4.1 Contexte et motivation : processus de conception des BDS

Comme nous l'avons vu au chapitre 2, de nombreuses BDS ont été conçues pour persister des ontologies. Ces BDS présentent une forte diversité sur différents aspects tels que le modèle d'ontologies supporté (PLIB, RDFS, etc.), l'architecture mise en œuvre (deux, trois ou quatre parties), le modèle de stockage utilisé (modèle binaire, vertical ou horizontal), le langage de requêtes implémenté (SPARQL, OntoQL, etc.) ou les techniques d'optimisation sélectionnées (index, vues matérialisées, etc.). Cette diversité résulte de la variété de besoins auxquels répondent ces différentes BDS (par exemple, les BDS qui ont une architecture quatre parties répondent au besoin d'évolution du modèle d'ontologies supporté).

Cette diversité complexifie cependant la tâche des concepteurs qui souhaitent définir une nouvelle BDS ou en sélectionner une qui répond à leurs besoins. Cette tâche nécessite, en effet, de plus en plus d'expertise, de temps et de ressources au fur et à mesure que cette diversité augmente. Nous notons qu'avec le développement de nouveaux systèmes de BD (par exemple, les BD NoSQL [101] ou NewSQL [102]), qui présentent également une grande diversité, cette problématique n'est pas spécifique aux BDS. Dans le contexte général des SGD, différents travaux ont été menés pour faciliter la conception de tels systèmes [63, 79]. Ils ont conduit à la définition de différents outils et méthodes.

- *Les outils CASE* [12] [63] : ils permettent de concevoir un modèle conceptuel du SGD, en supportant généralement plusieurs formalismes, puis de le traduire en modèle logique et de l'implanter dans une plateforme cible.
- *Les patrons de conception (design pattern)* [79–81] : ils consistent en de bonnes pratiques qui ont été identifiées et formalisées pour les phases conceptuelles et logiques des BD.
- *Les bancs d'essai, advisor et test* [42, 82, 103] : ils sont essentiellement utilisés lors de la phase physique. Les *bancs d'essais* permettent d'évaluer la performance d'une conception réalisée [42], les *advisors* proposent des structures d'optimisation à créer [82] et les approches de *test* permettent de vérifier si une conception répond à des besoins identifiés [103].

Nous constatons, d'une part, que ces approches se focalisent sur un sous-ensemble des phases de conception d'un SGD alors qu'elles sont toutes liées. D'autre part, ces approches ne proposent pas une représentation explicite des différents choix possibles dans ce processus de conception. C'est pourtant nécessaire pour des SGD tels que les BDS qui présentent une forte diversité. Pour répondre à ces limitations, nous nous sommes intéressés au domaine de l'*Ingénierie Logicielle* où le problème de la conception d'un produit appartenant à une famille présentant une forte diversité est étudié. Ce problème est connu sous le nom de la *gestion de la variabilité*. Celle-ci se définit comme la capacité d'un système à s'adapter, à se configurer et à se spécialiser en fonction du contexte de son utilisation [104]. La gestion de la variabilité a ainsi pour objectif de concevoir un produit personnalisé, en se basant sur la réutilisation et configuration des composants communs et variables de la famille du produit. Ceux-ci doivent être préalablement définis et implémentés en s'appuyant sur les techniques et outils des *Lignes de Produits (LDP)*.

Bien que les SGD soient différents par leurs données et requêtes, ces systèmes sont conçus selon le même processus composé de plusieurs phases. Dans chacune d'entre elles, des variantes sont possibles (par exemple, dans la phase logique, les tables peuvent être normalisées selon une des formes normales existantes). Les SGD ont donc des caractéristiques communes et se distinguent les uns des autres lorsque différentes variantes ont été choisies lors de leur conception. Réalisant ainsi, qu'un SGD peut-être vu comme une famille de produits à concevoir, nous avons proposé d'adapter les techniques et outils des LDP pour la conception de tels systèmes. Notre objectif général est ainsi de réduire la complexité du processus de conception d'un SGD. Plus précisément, nous visions principalement à :

- représenter explicitement le processus de conception d'un SGD avec toutes ses variantes ;
- établir des liens entre les différentes phases de conception d'un SGD pour permettre au concepteur de faire des choix de conception cohérents et pertinents ;
- aider le concepteur à comparer différentes variantes du SGD construits pour pouvoir identifier celles qui correspondent le mieux à ses besoins.

12. Par exemple, PowerDesigner ou erwin Data Modeler pour les BD

3.4.2 Principe général de notre démarche de conception basée sur la variabilité

Nous avons défini une démarche de conception, basée sur la variabilité, pour différents SGD tels que les BD, les entrepôts de données ou les BDS. Elle couvre leurs principales étapes de conception : conceptuelle, logique et physique [78]. Pour illustrer cette approche, nous prenons l'exemple de la conception logique d'une BDS implémentée dans une BDR. Supposons ainsi qu'un concepteur souhaite définir une nouvelle BDS en s'appuyant sur les approches existantes dans la littérature. Dans la phase logique, il doit définir l'ensemble des tables de cette BDS qui seront utilisées pour stocker les ontologies. Notre démarche de conception s'appuie sur les formalismes et outils des LDP pour expliciter les différents choix possibles. Elle suit trois principales étapes décrites succinctement dans ce qui suit.

- La représentation explicite des différents choix de conception possibles (voir section 3.4.2.1).
- L'identification des dépendances entre ces choix de conception et l'implémentation des fonctionnalités correspondantes (voir section 3.4.2.2).
- La dérivation d'une configuration valide pour la BDS conçue (voir section 3.4.2.3).

3.4.2.1 Modélisation de la variabilité de la conception logique des BDS

En suivant la démarche des LDP, notre approche propose une description de la variabilité du produit à concevoir. Elle est le résultat d'une étude visant à identifier les caractéristiques communes et variables pour la famille de produits considérés (dans notre exemple, les BDS). Cette description consiste en un *Modèle de Features (MF)* [105] qui permet de visualiser graphiquement la variabilité du produit en cours de conception. Un tel modèle est un arbre où chaque nœud est un *feature*, c'est-à-dire une caractéristique ou fonctionnalité qui est définie comme « une unité logique d'un comportement déterminé par un ensemble d'exigences fonctionnelles et non fonctionnelles » [104]. Les relations suivantes peuvent être établies entre un feature et ses fils dans l'arbre.

- *Obligatoire* ou *Optionel* : utilisables s'il n'y a qu'un feature fils ; précisent si celui-ci est obligatoire ou optionnel.
- *Or* ou *Alternative* : utilisables s'il y a plusieurs features fils. La relation *Or* précise qu'au moins un des features fils doit être sélectionné alors que, pour *Alternative*, il ne peut y en avoir qu'un seul.

Exemple. La figure 3.6 présente un exemple simplifié de notre MF concernant la conception logique d'une BDS.

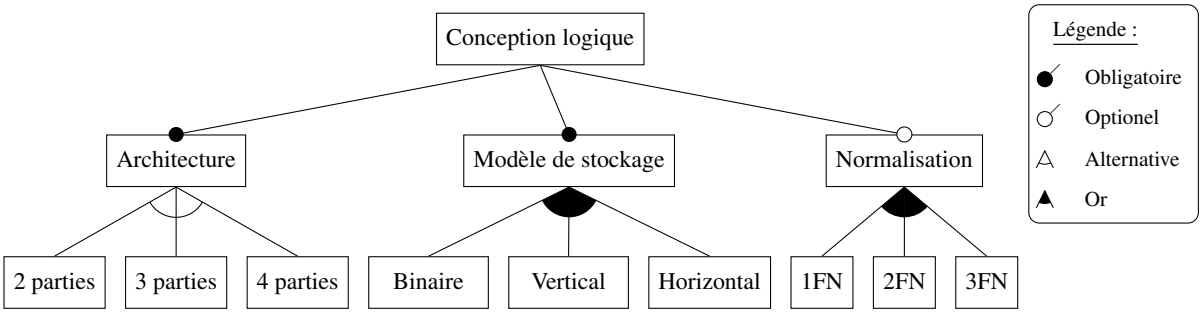


FIGURE 3.6 – Exemple de modèle de features pour la conception logique des BDS

Comme nous l'avons vu au chapitre 2, La variabilité au sein de cette phase peut concerner les features (points de variation) suivants.

- *L'architecture* : une BDS peut contenir deux, trois ou quatre parties, de manière exclusive, pour stocker les ontologies (instances et schémas) et éventuellement le modèle d'ontologies utilisé.
- *Le modèle de stockage* : il définit les tables utilisées par la BDS pour l'enregistrement des données. Différents modèles de stockage existent dans la littérature, dont les modèles binaire, vertical et horizontal, qui peuvent être utilisés conjointement.
- *La normalisation* : de manière optionnelle, les tables utilisées dans la BDS peuvent être normalisées selon une ou plusieurs formes normales données.

Une formalisation des MF ainsi produits est proposée dans [78]. Nous décrivons maintenant comment nous avons établi des dépendances entre les features de ces modèles.

3.4.2.2 Identification des dépendances entre features et implémentation

Toutes les combinaisons de features présentes dans un MF ne sont pas forcément valides. Dans notre contexte, cela signifie qu'elles ne correspondent pas à une démarche de conception d'une BDS valide. En effet, certains choix peuvent être incompatibles entre eux, d'autres sont liés. Aussi, afin de préciser les combinaisons valides, notre approche inclut des contraintes de cohérence et plus généralement des *dépendances* entre features. Nous avons pris en considération deux types de dépendances.

- *Strictes*, exprimées à l'aide des relations *exige* et *exclut*. Seules des BDS respectant ces contraintes pourront être définies.
- *Flexibles*, exprimées à l'aide des relations *avantage* et *désavantage*. Ces liens ne sont utilisés que pour conseiller et assister les concepteurs pendant le processus de définition d'une BDS.

Exemple. Dans le MF présenté sur la figure 3.6, la normalisation n'a pas de sens dans les modèles de stockage binaire et vertical puisque ces approches se basent sur des tables binaires ou une table de triplets. Nous avons donc les deux relations suivantes : ⟨*Binaire* ***exclut*** *Normalisation*⟩ et ⟨*Vertical* ***exclut*** *Normalisation*⟩.

Des liens pourraient aussi être établis avec les features des autres phases. Par exemple, le choix du modèle d'ontologies est une variante de la phase conceptuelle. Nous pourrions ainsi définir le lien suivant : ⟨*PLIB* ***avantage*** *Horizontal*⟩ puisque le modèle PLIB, grâce aux hypothèses de typage faites, facilite l'utilisation du modèle de stockage horizontal, même si un autre pourrait être utilisé.

En plus de la définition de ces dépendances, notre approche consiste à implémenter chaque feature. Ces implémentations vont prendre en entrées des paramètres qui devront être fournis par le concepteur ou par la sortie d'un autre feature.

Exemple. Dans notre exemple simplifié de MF (voir figure 3.6), l'implémentation du feature *4 parties* prend en entrée le modèle d'ontologies choisi lors de la phase conceptuelle et la BDR devant supporter la BDS. Elle produit en sortie les tables des parties *métaschéma* et *schéma* à créer dans la BDR choisie. Elle retourne également les lignes à insérer dans la partie *métaschéma* pour que celui-ci stocke le modèle d'ontologies choisi.

L'implémentation du feature *Binaire* prend en entrée une ontologie à charger dans la BDS ainsi que les éventuelles tables de la partie *schéma*, si une architecture trois ou quatre parties a été sélectionnée. Elle retourne les tables binaires à créer dans la partie *instance* pour stocker les instances de l'ontologie. Elle retourne également les lignes à créer pour le stockage de l'ontologie.

Nous montrons maintenant comment un concepteur peut exploiter le MF fourni et son implémentation pour obtenir une BDS.

3.4.2.3 Dérivation d'une configuration valide pour la conception logique d'une BDS

À partir du MF fourni, le concepteur va dériver une *configuration* de la BDS qu'il souhaite concevoir. Pour cela, il va sélectionner les features qui correspondent à ses exigences. Les dépendances flexibles définies entre les features vont l'aider à définir sa configuration tandis que les strictes vont l'obliger à en définir une valide. Notre approche est ainsi outillée pour que, lorsque le concepteur choisit un feature, tout ceux qui sont incompatibles avec ce choix ne puissent plus être sélectionnés.

Exemple. La figure 3.7 présente une configuration possible pour la conception logique d'une BDS. Dans cet exemple, l'utilisateur a choisi de concevoir une BDS *quatre parties* utilisant le modèle de stockage *horizontal* pour la persistance des instances et en faisant en sorte que les tables correspondantes soient en 3ème forme normale. La combinaison des implémentations des différents features sélectionnés permet d'obtenir un script de création des tables de cette BDS.

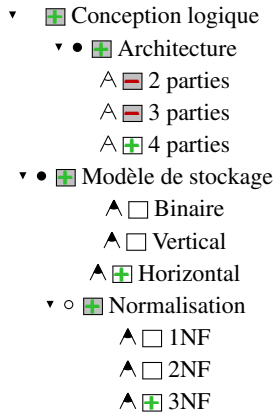


FIGURE 3.7 – Exemple de configuration possible pour la conception logique d'une BDS

L'étape de dérivation définie dans les LDP consiste à exécuter la configuration établie par l'utilisateur. Cependant, dans le contexte des BDS et plus généralement des SGD, de nombreuses configurations sont possibles et il peut être difficile d'évaluer celle qui correspond le mieux à un ensemble de besoins. Aussi, afin d'aider l'utilisateur dans le choix d'une configuration, nous avons proposé une approche permettant de comparer les différentes configurations possibles. Cette comparaison est faite par rapport à différents besoins non fonctionnels tels que minimiser le temps d'exécution d'une charge de requêtes ou réduire l'espace de stockage requis [78].

Exemple. Concernant les BDS, pour mettre en œuvre cette idée, nous avons développé l'outil OntoD-

Bench [13] [106] dont le principe est illustré sur la figure 3.8.

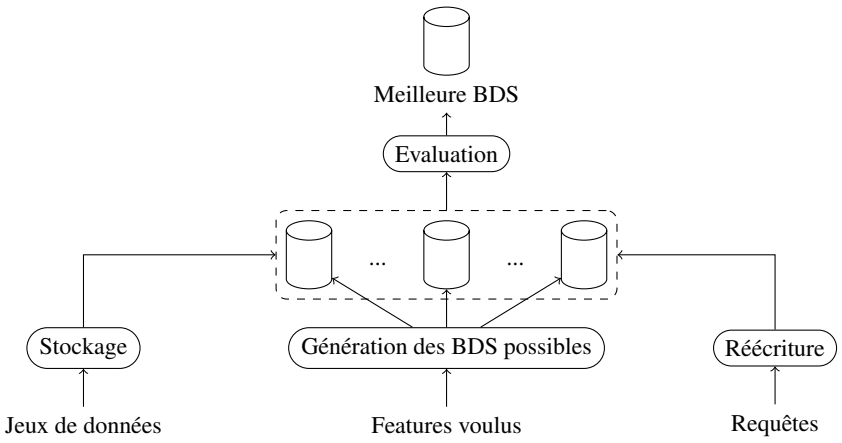


FIGURE 3.8 – Principe de la plateforme OntoDBench

Supposons que le concepteur souhaite déterminer la conception logique d'une BDS qui minimise le temps d'exécution d'une charge de requêtes. Celui-ci va pouvoir préciser s'il souhaite que certains features apparaissent absolument dans les configurations testées (par exemple, l'architecture *quatre parties*). A l'aide du MF défini et implémenté précédemment, l'outil OntoDBench va générer les différentes configurations possibles qui incluent les features voulus. Il va aussi les charger avec le jeu de données fourni par le concepteur. Pour pouvoir évaluer la performance des requêtes utilisateur, l'outil OntoDBench inclut un module de réécriture des requêtes utilisateur selon les trois modèles de stockage possibles. Ainsi, il peut exécuter la charge de requêtes utilisateur sur les différentes BDS testées pour pouvoir identifier celle qui fournit les meilleures performances.

3.4.3 Résultats

Constatant que les concepteurs de BDS, et plus généralement des SGD, font face à des choix lors de la conception d'un tel système, nous avons proposé une méthodologie pour gérer la variabilité de ce processus. Celle-ci permet de voir l'ensemble des phases de conception d'une BDS comme une LDP. Contrairement aux approches de l'état de l'art, qui se focalisent généralement sur la variabilité d'une seule phase de conception, notre approche explicite les choix à réaliser tout au long de ce processus au travers de MF. Par ailleurs, le caractère *holistique* de notre approche permet de considérer les interdépendances qui existent entre les phases de ce processus afin d'aider le concepteur à définir une configuration valide pour la BDS en cours de construction. En allant plus loin que ce qui est habituellement proposé par les LDP, notre démarche permet d'évaluer différentes configurations possibles pour identifier celles qui répondent le mieux à des besoins non fonctionnels du concepteur. Cette approche a été formalisée dans [78].

La mise en œuvre de notre approche repose sur la définition de MF avec la plateforme *FeatureIDE* [14]. Celle-ci permet de réaliser l'implémentation des features en Java. Notre approche ayant été définie de

13. https://forge.lias-lab.fr/projects/ontodbench
14. http://www.featureide.com

manière générale pour les SGD, nos MF comprennent de nombreux features que nous n'avons pas tous implémentés. Concernant les BDS, nous nous sommes ainsi focalisés sur la phase logique en proposant l'outil OntoDBench dont l'implémentation est disponible sur https://forge.lias-lab.fr/projects/ontodbench. Nous avons montré que cet outil peut être utilisé pour identifier la conception logique d'une BDS qui répond le mieux à un besoin de performance d'un ensemble de requêtes [106].

3.5 Contribution 3 : optimisation des BDS, le cas des vues matérialisées

Dans la section précédente, nous avons proposé une démarche générale pour représenter la variabilité dans le processus de conception des BDS. Dans cette section, nous nous focalisons sur la conception physique des BDS et plus particulièrement sur la sélection des vues matérialisées. Notre objectif est à nouveau de prendre en compte la diversité des BDS. Par rapport à notre démarche générale, les approches que nous proposons dans cette section peuvent être vues comme des implémentations d'un feature de *sélection des vues matérialisées* qui prend en compte la diversité des modèles logiques de BDS.

3.5.1 Contexte et motivation : sélection des vues matérialisées pour les BDS

Dans la phase physique de conception d'un SGD, des structures d'optimisation, telles que des index ou des vues matérialisées, sont sélectionnées pour optimiser le temps de traitement d'une charge de requêtes. Le processus de sélection est souvent guidé par un *modèle de coût* estimant le coût des requêtes, généralement en termes d'entrées-sorties, en présence ou non de ces structures. Le problème revient alors à rechercher l'ensemble des structures d'optimisation qui minimise le coût d'exécution d'une charge de requêtes considérée tout en respectant des contraintes liées, par exemple, à l'espace de stockage ou au coût de maintenance des structures sélectionnées. Ce problème est généralement difficile car l'espace de recherche, basé sur l'ensemble des structures d'optimisation candidates, peut être large. C'est le cas pour la sélection des vues matérialisées où chaque requête de la charge considérée ainsi que toutes les parties de celles-ci peuvent être candidates à la matérialisation.

Nous avons considéré le problème de sélection des vues matérialisées dans le contexte des BDS. Comme nous l'avons vu précédemment, une caractéristique des BDS est qu'elles présentent une diversité, notamment en termes de modèles logiques utilisés. Dans les approches de sélection des vues matérialisées conçues pour les BD [107–109], cette caractéristique n'est pas prise en compte : elles se basent sur un modèle logique figé. C'est également le cas des approches proposées dans le contexte des BDS [83–85] qui supposent que les données sont stockées dans une table de triplets (modèle de stockage vertical, architecture deux parties). À notre connaissance, au moment de nos travaux, seule l'approche de Dritsou *et al.* [110] avait été conçue pour pouvoir être appliquée à différentes BDS. Trois solutions sont proposées pour cela.

- Définir le modèle de coût par rapport au nombre de nœuds du graphe RDF que les vues proposées évitent de parcourir. Cependant, suivant le modèle logique utilisé par la BDS, le nombre de nœuds évités implique une réduction du coût plus ou moins importante, ce qui n'est pas pris en compte.
- Utiliser comme modèle de coût le temps d'exécution réel des requêtes. Cela suppose que ces dernières ont déjà été exécutées, ce qui ne sera pas toujours le cas dans la pratique.

– Se baser sur les modèles de coût des BDS considérées. Cependant, ceux-ci ne sont pas toujours accessibles, ce qui est souvent le cas des BDS propriétaires. De plus, les coûts estimés par ces modèles sont difficilement comparables. En conséquence, ils ne peuvent pas être utilisés pour comparer des BDS afin d'en choisir une qui soit adaptée à des besoins donnés.

Pour répondre à ces limitations et ainsi prendre en compte la diversité des BDS dans la sélection des vues matérialisées, nous nous sommes fixés les objectifs suivants :

– définir un modèle de coût générique s'appliquant à différentes BDS et prenant en compte leurs particularités ;
– proposer des approches de sélection des vues matérialisées basées sur ce modèle de coût : l'une basée sur le fait que la plupart des requêtes exécutées sur des BDS ont une forme particulière [111] et l'autre sur le fait que l'interaction entre ces requêtes est forte [112].

3.5.2 Principe général de notre approche de sélection des vues matérialisées

Pour pouvoir comparer nos approches de sélection des vues matérialisées à celles proposées dans la littérature, nous les avons développées dans le contexte RDF/SPARQL. Dans ce qui suit, nous définissons les notions utilisées dans nos travaux.

3.5.2.1 Définition du problème traité

Nous considérons les *requêtes RDF* définies comme une *conjonction de patrons de triplet* : $Q = t_1 \wedge \cdots \wedge t_n$, c'est-à-dire des triplets RDF pouvant contenir des variables. Cette définition correspond aux requêtes *Basic Graph Pattern* de SPARQL. Ce type de requête est celui considéré dans la plupart des autres travaux de l'état de l'art, à l'exception de ceux de Ibragimov *et al.* [85] qui considèrent des requêtes analytiques. Pour que nos approches puissent s'appliquer à d'autres langages de requêtes, dont OntoQL [7], nous avons imposé que le prédicat de chaque patron de triplet soit une constante. Nous avons aussi supposé que ces requêtes sont exécutées sur des *BDS saturées*, c'est-à-dire des BDS qui contiennent les triplets RDF *implicites* pouvant être dérivés grâce à des règles de raisonnement.

Problème. *Étant donné (i) une BDS B ayant une architecture et un modèle de stockage choisis parmi ceux identifiés précédemment (voir figure 3.6), (ii) une charge de requêtes RDF Q, où chaque requête a une fréquence d'accès, et (iii) une contrainte de stockage S. Le problème que nous abordons consiste à identifier un ensemble de vues qui minimise le coût d'exécution de Q sur B tout en respectant la contrainte de stockage S.*

Nous commençons par présenter nos deux approches d'identification de vues candidates. Nous verrons ensuite comment nous faisons la sélection parmi l'ensemble de ces vues.

3.5.2.2 Identification des vues candidates : Class-Based Approach (CBA)

Notre première approche pour identifier des vues candidates, nommée *Class-Based Approach (CBA)*, se base sur l'hypothèse qu'un schéma est défini pour l'ontologie considérée et qu'en particulier les domaines des propriétés sont définis. Elle est basée sur l'observation que les requêtes RDF font souvent

appel à plusieurs propriétés liées à une même entité, elles tendent donc à être, sinon à contenir, des *requêtes en étoile* (c'est-à-dire, des requêtes dont tous les patrons de triplet ont le même sujet) [111]. Quand une même variable `?X` apparaît comme sujet dans plusieurs patrons de triplet, des jointures sont nécessaires dans la plupart des modèles de stockage. Notre approche consiste donc à matérialiser la classe de `?X` avec ses propriétés qui sont utilisées dans les requêtes. Ainsi, ces patrons de triplet seront exécutés sur cette vue et le nombre de jointures sera réduit.

Pour mettre en œuvre cette idée, notre approche CBA suit les étapes suivantes.

- Identifier les classes et propriétés impliquées dans la requête. Nous utilisons pour cela les patrons de triplet basés sur le prédicat `type` ainsi que le ou les domaines des propriétés pour ceux qui ne sont pas de cette forme.
- Définir une *matrice d'usage* des classes et des propriétés à partir de l'étape précédente. Ces matrices sont binaires, elles indiquent si une classe ou propriété est utilisée dans une requête de Q.
- Sélectionner les vues candidates : ce sont les vues correspondant aux classes et à leurs propriétés utilisées par au moins une requête de Q ; elles peuvent donc être extraites des matrices d'usage.

Exemple. Soit Q une charge composée des requêtes RDF suivantes [15] :

`Q1(X,Y1,Y2): ?X type Univ ∧ ?X member ?Y1 ∧ ?Y1 workFor Dept1 ∧ ?Y1 phone ?Y2`
`Q2(X,Y2,Y3): ?X type Professor ∧ ?X workFor ?Y1 ∧ ?Y1 alumnus ?Y2 ∧ ?Y1 member ?Y3`
`Q3(X,Y1,Y2): ?X type Professor ∧ ?X workFor Dept1 ∧ ?X name ?Y1 ∧ ?X phone ?Y2`

Pour chacune des requêtes, nous identifions le ou les classes de chaque variable apparaissant comme sujet d'un patron de triplet. Par exemple, pour `Q1`, la classe de `?X` est `Univ`, qui est identifiée grâce au prédicat `type` et au domaine de la propriété `member` ; la classe de `?Y1` est `Professor` qui est identifiée en utilisant les domaines de `workFor` et `phone`. Nous conservons aussi les propriétés utilisées pour chacune de ces classes. Nous définissons ainsi les matrices d'usage suivantes :

Matrice d'usage des classes		
Requêtes	`Univ`	`Professor`
`Q1`	1	1
`Q2`	1	1
`Q3`	0	1

Matrice d'usage des propriétés					
Requêtes	`member`	`workFor`	`phone`	`alumnus`	`name`
`Q1`	1	1	1	0	0
`Q2`	1	1	0	1	0
`Q3`	0	1	1	0	1

À partir des matrices d'usage, nous constatons que deux classes sont utilisées : `Univ` et `Professor`. Nous identifions donc deux vues candidates qui correspondent à ces deux classes et à leurs propriétés qui sont utilisées dans au moins une requête de Q. Ces vues candidates sont les suivantes :

`V1(X,Y1,Y2): ?X type Univ ∧ ?X member ?Y1 ∧ ?X alumnus ?Y2`
`V2(X,Y1,Y2,Y3): ?X type Professor ∧ ?X workFor ?Y1 ∧ ?X phone ?Y2 ∧ ?X name ?Y3`

Même si ce n'est pas montré dans cet exemple, notre approche prend en compte les hiérarchies de classes et de propriétés ainsi que le fait qu'une propriété puisse avoir plusieurs domaines. En conséquence, de nombreuses vues peuvent être candidates [113]. Nous présentons à présent une autre approche d'identification de vues candidates qui, contrairement à CBA, prend en compte les sélections faites dans les patrons de triplet.

15. Pour simplifier, nous utilisons des noms au lieu des URI

3.5.2.3 Identification des vues candidates : Triple-Based Approach (TBA)

Notre seconde approche, *Triple-Based Approach (TBA)*, consiste à identifier comme vues potentielles des résultats intermédiaires utilisés dans des requêtes de Q. Elle se base sur la notion de *plan unifié de requêtes* initialement proposée dans le contexte relationnel [107]. Un plan unifié est construit à partir de celui de chaque requête de Q. Il peut être vu comme la fusion de ces différents plans.

Exemple. La figure 3.9 présente un plan unifié possible pour les requêtes Q_1 et Q_3 de notre exemple précédent. Les feuilles de ce plan sont les patrons de triplet des requêtes dont les variables sont harmonisées. Pour simplifier, nous n'avons montré ici que les interactions entre les requêtes pour l'opération de jointure. Notre approche prend également en compte les sélections qui sont faites sur chaque patron de triplet et qui dépendent du modèle de stockage utilisé [113].

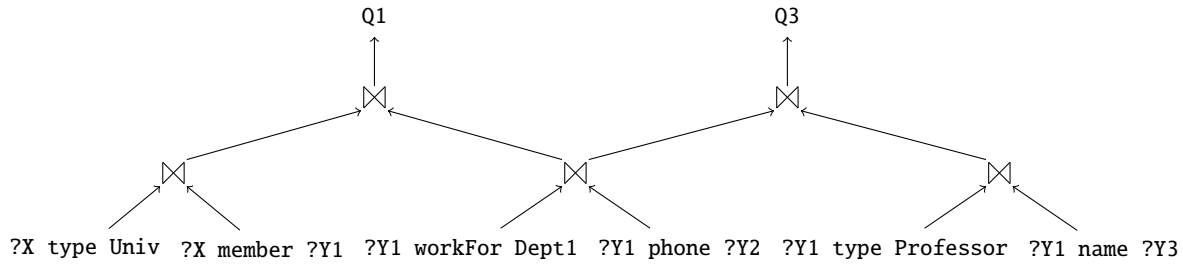


FIGURE 3.9 – Un plan unifié possible des requêtes Q1 et Q3 de notre exemple

De nombreux plans unifiés sont possibles pour une charge de requêtes. Le but est d'identifier celui ayant le coût minimal pour que ses noeuds intermédiaires, qui sont des vues candidates, soient utilisés lors de l'exécution des requêtes de Q. Ce problème étant difficile [107], nous suivons l'approche suivante :

1. optimiser le plan de chaque requête individuellement en triant les patrons de triplet suivant leur sélectivité croissante ;
2. trier les requêtes selon un critère donné (par exemple, selon leurs fréquences d'accès) ;
3. obtenir les plans unifiés correspondant à cette liste de requêtes et à ses permutations circulaires ;
4. retenir le plan de coût minimum (le calcul de ce coût est décrit dans la section suivante).

Ce processus ne garantit pas d'obtenir le plan unifié optimal mais donne de bons résultats en pratique [109]. Tous les nœuds intermédiaires du plan obtenu sont identifiés comme des vues candidates.

Exemple. Supposons que le plan unifié de la figure 3.9 soit celui obtenu par le processus précédent (rappelons qu'il est simplifié car limité à l'opération de jointure). Les vues suivantes seraient candidates :

```
   V1(X,Y1): ?X type Univ ∧ ?X member ?Y1
  V2(Y1,Y2): ?Y1 workFor Dept1 ∧ ?Y1 phone ?Y2
  V3(Y1,Y3): ?Y1 type Professor ∧ ?Y1 name ?Y3
V4(X,Y1,Y2): ?X type Univ ∧ ?X member ?Y1 ∧ ?Y1 workFor Dept1 ∧ ?Y1 phone ?Y2
V5(Y1,Y2,Y3): ?Y1 workFor Dept1 ∧ ?Y1 phone ?Y2 ∧ ?Y1 type Professor ∧ ?Y1 name ?Y3
```

Nous précisons maintenant comment nous retenons, parmi ces vues candidates, celles qui minimisent le coût d'exécution de la charge de requêtes tout en respectant la contrainte de stockage.

3.5.2.4 Modèle de coût générique et respect des contraintes

Pour estimer le coût d'exécution des requêtes et le coût de stockage des vues candidates, nous avons défini un modèle de coût *générique*, dans le sens où il est défini pour les différentes architectures et modèles de stockage de BDS. Il permet ainsi que nos approches puissent s'appliquer aux différents types de BDS et prennent ainsi en compte leur diversité. De manière succincte, ce modèle de coût est conçu selon les principes suivants.

- *Concernant l'architecture des BDS*, notre modèle de coût distingue le coût d'accès aux différentes parties d'une BDS : coût d'accès au métaschéma (pour les architectures quatre parties), coût d'accès au schéma de l'ontologie (pour les architectures trois et quatre parties) et coût d'accès aux données (pour toutes les architectures).
- *Concernant le modèle de stockage*, celui-ci a un impact sur le coût d'accès aux données. De manière simplifiée, notre approche consiste à traduire les requêtes selon les différents modèles de stockage et à utiliser un modèle de coût relationnel pour estimer leurs coûts. Ceux-ci étant dépendants de l'estimation de la taille des résultats intermédiaires, nous avons combiné et adapté les approches d'estimation proposées dans le contexte relationnel [114] et des BDS [115, 116] aux différents modèles de stockage considérés.

Le modèle de coût ainsi défini permet d'estimer l'intérêt d'un ensemble de vues candidates. Cependant, trouver celui qui minimise le coût d'exécution de la charge de requêtes tout en respectant la contrainte de stockage est difficile [108]. Aussi , nous avons proposé de faire cette exploration avec un algorithme génétique. Pour cela, nous avons défini le profit d'une vue V comme étant la différence entre le coût total de la charge de requêtes sans vue et celui si V est matérialisée. Le problème revient à trouver un ensemble de vues dont la somme des espaces occupés est inférieure ou égale à la contrainte de stockage et qui maximise les profits. Plus de détails sur le paramétrage de notre algorithme génétique sont donnés dans [113].

3.5.3 Résultats

L'originalité de nos approches CBA et TBA est de prendre en compte la diversité des BDS à la fois sur leurs architectures et modèles de stockage. Pour cela, elle repose sur un modèle de coût *générique* qui permet d'évaluer le coût d'une requête sur les différentes variantes de BDS. Dans l'approche TBA, la prise en compte de cette diversité est aussi faite dans la construction du plan unifié de requêtes où les sélections réalisées dans les patrons de triplet sont explicitées selon le modèle de stockage utilisé. Grâce à la prise en compte de la diversité des conceptions logiques de BDS, nos approches pourraient être utilisées comme implémentations d'un feature de sélection des vues matérialisées dans l'approche présentée à la section 3.4.

Pour évaluer nos propositions, nous les avons implémentées en Java. En termes d'expérimentations, nous avons commencé par évaluer notre modèle de coût générique. Pour cela, nous avons estimé le coût des requêtes du banc d'essai LUBM [42] pour différents types de BDS avec notre modèle de coût générique. Puis, nous avons exécuté ces requêtes sur six BDS, trois industrielles (Oracle [117], IBM SOR [118] et DB2RDF [119]) et trois académiques (Jena [26], Sesame [36] et OntoDB [2]), qui utilisent différents modèles de stockage et architectures. Nous avons ensuite comparé les résultats théoriques et

expérimentaux pour voir si notre modèle de coût permettait d'identifier les tendances obtenues expérimentalement. Nous avons tiré de cette comparaison les conclusions suivantes [113].

- Pour les BDS Oracle, OntoDB et Sesame, notre modèle de coût confirme les tendances obtenues et, en particulier, le fait que le modèle de stockage binaire est pertinent si peu de propriétés sont utilisées dans les requêtes et inversement pour le modèle horizontal.
- Pour les BDS Jena, SOR et DB2RDF, les tendances ne sont que partiellement confirmées. Nous arrivons ici à la limite de notre approche qui se veut générique, et donc qui ne prend pas en compte les optimisations spécifiques qui peuvent être réalisées sur certaines BDS. Une perspective est ainsi de permettre la spécialisation de notre modèle pour qu'il soit plus précis pour certaines BDS.

Concernant nos approches de sélection des vues matérialisées, nous les avons mises en œuvre sur le banc d'essai LUBM et sur des BDS réelles qui utilisent différents modèles de stockage : vertical (Jena), binaire (Sesame) et horizontal (OntoDB). Nous avons également utilisé une BDS implémentant le modèle vertical, créée spécialement pour les expérimentations, afin de pouvoir comparer nos approches à celle de référence [83]. Nous avons obtenu les résultats suivants [112].

- Nos approches de sélection des vues matérialisées apportent un gain important (un facteur d'environ 2) sur le temps d'exécution total de la charge de requêtes par rapport à une exécution sans vue matérialisée. Globalement, notre approche TBA donne de meilleures performances que CBA. Cela est dû à la prise en compte des sélections faites dans les patrons de triplet.
- Notre approche TBA donne des résultats proches de l'approche de référence [83] sur une BDS spécifique. Son avantage est de pouvoir être exécutée sur d'autres types de BDS.

3.6 Bilan de ces travaux de recherche

Nous tirons à présent un bilan des travaux présentés dans ce chapitre en commençant par mettre en avant leur originalité par rapport à l'état de l'art.

3.6.1 Contributions à l'état de l'art

Nous avons réalisé des contributions dans différents domaines. Nous commençons par considérer nos travaux sur l'Ingénierie des Besoins.

3.6.1.1 Ingénierie des Besoins et conception des BDS

Nous avons d'abord proposé une approche d'intégration de besoins hétérogènes qui peut être liée aux autres phases de conception des BDS. Le tableau 3.1 positionne cette approche (en italique) par rapport à différents travaux de l'état de l'art [87–91]. Nous pouvons constater que ceux-ci se sont principalement focalisés sur le fait de pouvoir combiner plusieurs formalismes de définition de besoins et analyser les besoins produits. Ces approches diffèrent sur le type de formalismes supportés et les méthodes utilisées pour analyser les besoins produits.

Dans notre approche, nous nous sommes focalisés sur l'intégration de besoins hétérogènes et leur uti-

Approche	Formalismes des besoins	Unification Vocabulaire	Analyse des besoins	Liens SGD
Tueno Fotso *et al.* [87]	SysML, Goal	non	méthode B	non
Vicente-Chicote *et al.* [88]	métamodèle spécifique	non	contraintes OCL	non
Brottier *et al.* [89]	spécification textuelle	non	règles de fusion	non
Nguyen *et al.* [90]	Use case, Goal	non	raisonnement	non
Mirbel et Villata [91]	Goal	non	argumentation	non
Boukhari et al. [100]	Uml, Goal, Merise	ontologies	raisonnement	optimisation

TABLE 3.1 – Positionnement de notre approche d'intégration des besoins (en italique)

lisation dans la conception d'un SGD tel qu'une BDS. Aussi, son originalité est double. D'une part, elle permet d'unifier les vocabulaires utilisés pour la définition des besoins en utilisant la couche linguistique d'une ontologie. D'autre part, notre approche permet de persister des besoins annotés par des concepts d'une ontologie dans la BDS OntoDB et de les utiliser pour sélectionner des structures d'optimisation. Ainsi, notre contribution aura été de proposer une phase d'intégration des besoins dans le processus de conception d'une BDS et de montrer que celle-ci peut être utile dans d'autres phases de ce processus.

3.6.1.2 Ligne de Produits Logiciels et conception des BDS

Nous avons ensuite présenté une méthodologie de conception des BDS, et plus généralement des SGD, basée sur les LDP. Le tableau 3.2 compare cette approche (en italique) aux principales approches qui facilitent la conception de tels systèmes. Nous pouvons constater que celles-ci sont spécialisées pour un sous-ensemble des phases de conception d'un SGD (par exemple, les *advisors* se focalisent sur la phase physique). Par ailleurs, elles ne permettent pas d'expliciter les différents choix de conception et les liens qu'ils ont entre eux.

Approche	Phases	Objectifs	Variabilité	Type
Advisor, benchmark, test [42, 82, 103]	Physique	Aide à la décision	Implicite	Outil
Patrons de conception [79–81]	Concept./Logique	Réutilisation	Implicite	Méthode
Outils CASE [63]	Concept./Logique	Automatisation	Implicite	Outil
Basée sur les LDP [78]	Toutes	Configuration et Aide à la décision	Explicite	Méthode et outil

TABLE 3.2 – Positionnement de notre méthodologie de conception des SGD basée sur les LDP (en italique)

En comparaison, nous avons proposé une méthodologie qui vise à expliciter les choix de conception au travers de *modèles de features* issus des LDP. Nous avons montré, dans le contexte des BDS, comment établir des liens entre les features ainsi définis et les implémenter pour faciliter la conception d'un SGD. Notre méthodologie vise également à aider le concepteur à choisir la configuration qui répond le mieux à ces besoins. Dans le contexte des BDS, cela s'est traduit par le développement de l'outil OntoDBench

qui permet de comparer différentes conceptions logiques de BDS.

3.6.1.3 Conception physique des BDS

Concernant la conception physique des BDS, notre contribution a été de définir un modèle de coût qui prend en compte la diversité des BDS et de proposer deux approches de sélection des vues matérialisées basées sur ce modèle. Le tableau 3.3 compare ces deux approches par rapport à d'autres de l'état de l'art. Concernant le type de BDS, nous pouvons constater que seules nos approches et celle de Dritsou *et al.* [110] sont conçues pour différentes BDS. Cependant, à l'inverse de nos approches, cette dernière ne prend pas en compte les spécificités de chaque modèle de stockage. Ainsi, elle retournera toujours le même ensemble de vues pour toutes BDS malgré le fait qu'elles utilisent des modèles logiques différents.

Approche	Type	Contraintes	Raisonnement	BDS
Goasdoué *et al.* [83]	Transitions	Espace, Maintenance	Réécriture	Vertical
MARVEL [85]	Requêtes analytiques	Nombre de vues	Réécriture	Vertical
RDFMatView [84]	Index	Nombre d'index	Non considéré	Vertical
Dritsou *et al.* [110]	Expressions de chemins	Espace	Non considéré	Multiple
CBA [112]	Matrices d'usage	Espace	Saturation	Multiple
TBA [112]	Plan unifié	Espace	Saturation	Multiple

TABLE 3.3 – Positionnement de nos approches de sélection des vues matérialisées (en italique)

3.6.2 Production scientifique

Le tableau 3.4 synthétise notre production scientifique liée aux travaux de recherche présentés dans ce chapitre. Ils ont débuté en 2011 et ont été réalisés dans le cadre des thèses soutenues d'Ilyès Boukhari [98], Bery Mbaiossoum [113] et Selma Bouarar [78] (co-encadrements avec Ladjel Bellatreche).

Production	Description
Publications	2 RI, 1 RN, 7 CI, 1 CN, 2 API
Encadrements doctoraux	Ilyès Boukhari (2014), Bery Mbaiossoum (2014), Selma Bouarar (2016)
Projets et contrats	CFCA (prestation), Bimedia (prestation)
Logiciels	Extension d'OntoDB/OntoQL, OntoDBench

TABLE 3.4 – Production scientifique liée à nos travaux sur l'ingénierie des BDS

Ces travaux ont donné lieu à 2 RI, 1 RN, 7 CI, 1 CN et 2 API (atelier et démonstration). Parmi ces publications, les principales sont les suivantes.

- Un article dans la revue internationale *International Journal on Software Tools for Technology Transfer (STTT)* [100] qui présente notre approche d'intégration de besoins hétérogènes.
- Un article dans la revue internationale *Computers in Industry Journal* [120] qui utilise notre approche d'intégration des besoins hétérogènes dans le contexte des entrepôts de données.

– Deux articles dans la conférence internationale *East-European Conference on Advances in Databases and Information Systems (ADBIS 2014)* [112, 121]. Le premier présente une mise en œuvre de notre démarche de conception orientée variabilité pour la problématique de la conception logique des entrepôts de données [121]. Le second présente notre approche de sélection des vues matérialisées qui prend en compte la diversité des BDS [112].

Ces travaux ont impliqué le développement des logiciels OntoDBench [16] (1K lignes de code, environ 200 téléchargements) et d'une extension de la plateforme OntoDB/OntoQL pour la gestion de besoins hétérogènes [17]. Ces développements ont été valorisés dans le cadre de prestations avec l'entreprises Bimedia et la Compagnie Française de Câblage (CFCA).

3.7 Conclusion et perspectives de recherche

Dans cet axe de recherche nous avons étudié l'impact des BDS sur le processus de conception classique des SGD et proposé deux types d'évolutions.

– *Évolution horizontale* : constatant que les ontologies sont souvent utilisées par différents partenaires comme modèles communs, nous avons proposé une approche pour intégrer leurs besoins hétérogènes. Celle-ci nécessite qu'une ontologie globale soit acceptée par les différents partenaires et que les besoins soient exprimés dans l'un des formalismes supportés. En contrepartie, elle permet l'intégration des besoins suivants les deux dimensions de leur hétérogénéité : vocabulaire et formalisme. Nous avons aussi montré que les besoins ainsi intégrés peuvent être utilisés pour sélectionner des structures d'optimisation sur la BDS avant la mise en production de celle-ci.

– *Évolution verticale* : les BDS présentant une forte diversité à différents niveaux (conceptuel, logique et physique), la conception ou sélection d'une BDS adaptée à un problème donné repose sur des choix entre différentes variantes possibles. Pour prendre en compte cette diversité, nous avons proposé une démarche générale basée sur les formalismes et outils des LDP. Cette démarche explicite les choix de conception, à chaque phase, et leurs interdépendances. Pour outiller cette démarche, les différentes variantes identifiées doivent être implémentées. Aussi, au niveau de la phase physique, nous avons considéré le problème de sélection des vues matérialisées. L'originalité des deux approches que nous avons proposées est que celles-ci sont basées sur un modèle de coût qui prend en compte la diversité des BDS.

Nous décrivons à présent deux perspectives de cet axe de recherche qui nous semble prometteuses.

Gestion de l'évolution dans l'ingénierie des BDS. Dans les travaux présentés dans ce chapitre, nous avons considéré que le processus de conception des BDS est séquentiel et que les besoins, ontologies et charge de requêtes, sur lesquels il se base, sont stables. Ce n'est pas forcément le cas dans la pratique et c'est encore moins probable si on souhaite les mettre en œuvre dans le contexte des données massives dont deux caractéristiques fondamentales sont la *variété* et la *vélocité*. Aussi, une perspective de nos travaux est de prendre en compte le fait que les besoins exprimés par les partenaires puissent évoluer, que les ontologies utilisées pourraient être mises à jour fréquemment, à la fois sur la partie schéma et instance, et que les requêtes importantes pourraient ne pas être connues à l'avance. Nous envisageons

16. https://forge.lias-lab.fr/projects/ontodbench
17. https://forge.lias-lab.fr/projects/ontodb

d'intégrer ces caractéristiques via de nouveaux features dans notre démarche générale de conception des BDS. La difficulté réside dans le fait de mesurer précisément l'impact de telles évolutions sur le reste du processus de conception.

Généralisation du processus de conception. Dans nos travaux nous n'avons considéré que la persistance d'ontologies. Dans un contexte plus général, la nécessité de pouvoir traiter des données de diverses natures, qui peuvent être ontologiques mais également semi-structurées, textuelles, multimédia, etc., a entrainé la définition de nouvelles solutions de stockage (par exemple, les BD NoSQL [101] ou NewSQL [102]). La disponibilité d'un nombre croissant de solutions de stockage complexifie la tâche des concepteurs qui doivent choisir la solution ou les solutions adaptées à un problème donné. Une perspective de nos travaux est d'étendre notre démarche basée sur les LDP pour aider les concepteurs à répondre à cette question. Cela pose deux principaux challenges. D'une part, il paraît difficile de maintenir et d'implémenter un modèle de features qui comprendrait les variantes de conception des différents SGD. Nous envisageons donc une définition *collaborative* qui permettrait à des contributeurs de pouvoir l'enrichir avec de nouvelles variantes ou implémentations. D'autre part, notre modèle de features risque de devenir complexe et peu utilisable. Il est donc nécessaire de définir une approche permettant de le décomposer en plusieurs modèles de features compréhensibles pour les concepteurs tout en gardant la possibilité de les combiner pour mesurer l'impact des choix faits. Nous envisageons pour cela d'adapter des techniques de variabilité multi-dimensionnelle [122].

Chapitre 4

Techniques coopératives pour les Bases de Données Sémantiques

4.1 Introduction

Dans le chapitre précédent, nous nous sommes intéressés à la conception des *Bases de Données Sémantiques (BDS)*. Nous nous penchons à présent sur leur exploitation. De manière générale, les *Bases de Données (BD)* sont difficiles à exploiter pour des utilisateurs usuels [123]. Ceux-ci expriment des requêtes sur ces systèmes, souvent via des interfaces graphiques, en espérant obtenir un résultat de qualité (c'est-à-dire qui soit utile pour trouver l'information recherchée). Les systèmes, en retour, se contentent de répondre littéralement aux requêtes exprimées. Cela peut conduire à différents problèmes tels que le celui des *réponses vides (empty-answer problem)* : l'utilisateur obtient « Zéro résultat » comme réponse à sa requête alors qu'il espérait trouver des résultats.

Le problème des réponses vides a été abordé dans différents contextes tels que les *Bases de Données Relationnelles (BDR)* [124] ou les systèmes de recommandation [125]. Il est important dans le contexte des BDS pour les raisons suivantes.

- Les ontologies sont *incomplètes*. Par exemple, dans l'ontologie *DBPedia* [22], le nombre de personnels de 82% des instituts de formation n'est pas connu [18]. Ainsi, lorsqu'une requête posée par un utilisateur ne retourne pas de résultat, ce n'est pas nécessairement dû au fait que celle-ci est mal formulée ou trop restrictive. Le résultat attendu peut aussi ne pas être dans l'ontologie interrogée.
- La structure des données sémantiques est rarement connue des utilisateurs. Contrairement à ce que pourrait penser un utilisateur, cette structure n'est pas définie par le schéma de l'ontologie [15]. En effet, ce dernier ne fait que *décrire* des propriétés possibles pour caractériser des instances de classes. Il ne *prescrit* pas celles qui seront réellement utilisées pour une application donnée. En conséquence, une requête peut retourner un résultat vide simplement parce qu'une propriété impliquée dans la requête n'est pas utilisée pour décrire les instances recherchées.
- Le contenu d'une ontologie peut être méconnu de l'utilisateur. Cela vient du fait qu'une ontologie est souvent construite par l'intégration de données provenant de différentes sources [21, 126]. De

18. Cette statistique date d'octobre 2020.

plus, les ontologies peuvent être larges (par exemple, au moment de l'écriture de ce manuscrit, *Knowledge Vault* [126] et *DBPedia* [22] contenaient respectivement 1.6 et 3 milliard de faits), ce qui rend difficile d'appréhender leur contenu. Aussi, l'utilisateur peut penser que l'ontologie contient certaines informations qui n'y sont pas, ou pas sous cette forme, et ainsi exprimer des requêtes qui *échouent* (c'est-à-dire, qui retournent un résultat vide).

Les raisons évoquées précédemment font que le problème des réponses vides est fréquent dans le contexte des ontologies [16]. Pour répondre à ce type de problèmes, l'approche habituellement suivie consiste à fournir des *réponses coopératives* comme résultats d'une requête. Ce sont des réponses indirectes plus utiles que celles retournées par la requête [123]. Dans le contexte des BDS, cette approche a été mise en œuvre lors de travaux proposant des techniques de *relaxation* de requêtes sémantiques [17–20]. Celles-ci permettent de générer des *requêtes relaxées*, qui sont plus générales que la requête utilisateur. Les éventuels résultats de ces requêtes sont des *réponses approximatives* qui sont fournies à l'utilisateur. Le processus suivi dans ces travaux consiste à générer et exécuter des requêtes relaxées jusqu'à trouver le nombre de réponses approximatives voulues par l'utilisateur. Cependant, ce processus est réalisé *à l'aveugle*, c'est-à-dire sans identifier d'abord pourquoi la requête utilisateur a échoué. En conséquence, il peut exécuter de nombreuses requêtes relaxées ne résolvant pas les problèmes de la requête utilisateur et ainsi être inefficace .

Nous présentons, dans ce chapitre, des approches que nous avons développées pour pallier ce problème. Nous commençons par définir, dans la section 4.2, les limitations de l'état de l'art qui nous ont conduits à proposer ces approches. Puis, nous présentons, dans la section 4.3, deux approches permettant d'identifier les causes d'échec d'une requête sémantique. Elles sont inspirées d'approches définies dans le contexte des BDR [124] et des systèmes de recommandation [125]. Nous proposons ensuite, dans la section 4.4, des extensions de ces approches pour le contexte des *ontologies incertaines*, c'est-à-dire celles qui associent un degré de confiance à leurs faits. Dans la section 4.5, nous montrons l'intérêt des causes d'échec que nos approches calculent. Elles permettent, en effet, de définir des approches de relaxation de requêtes sémantiques plus efficaces que celles de la littérature. Nous tirons ensuite un bilan de cet axe de recherche dans la section 4.6 et concluons dans la section 4.7.

4.2 Objectifs de recherche

Nous présentons, dans la figure 4.1, une vue synthétique des approches visant à résoudre le problème des réponses vides. Nous distinguons d'abord les approches *basées sur les données*, qui visent à traiter ce problème en améliorant l'ontologie, soit en la *complétant* [127], soit en identifiant le *schéma relationnel* des données sémantiques [15], des approches *basées sur les requêtes*, qui visent à modifier les requêtes utilisateur. Les approches que nous proposons dans ce chapitre font partie de cette dernière catégorie, qui est composée des sous-catégories suivantes.

– *Aide à la construction des requêtes*. Ces approches visent à aider l'utilisateur, via des interfaces graphiques, à exprimer sa requête. L'approche la plus connue, nommée *Query By Example (QBE)*, a été initialement définie dans le contexte relationnel et récemment adaptée aux requêtes sémantiques [128]. Toujours dans l'idée d'aider l'utilisateur à construire sa requête, des approches suggèrent des éléments pour *compléter* la requête utilisateur en se basant sur le schéma de l'onto-

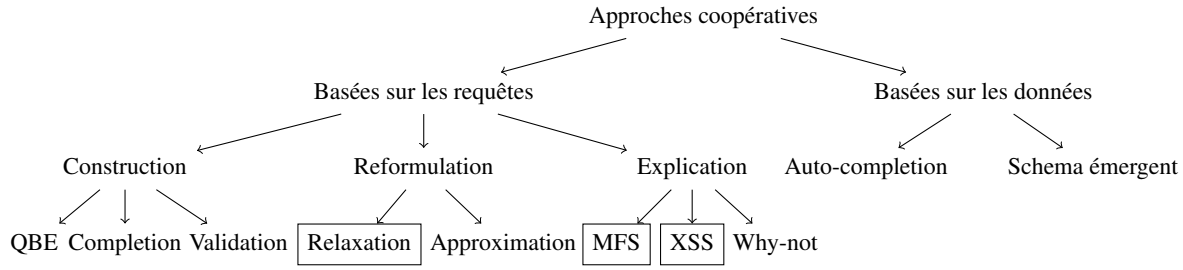


FIGURE 4.1 – Vue d'ensemble des approches proposées pour traiter le problème des réponses vides (nos travaux suivent les approches encadrées)

logie [129], les préférences utilisateur [130] ou l'historique des requêtes [131]. Les approches évoquées précédemment visent à aider l'utilisateur à définir sa requête pour éviter le problème des réponses vides. Cependant, elles ne garantissent pas que la requête finale formulée ne va pas échouer. Aussi des approches proposent de vérifier en continu si c'est le cas, en *validant* chaque ajout fait à la requête en cours de construction [132].

– *Reformulation des requêtes qui ont échoué*. Ces approches consistent à reformuler automatiquement ou interactivement la requête utilisateur. Cette reformulation peut consister en une *relaxation* de la requête utilisateur [17–20], c'est-à-dire à la définition d'une *requête relaxée* moins contraignante que la requête utilisateur. Quelle que soit l'ontologie interrogée, les réponses d'une requête relaxée incluent toutes celles de la requête utilisateur et, en général, en rajoutent. La reformulation de la requête utilisateur peut aussi consister en une *approximation* de celle-ci [18, 133–135]. Contrairement à une relaxation, il n'y a pas de relation d'inclusion entre les réponses de la requête utilisateur et celles de son approximation.

– *Explications liées à l'échec des requêtes*. Ces approches visent à fournir des explications à l'utilisateur sur l'échec de sa requête. Parmi celles-ci, certaines proposent de fournir la liste des *causes d'échec* de la requête utilisateur. Celles-ci sont exprimées sous la forme de *Minimal Failing Subquery (MFS)* [124]. Une MFS est une sous-requête minimale de la requête utilisateur qui échoue. Tant qu'une MFS sera présente dans la requête utilisateur, celle-ci échouera. Une MFS représente ainsi une cause d'échec. Les requêtes duales des MFS, nommées *MaXimal Succeeding Subquery (XSS)* sont également informatives pour l'utilisateur. Ce sont les sous-requêtes maximales de la requête utilisateur qui *réussissent*, c'est-à-dire qui retournent des résultats. Elles peuvent donc être exécutées par l'utilisateur pour trouver des réponses approximatives. Enfin, si l'utilisateur s'attendait à obtenir des résultats spécifiques dans la réponse à sa requête, des approches expliquent pourquoi ceux-ci n'apparaissent pas (*why-not questions*) [136, 137].

Comme l'étude précédente le montre, de nombreuses approches, qui pourraient être utilisées de manière complémentaire, ont été proposées pour le problème des réponses vides. Dans nos travaux, nous nous sommes d'abord intéressés aux approches permettant de fournir une explication à l'utilisateur lorsque sa requête échoue, c'est-à-dire à l'identification des MFS et XSS. Nous avons constaté que (i) les principales approches proposées pour fournir ces explications [124, 125] n'avaient jamais été adaptées au contexte des données et requêtes sémantiques et que (ii) les approches consistant à relaxer une requête sémantique le faisait, *à l'aveugle*, c'est-à-dire sans connaître les raisons pour lesquelles la requête utilisateur a échoué. Aussi, nous avons mené des travaux afin de proposer des techniques coopératives pour

les BDS. Leurs objectifs sont les suivants :

- proposer des approches calculant simultanément les MFS et XSS d'une requête sémantique en s'inspirant des deux principales approches de l'état de l'art [124, 125] ;
- étendre ces approches au contexte des ontologies contenant des informations incertaines ;
- s'appuyer sur l'identification des causes d'échec pour proposer des approches de relaxation de requêtes sémantiques plus efficaces que celles proposées dans la littérature.

Nous avons répondu à ces objectifs par les contributions suivantes :

- contribution 1 : la proposition de deux approches calculant les MFS et XSS d'une requête sémantique qui échoue (voir section 4.3) ;
- contribution 2 : la prise en compte de l'incertitude des ontologies dans nos approches d'explication de l'échec d'une requête sémantique (voir section 4.4) ;
- contribution 3 : la proposition d'approches automatiques de relaxation de requêtes sémantiques basées sur les causes d'échec (voir section 4.5).

4.3 Contribution 1 : calcul des causes d'échec d'une requête sémantique

Nous commençons par détailler les raisons qui ont motivé nos travaux. Ceux-ci sont développés dans le contexte du Web sémantique (c'est-à-dire, avec les langages RDF et SPARQL) pour pouvoir comparer nos approches à celles existantes.

4.3.1 Contexte et motivation : identification des MFS et XSS

Comme nous l'avons indiqué précédemment, le fait que les ontologies puissent être volumineuses mais incomplètes et que leurs structures et contenus sont souvent méconnus des utilisateurs font que le problème des réponses vides n'est pas anodin dans le contexte des ontologies. Par exemple, une étude de différents *points d'accès SPARQL* (*SPARQL endpoint*) pendant plusieurs mois a montré que 10% des requêtes soumises sur *DBpedia* pendant cette période avaient retourné un résultat vide [16]. Face à ce problème, un utilisateur va généralement essayer de nouvelles requêtes en modifiant ou rendant optionnelles les conditions de sa requête initiale. La difficulté est que celui-ci ne peut pas savoir si sa requête est mal formulée, trop sélective ou si le résultat recherché n'est pas dans l'ontologie interrogée. En conséquence, le processus d'*essais et erreurs*, durant lequel l'utilisateur va modifier sa requête afin de trouver une reformulation adéquate, risque d'être long et fastidieux.

Dans le contexte des ontologies, les travaux menés pour répondre au problème des réponses vides se sont focalisés sur la proposition de techniques de *relaxation de requêtes* [17, 19, 20, 138]. Le principe de ces approches est d'utiliser des opérateurs de relaxation pour créer manuellement ou automatiquement des *requêtes relaxées*, moins contraignantes que celle de l'utilisateur. Cependant, dans toutes ces approches, le processus de relaxation se fait *à l'aveugle*, c'est-à-dire sans savoir pourquoi la requête a échoué. Ainsi, les raisons qui font que la requête exprimée par l'utilisateur a échoué, ne sont pas explicitées. Cela aurait pourtant un double intérêt. D'une part, cela accélérerait le processus de relaxation en évitant d'exécuter des requêtes relaxées qui contiennent encore une cause d'échec de la requête utilisa-

teur (cet aspect est détaillé dans la section 4.5). D'autre part, connaître les causes d'échec d'une requête permet à l'utilisateur de mieux comprendre la distribution des données dans l'ontologie interrogée et ainsi de formuler des requêtes plus pertinentes pour répondre à ses besoins [124].

Dans le contexte des BDR, l'identification des causes d'échec d'une requête SQL qui échoue a été largement étudiée [124,125,139,140]. Ces travaux ont conduit à la définition des notions de MFS et XSS, à la preuve que l'énumération de l'ensemble des MFS est un problème $\mathcal{NP}$-Difficile et à la proposition d'algorithmes pour les calculer. Parmi ces algorithmes, deux principales approches se sont distinguées.

- L'approche proposée par Godfrey [124] qui consiste en un parcours du treillis formé par les sous-requêtes de la requête utilisateur. Cette approche calcule soit les MFS, soit les XSS.
- L'approche proposée par Jannach [125] qui consiste à calculer une matrice dont il est possible d'extraire les XSS de la requête utilisateur.

Nous remarquons, d'une part, que ces approches ne calculent pas *simultanément* les MFS et XSS d'une requête qui échoue. Il serait pourtant utile de fournir à l'utilisateur non seulement les raisons pour lesquelles sa requête a échoué, mais, aussi, des requêtes alternatives qu'il pourrait exécuter avec la garantie d'obtenir des résultats. D'autre part, ces deux principales approches n'ont pas été adaptées au contexte des données et requêtes sémantiques qui apportent pourtant de nouvelles dimensions.

- Le langage RDF présentent plusieurs particularités comme la possibilité d'utiliser des *ressources anonymes* (*nœuds blancs*) à la place d'un sujet ou objet d'un triplet RDF, de modéliser des *triplets implicites* qui seront dérivés grâce à des règles de raisonnement ou d'utiliser un schéma pour décrire les classes et propriétés des données représentées.
- Le langage SPARQL présente plusieurs particularités par rapport à SQL. Par exemple, SPARQL permet l'utilisation de variables à la place de prédicats dans les requêtes. La sémantique de ces deux langages diffèrent également. Ainsi, en SQL, une jointure rejette les valeurs *nulles* alors que c'est le contraire en SPARQL [141].
- Les requêtes SPARQL sont décomposées en *patrons de triplet*, ce qui fait que la taille d'une requête SPARQL peut être importante. Par exemple, des requêtes ayant jusqu'à 15 patrons de triplet ont été trouvées dans les journaux de requêtes de *DBpedia* [111].
- La décomposition des données en triplets RDF fait que la taille d'une ontologie peut être de millions voir de milliards de triplets.

Ainsi, les approches conçues dans le contexte relationnel, pour des tables ayant des milliers de lignes et des requêtes SQL ayant peu de conditions dans la clause `WHERE`, sont difficilement applicables en pratique dans le contexte RDF. Nous montrons dans ce qui suit comment nous nous sommes inspirés de ces travaux pour proposer des approches d'identification des MFS et XSS dans le contexte des BDS.

4.3.2 Principe général de nos approches d'identification des MFS et XSS

Nous commençons par définir formellement les notions de MFS et XSS et illustrer leur intérêt. Nos approches sont définies pour les langages RDF et un sous-ensemble de SPARQL. Nos définitions et notations se basent sur celles données dans [141].

4.3.2.1 Notions de MFS et XSS

Nous considérons les *requêtes RDF* définies comme une conjonction de patrons de triplet : $Q = t_1 \wedge \cdots \wedge t_n$. Soit D une BDS stockant des triplets RDF et Q une requête RDF, l'évaluation de Q sur D est notée $[[Q]]_D$. Cette évaluation peut être faite suivant les différents modes de raisonnement (*entailment regime*) définis dans la spécification SPARQL. Dans les exemples de ce chapitre, nous n'utilisons pas de raisonnement (*simple entailment regime*) même si nos algorithmes peuvent être utilisés avec tous les modes de raisonnement.

Étant donnée une requête $Q = t_1 \wedge \cdots \wedge t_n$, une requête $Q' = t_i \wedge \cdots \wedge t_j$ est une *sous-requête* de Q, $Q' \subseteq Q$, si et seulement si $\{t_i, \cdots, t_j\} \subseteq \{t_1, \cdots, t_n\}$. Si $\{t_i, \cdots, t_j\} \subset \{t_1, \cdots, t_n\}$, Q' est une *sous-requête directe* de Q ($Q' \subset Q$). Si une sous-requête Q' de Q échoue, alors la requête Q échoue.

Définition 1. *Une MFS Q^* d'une requête Q est définie comme suit : $[[Q^*]]_D = \emptyset \ \wedge \nexists\, Q' \subset Q^*$ telle que $[[Q']]_D = \emptyset$. L'ensemble de toutes les MFS d'une requête Q est noté $mfs(Q)$.*

Définition 2. *Une XSS Q^* d'une requête Q est définie comme suit : $[[Q^*]]_D \neq \emptyset \ \wedge \nexists\, Q'$ telle que $Q^* \subset Q' \wedge [[Q']]_D \neq \emptyset$. L'ensemble de toutes les XSS d'une requête Q est noté $xss(Q)$.*

Exemple. Pour illustrer les notions de MFS et XSS, nous considérons la requête $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$ indiquée dans la figure 4.2 (b). Cette requête recherche les enseignants (`Lecturer`) avec leur âge, qui enseignent les BD (`DB`) et sont intéressés par des recherches en Web sémantique (`SW`). Lorsque cette requête est exécutée sur la BDS contenant les triplets RDF représentés dans la figure 4.2 (a), elle retourne un résultat vide.

Triplets		
sujet	**prédicat**	**objet**
p_1	type	Professor
p_1	researchInterest	SW
p_1	teacherOf	DB
p_2	type	Lecturer
p_2	teacherOf	DB
p_3	teacherOf	DB
p_4	researchInterest	SW
p_5	type	Lecturer
p_5	teacherOf	AI

(a) Triplets RDF

```
SELECT ?p ?a WHERE {
?p age ?a                        (t1)
?p type Lecturer                 (t2)
?p researchInterest "SW"         (t3)
?p teacherOf "DB" }              (t4)
```

(b) La requête Q

$mfs(Q) = \{t_1, t_2 \wedge t_3\}$
$xss(Q) = \{t_2 \wedge t_4, t_3 \wedge t_4\}$

(c) Les MFS et XSS de Q

FIGURE 4.2 – Exemples de MFS et XSS d'une requête RDF

Comme indiqué dans la figure 4.2 (c), cette requête a deux MFS : t_1, qui signifie que les âges des personnes ne sont pas connus dans la BDS, et $t_2 \wedge t_3$ indiquant qu'aucun enseignant n'est intéressé par des recherches en Web sémantique dans la BDS. Elle a également deux XSS : $t_2 \wedge t_4$ qui indique qu'il existe au moins un enseignant de BD et $t_3 \wedge t_4$ signifiant qu'il existe au moins une personne intéressée par des recherches en Web sémantique qui enseigne les BD.

L'utilisateur peut essayer de corriger sa requête grâce aux MFS. Dans notre exemple, il devra forcément enlever le patron de triplet t_1 puisqu'il n'y a pas d'âge dans la BDS. De plus, il sait qu'il y a un

problème dans le fait de chercher un `Lecturer` intéressé par des recherches en Web sémantique (MFS $t_2 \wedge t_3$). Avec ces informations, s'il choisit d'enlever t_1 et de remplacer `Lecturer` par `Professor` dans sa requête, le problème de réponses vides est résolu.

L'utilisateur peut aussi choisir d'exécuter une des XSS proposées, puisqu'elles donnent la garantie d'obtenir un résultat. Il sait ainsi qu'il peut trouver des enseignants de BD (XSS $t_2 \wedge t_4$) ou des personnes intéressées par des recherches en Web sémantique qui enseignent les BD (XSS $t_3 \wedge t_4$). Cependant, ces XSS peuvent ne plus contenir des patrons de triplet importants pour l'utilisateur et il peut ainsi préférer la première option (c'est-à-dire, essayer de corriger sa requête à l'aide des MFS).

Vu l'intérêt des MFS et XSS que nous venons d'illustrer, notre objectif était de définir des approches pour les calculer de la manière la plus efficace possible. Nous décrivons le résultat de ces travaux succinctement dans ce qui suit. Le lecteur intéressé pourra consulter [142] pour plus de détails.

4.3.2.2 Approche LBA (Lattice-Based Approch)

L'approche LBA, inspirée des travaux de Godfrey [124], vise à explorer un minimum d'éléments du treillis formé par les sous-requêtes de la requête utilisateur Q. La figure 4.3 illustre ce treillis des sous-requêtes pour notre exemple (voir figure 4.2). Cette figure indique également les sous-requêtes qui réussissent et celles qui échouent (par convention, la requête vide réussit).

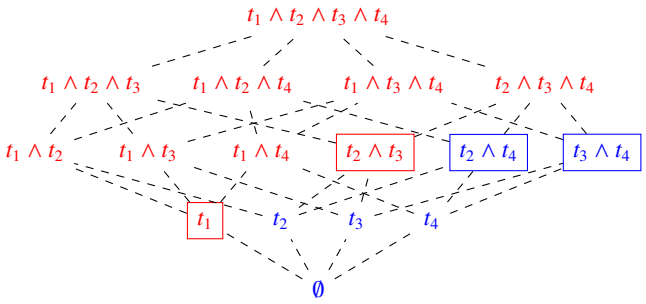


FIGURE 4.3 – Treillis des sous-requêtes de notre exemple de requête avec ses MFS et XSS

L'approche LBA parcourt ce treillis en suivant les trois étapes suivantes.

1. *Trouver une MFS*. Cette étape est réalisée avec l'algorithme *a_mel_fast* proposé par Godfrey [124]. Si n est la taille de la requête Q, cet algorithme permet de trouver une MFS en n étapes. Sa complexité est donc en $O(n)$.
2. *Calculer les XSS potentielles*. Par définition, les requêtes qui incluent la MFS Q^*, trouvée à l'étape précédente, échouent. Ainsi, elles ne peuvent être ni MFS, ni XSS de Q et l'espace de recherche peut ainsi être élagué. L'exploration du treillis de sous-requêtes continue avec les plus grandes sous-requêtes de Q qui n'incluent pas Q^*. Si ces sous-requêtes réussissent, ce sont des XSS de Q. Ainsi, nous les appelons *XSS potentielles* et notons cet ensemble de requêtes $pxss(Q, Q^*)$.
3. *Trouver toutes les MFS et XSS*. Cette étape consiste à exécuter chaque XSS potentielle Q', calculée dans l'étape précédente. Si Q' réussit, c'est une XSS de Q. Sinon, Q' a au moins une MFS, qui

est aussi une MFS de Q. Cette MFS peut être identifiée en suivant les étapes précédentes de l'approche LBA. Ainsi cette approche est appliquée récursivement pour chaque XSS potentielle qui échoue. Ce processus s'arrête quand toutes les XSS potentielles ont été parcourues.

Exemple. La figure 4.4 présente une exécution de l'algorithme LBA pour calculer les MFS et XSS de notre exemple : $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$. La MFS $Q^* = t_1$ est trouvée avec l'algorithme *a_mel_fast*, ce qui conduit au calcul d'une seule XSS potentielle $t_2 \wedge t_3 \wedge t_4$. Le processus se poursuit en exécutant cette requête. Comme un résultat vide est obtenu, l'algorithme *a_mel_fast* est à nouveau appliqué sur cette requête pour trouver la seconde MFS $Q^{**} = t_2 \wedge t_3$. Les XSS potentielles correspondantes sont $t_3 \wedge t_4$ et $t_2 \wedge t_4$. Ces deux requêtes sont ensuite exécutées. Puisqu'elles réussissent, elles sont identifiées comme des XSS. La liste des XSS potentielles $pxss$ étant vide, l'algorithme s'arrête et retourne ces deux XSS et les MFS trouvées précédemment.

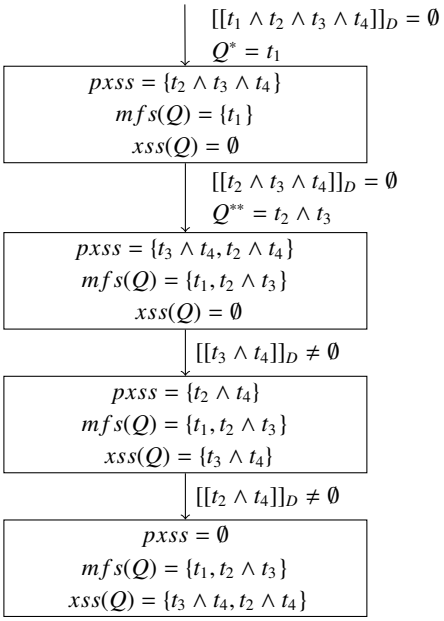


FIGURE 4.4 – Un exemple d'exécution de l'algorithme LBA pour trouver les MFS et XSS de Q

4.3.2.3 Approche MBA (Matrix-Based Approach)

Dans l'approche LBA, la taille de l'espace de recherche augmente de manière exponentielle en fonction du nombre de patrons de triplet de la requête utilisateur. Pour éviter ce problème, nous nous sommes inspirés de l'approche de Jannach [125] qui avait été proposée dans le contexte des systèmes de recommandation. L'idée est de calculer une matrice, que nous appelons la *matrice relaxée*, en utilisant n requêtes, où n est la taille de la requête utilisateur. Les MFS et XSS peuvent alors être obtenues à partir de la matrice relaxée sans exécuter de requêtes supplémentaires.

Définition de la matrice relaxée d'une requête RDF

Soit Q une requête RDF exécutée sur un ensemble de triplets RDF D, les *solutions potentielles* de Q sont celles qui satisfont au moins un patron de triplet de Q sur D. Cet ensemble de solutions potentielles

de Q est noté $ps(Q, D)$. La matrice relaxée d'une requête RDF est définie comme suit.

Définition 3. *La matrice relaxée M d'une requête $Q = t_1 \wedge ... \wedge t_n$ sur une BDS contenant des triplets RDF D est une table à deux dimensions avec comme lignes les solutions potentielles $\mu \in ps(Q, D)$ et comme colonnes les patrons de triplet $t_i \in Q$. Pour une solution potentielle $\mu \in ps(Q, D)$ et un patron de triplet $t_i \in Q$, $M[\mu][t_i] = 1$ si μ satisfait t_i, sinon $M[\mu][t_i] = 0$.*

Exemple. La figure 4.5 présente la matrice relaxée de notre exemple (voir figure 4.2). Chaque ligne de la matrice relaxée est une solution qui satisfait au moins un patron de triplet. Par exemple, la première ligne correspond à la solution $?p \rightarrow p_1$. Elle satisfait les patrons de triplet t_3 et t_4 et donc la valeur 1 est mise dans les colonnes correspondantes et 0 pour les autres.

?p	t_1	t_2	t_3	t_4	
p_1	0	0	1	1	*
p_2	0	1	0	1	*
p_3	0	0	0	1	
p_4	0	0	1	0	
p_5	0	1	0	0	

FIGURE 4.5 – Matrice relaxée de notre exemple de requête Q

Calcul de la matrice relaxée d'une requête RDF

Nous avons montré que, dans le cas général, le calcul de la matrice relaxée nécessite l'introduction d'un opérateur de *jointure étendue* à cause de la différence de sémantique entre les langages SQL et SPARQL (en SQL, une jointure rejette les valeurs nulles alors que c'est le contraire en SPARQL). Cependant, nos expérimentations ont montré que la taille de la matrice relaxée peut être grande et ainsi nécessiter un temps de calcul important. Cela vient, en particulier, du fait qu'une sous-requête de la requête utilisateur peut contenir un produit cartésien. Aussi, dans le cadre de l'approche MBA, nous nous sommes restreints aux *requêtes en étoile* (c'est-à-dire, aux requêtes constituées de patrons de triplet qui ont la même variable comme sujet). Ce sont des requêtes fréquentes en pratique [111].

Le principe de l'algorithme de calcul de la matrice relaxée pour une requête en étoile est le suivant. Il exécute une requête pour chaque patron de triplet t_i. Pour chaque résultat de ces requêtes, la matrice est mise à jour selon la valeur de la variable de jointure (c'est-à-dire, celle en sujet des patrons de triplet).

- Si cette valeur n'est pas dans la matrice, elle est ajoutée comme une nouvelle ligne de la matrice relaxée. Cette ligne a la valeur 1 pour la colonne qui correspond à t_i et 0 pour les autres.
- Si cette valeur était déjà dans la matrice relaxée, la ligne correspondante dans la matrice est mise à jour avec la valeur 1 pour la colonne t_i.

Exemple. La figure 4.6 présente l'exécution de l'algorithme calculant la matrice relaxée pour notre exemple (voir figure 4.2). L'exécution du patron de triplet t_1 ne retourne pas de résultat et donc la matrice relaxée reste vide. L'exécution du patron de triplet t_2 retourne deux résultats qui sont ajoutés comme nouvelles lignes à la matrice relaxée avec une valeur 1 pour la colonne t_2 et 0 pour les autres. Le même processus est appliqué pour le patron de triplet t_3, ce qui ajoute deux nouvelles lignes dans la matrice relaxée. Enfin le patron de triplet t_4 est exécuté. Il retourne les résultats $?p \rightarrow p_1$, $?p \rightarrow p_2$

et $?p \rightarrow p_3$. Comme les deux premiers étaient déjà dans la matrice relaxée, les lignes correspondantes sont juste mises à jour avec la valeur 1 pour la colonne t_4. Le résultat $?p \rightarrow p_3$ est lui inséré comme nouvelle ligne de la matrice relaxée avec la valeur 1 uniquement pour t_4.

$M = \emptyset$
$[[t_1]]_D \hookrightarrow M = \emptyset$
$[[t_2]]_D \rightarrow M =$

?p	t_1	t_2	t_3	t_4
p_2	0	1	0	0
p_5	0	1	0	0

$[[t_3]]_D \rightarrow M =$

?p	t_1	t_2	t_3	t_4
p_2	0	1	0	0
p_5	0	1	0	0
p_1	0	0	1	0
p_4	0	0	1	0

$[[t_4]]_D \rightarrow M =$

?p	t_1	t_2	t_3	t_4
p_2	0	1	0	1
p_5	0	1	0	0
p_1	0	0	1	1
p_4	0	0	1	0
p_3	0	0	0	1

FIGURE 4.6 – Un exemple illustrant le calcul de la matrice relaxée de notre requête Q

Utilisation de la matrice relaxée pour trouver les MFS et XSS d'une requête RDF

Une fois la matrice relaxée calculée, les XSS de cette requête peuvent être identifiées en calculant le *skyline* [19] de cette matrice, c'est-à-dire les lignes ayant un maximum de 1. Ce calcul peut être fait en utilisant un des nombreux algorithmes définis dans la littérature (un état de l'art est proposé dans [143]). Dans la figure 4.5, les lignes qui composent le skyline de la matrice relaxée sont marquées avec le caractère $*$. Les requêtes correspondant à ces lignes sont les XSS ($t_3 \wedge t_4$ et $t_2 \wedge t_4$).

Pour le calcul des MFS, la matrice relaxée permet de savoir si une requête échoue. Il suffit pour cela de calculer l'intersection des colonnes de la matrice relaxée qui correspondent aux patrons de triplet de la requête. Si la colonne qui en résulte est uniquement composée de 0, alors la requête échoue. Ainsi, la matrice relaxée peut être utilisée dans l'approche LBA pour éviter d'exécuter des requêtes. Cette approche nécessite toujours d'explorer un espace de recherche qui augmente de manière exponentielle avec le nombre de patrons de triplet de la requête ; mais, cette exploration ne nécessite plus l'exécution de nouvelles requêtes une fois que la matrice relaxée a été calculée.

4.3.3 Résultats

À notre connaissance, nos travaux sont les premiers à avoir proposé des approches pour calculer simultanément les MFS et XSS d'une requête RDF qui échoue. La première, nommée LBA, se base sur une exploration du treillis des sous-requêtes de la requête utilisateur qui exploite les propriétés des MFS et XSS pour réduire l'espace de recherche. Nous avons montré dans [144] que l'algorithme LBA est *correct*, c'est-à-dire qu'il retourne exactement toutes les MFS et XSS. Nous avons également prouvé que

19. Soit un ensemble d'objets décrits par une liste de critères. Un skyline est un sous-ensemble d'objets qu'aucun autre objet ne domine (au sens de Pareto) sur des critères d'intérêt.

sa complexité au pire des cas est $O(\sqrt{n} * 2^n)$, où n est la taille de la requête utilisateur. Malgré cette complexité, nos expérimentations montrent que cet algorithme peut retourner les MFS et XSS d'une requête ayant moins de 15 patrons de triplet en un temps raisonnable. Enfin, nous avons proposé différentes heuristiques pour améliorer les performances de cet algorithme. Elles visent en particulier à éviter d'exécuter des produits cartésiens et à mettre en cache des requêtes pour limiter le nombre de requêtes exécutées. Ce sont ces optimisations qui rendent l'algorithme LBA différent d'autres algorithmes de parcours de treillis et, en particulier, ceux définis pour la découverte des fréquents maximaux [145, 146].

Notre seconde approche, nommée MBA, est basée sur le calcul d'une matrice qui enregistre, pour chaque solution potentielle de la requête utilisateur, l'ensemble des patrons de triplet que cette solution satisfait. Les XSS correspondent au skyline de cette matrice. Pour les MFS, cette dernière permet d'utiliser l'approche LBA sans avoir besoin d'exécuter des requêtes. Nous avons montré dans [144] que, dans le cas général, le calcul de cette matrice nécessite d'utiliser un opérateur spécifique. Cela est dû à la sémantique spécifique de SPARQL sur le traitement des valeurs nulles. Cependant, nous avons montré expérimentalement que ce calcul est coûteux et difficilement acceptable en pratique. Aussi, nous avons spécialisé l'approche MBA pour les requêtes en étoile.

Nous avons implémenté les approches LBA et MBA en Java. Elles peuvent être exécutées sur différents triplestores tels que Jena TDB ou Virtuoso. Notre implémentation est disponible sur `http://www.lias-lab.fr/forge/projects/qars`. En complément, nous avons mené un ensemble d'expérimentations pour évaluer nos approches sur le banc d'essai *LUBM* [42]. Ces expérimentations ont été menées avec différents types de requêtes et plusieurs tailles de données. Elles ont consisté à évaluer la performance de nos algorithmes en termes de temps de réponse et de nombre de requêtes exécutées. Nous avons comparé nos approches avec une méthode de base, nommée *DFS*, qui consiste en un parcours en profondeur du treillis des sous-requêtes et un algorithme de l'état de l'art nommé *ISHMAEL* [124]. Nous avons testé deux versions de MBA selon que la matrice relaxée soit pré-calculée (*MBA-M*) ou non (*MBA+M*). Les résultats de ces expérimentations nous ont permis de tirer les conclusions suivantes.

- MBA-M a le meilleur temps de réponse qui est de quelques millisecondes même pour les requêtes avec 15 patrons de triplet sur *LUBM1K* (130M de triplets RDF). Cela vient du fait que cet algorithme doit uniquement calculer l'intersection de colonnes de la matrice relaxée au lieu d'exécuter des requêtes. Cependant cet algorithme n'est disponible que pour les requêtes en étoile et repose sur l'hypothèse forte que la matrice relaxée a été pré-calculée. Cela implique d'avoir identifié la requête utilisateur comme en étant une qui échoue fréquemment (par exemple, en utilisant les journaux des requêtes exécutées). Si la matrice relaxée est calculée pendant cet algorithme (MBA+M), ce temps de réponse est beaucoup plus important car le calcul de cette matrice est coûteux (environ 30 secondes pour une requête de 15 patrons de triplet).
- LBA, qui peut être utilisée pour tout type de requêtes, a de meilleures performances que DFS et ISHAMEL. Cette différence est importante pour des requêtes avec plus de 10 patrons de triplet pour lesquelles LBA retourne le résultat environ 5 fois plus rapidement qu'ISHMAEL et DFS.

Nos expérimentations ont montré l'intérêt des approches LBA et MBA. Vu les hypothèses fortes liées à l'utilisation de MBA, nous considérons que l'approche LBA est celle qui, dans le cas général, doit être utilisée pour calculer les MFS et XSS d'une requête RDF. Cependant, cette approche ne prend pas en compte une caractéristique de nombreuses ontologies réelles qui est d'associer un *degré de confiance* à chaque fait représenté. Nous considérons cette caractéristique dans ce qui suit.

4.4 Contribution 2 : prise en compte de l'incertitude des données RDF

Nous commençons par expliquer brièvement les raisons qui font que de nombreuses ontologies contiennent des faits incertains. Nous décrivons ensuite les conséquences sur nos travaux.

4.4.1 Contexte et motivation : incertitude des données RDF

De nombreuses ontologies réelles telles que *YAGO* [21], *Knowledge Vault* [126] ou *Nell* [147] sont construites via un processus d'extraction et de fusion d'informations provenant de multiples sources externes, et notamment celles du Web. Comme la fiabilité de ces sources n'est pas totale, les faits de ces ontologies peuvent être entachés d'*incertitude*, c'est-à-dire que leur véracité n'est pas garantie. Les approches qui ont permis de construire ces ontologies incluent des méthodes pour quantifier l'incertitude des faits représentés. Par exemple, dans *Knowledge Vault*, un score de confiance est associé à chaque fait. Il représente la probabilité que ce fait soit correct. Ce score est calculé à partir de la fiabilité des sources de données utilisées et par rapport au nombre de fois où le fait a été extrait des sources de données (plus le fait est extrait, plus son score de confiance augmente).

Sans perte de généralité, nous considérons que les *ontologies incertaines* associent à chacun de leurs faits une valeur réelle comprise entre 0 et 1. Plus celle-ci se rapproche de 1, plus le fait correspondant est considéré comme certain. Nous qualifions cette valeur de *degré de confiance*. Pour prendre en compte le caractère incertain des faits d'une ontologie, des extensions des langages RDF et SPARQL ont été proposées [148, 149]. Elles associent un degré de confiance à chaque triplet RDF ainsi qu'aux résultats des requêtes SPARQL.

Lors de l'interrogation des ontologies incertaines, les utilisateurs s'attendent à obtenir des résultats de qualité, c'est-à-dire des résultats possédant un degré de confiance supérieur à un seuil donné α. Si une requête retourne un résultat vide, cela peut être non seulement dû aux raisons que nous avons évoquées dans le contexte des ontologies certaines (par exemple, au fait que la requête est trop restrictive) mais aussi au seuil de confiance défini. Nous avons donc étudié comment adapter et étendre nos approches d'identification des MFS et XSS pour prendre en compte cette nouvelle dimension.

À notre connaissance, aucune approche n'avait été proposée pour l'identification des MFS et XSS dans le contexte des ontologies incertaines. Cependant, plusieurs travaux sur les MFS et XSS avaient été menés dans le contexte des requêtes *floues* exprimées sur une BDR [140, 150, 151]. Dans ce contexte, un *degré de satisfaction* peut être associé à chaque résultat d'une requête contenant des prédicats flous. Lorsque l'utilisateur exige un seuil minimum de degré de satisfaction, le problème des réponses vides peut se poser. Pour y répondre, Pivert et Smits [150] ont proposé de calculer les MFS et XSS pour différents degrés de satisfaction. Cette approche suit l'approche matricielle de Jannach [125] en construisant un résumé de la BD afin d'en extraire les MFS et XSS. Par rapport à nos objectifs, ces travaux proposent une approche pour calculer les MFS et XSS dans le contexte de l'interrogation floue sur des BD certaines, alors que nous visons des requêtes sémantiques sans prédicats flous sur des ontologies incertaines. Nous avons vu que, dans ce contexte, une approche basée sur le calcul d'une matrice peut être difficilement utilisée en pratique [144]. Aussi, nous avons commencé par étudier dans quelle mesure l'approche LBA pouvait être adaptée au contexte des ontologies incertaines.

4.4.2 Principe général de nos approches basées sur l'incertitude des données RDF

Nous commençons par transposer les notions de MFS et XSS au contexte des ontologies incertaines. Nous les qualifions d'*αMFS* et *αXSS*. Nous formalisons ces notions dans ce qui suit en nous appuyant sur les définitions données dans la section 4.3.2.1 et sur le modèle de confiance proposé par Hartig [148].

4.4.2.1 Notions d'αMFS et αXSS

Une ontologie incertaine contient un ensemble de triplets RDF (noté T_{RDF}) où chaque triplet est associé à un *degré de confiance* représentant la fiabilité du fait représenté. Ce degré est associé au triplet par une fonction $tv : T_{RDF} \rightarrow [0, 1]$. Soit μ une solution d'une requête RDF $Q = t_1 \wedge \cdots \wedge t_n$ et *aggreg* une fonction d'agrégation (par exemple, l'opérateur minimum), le degré de confiance de μ est défini par $tv(\mu, Q) = aggreg(tv(\mu(t_1)), \cdots, tv(\mu(t_n)))$ où $\mu(t_i)$ est le triplet obtenu en remplaçant les variables du patron de triplet t_i selon μ. Une requête RDF peut être associée à un seuil de confiance α afin de ne retourner que les résultats dont le degré de confiance est supérieur ou égal à α. L'évaluation d'une telle requête est définie par : $[[Q]]^{\alpha}_{D} = \{\mu \in [[Q]]_D \mid tv(\mu) \geq \alpha\}$. Les définitions des αMFS et αXSS sont directement dérivées de celles des MFS et XSS.

Définition 4. *Pour un α donné, une αMFS Q^* d'une requête Q est définie par : $[[Q^*]]^{\alpha}_{D} = \emptyset \wedge \nexists\, Q' \subset Q^*$ tel que $[[Q']]^{\alpha}_{D} = \emptyset$. L'ensemble de toutes les αMFS d'une requête Q est noté $mfs^{\alpha}(Q)$.*

Définition 5. *Pour un α donné, une αXSS Q^* d'une requête Q est définie par : $[[Q^*]]^{\alpha}_{D} \neq \emptyset \wedge \nexists\, Q'$ tel que $Q^* \subset Q' \wedge [[Q']]^{\alpha}_{D} \neq \emptyset$. L'ensemble de toutes les αXSS d'une requête Q est noté $xss^{\alpha}(Q)$.*

Exemple. La figure 4.7 étend l'exemple utilisé jusqu'à présent.

Triplets			
sujet	**prédicat**	**objet**	**tv**
p_1	type	Professor	0.7
p_1	researchInterest	SW	1
p_1	teacherOf	DB	0.9
p_2	type	Lecturer	0.7
p_2	teacherOf	DB	0.7
p_3	teacherOf	DB	0.7
p_4	researchInterest	SW	0.7
p_5	type	Lecturer	0.7
p_5	teacherOf	AI	0.7
p_6	type	Lecturer	0.3
p_6	age	35	0.5
p_6	researchInterest	SW	0.3
p_7	type	Lecturer	0.3
p_7	teacherOf	DB	0.3
p_7	researchInterest	SW	0.3

(a) Triplets RDF incertains

```
SELECT ?p ?a WHERE {
?p age ?a                        (t1)
?p type Lecturer                 (t2)
?p researchInterest "SW"         (t3)
?p teacherOf "DB" }              (t4)
```

(b) La requête Q

$mfs^{0.2}(Q) = \{t_1 \wedge t_4\}$
$xss^{0.2}(Q) = \{t_1 \wedge t_2 \wedge t_3, t_2 \wedge t_3 \wedge t_4\}$
$mfs^{0.4}(Q) = \{t_1 \wedge t_2, t_1 \wedge t_3, t_1 \wedge t_4, t_2 \wedge t_3\}$
$xss^{0.4}(Q) = \{t_1, t_2 \wedge t_4, t_3 \wedge t_4\}$
$mfs^{0.6}(Q) = \{t_1, t_2 \wedge t_3\}$
$xss^{0.6}(Q) = \{t_2 \wedge t_4, t_3 \wedge t_4\}$
$mfs^{0.8}(Q) = \{t_1, t_2\}$
$xss^{0.8}(Q) = \{t_3 \wedge t_4\}$

(c) Des αMFS et αXSS de Q

FIGURE 4.7 – Exemples d'αMFS et αXSS d'une requête RDF

Cette figure présente un ensemble de triplets RDF incertains et les αMFS et αXSS de notre exemple de requête Q pour les seuils : $\{0.2, 0.4, 0.6, 0.8\}$. Cet exemple est basé sur l'utilisation du *minimum*

comme fonction d'agrégation pour le calcul du degré de confiance d'un résultat (fonction *aggreg* dans les définitions données précédemment). Les αMFS et αXSS de Q sont également illustrées dans la figure 4.8 sur le treillis des sous-requêtes de Q pour les différents seuils considérés.

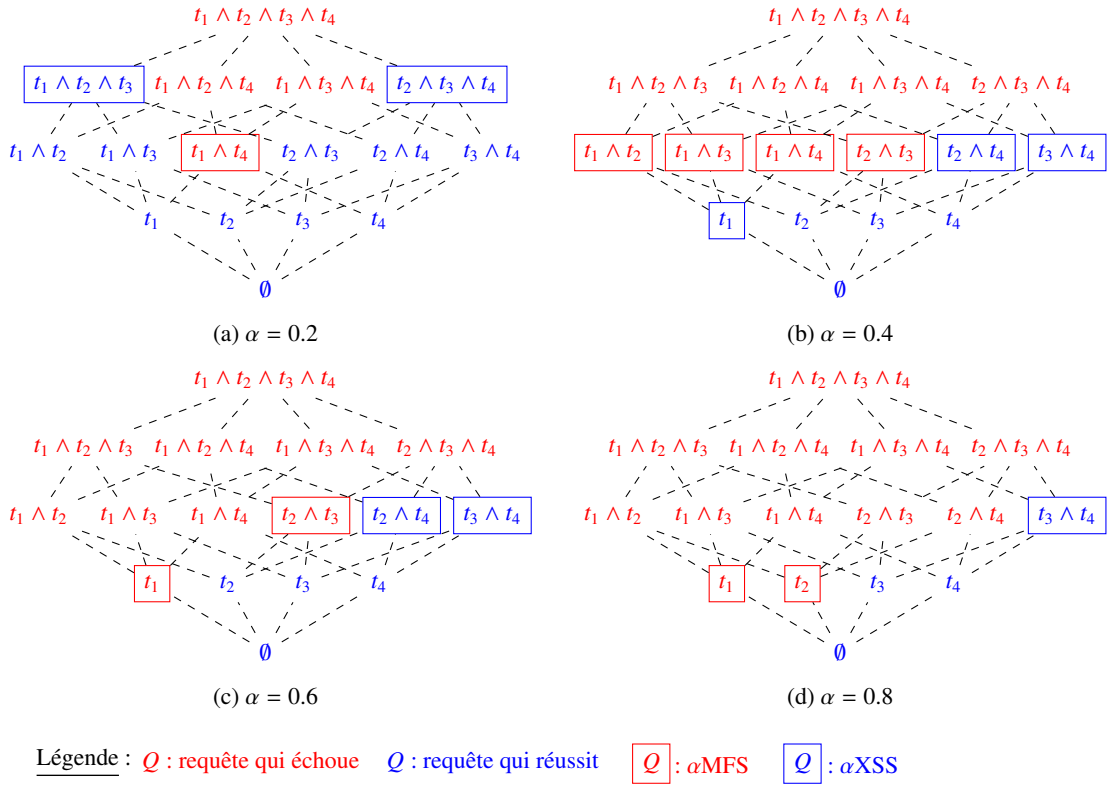


FIGURE 4.8 – Treillis des sous-requêtes avec des αMFS et αXSS pour différents α

Supposons qu'un utilisateur souhaite obtenir des résultats à la requête Q ayant un degré de confiance supérieur à 0.8. Dans notre exemple, cette requête va échouer. Grâce aux αMFS et αXSS pour le seuil 0.8, nous pouvons donner les explications suivantes : avec un degré de confiance supérieur à 0.8, il n'y a pas d'âge associé aux personnes (t_1), ni d'enseignant (`Lecturer`) (t_2) ; par contre, il y a au moins une personne intéressée par des recherches en Web sémantique qui enseigne les BD ($t_3 \wedge t_4$). Les αMFS et αXSS pour le seuil spécifié dans la requête permettent ainsi à l'utilisateur de comprendre les causes d'échec de sa requête et lui proposent des requêtes alternatives pour ce degré.

Cependant, l'utilisateur pourrait être enclin à baisser le degré de confiance désiré. Nous proposons donc de calculer les αMFS et αXSS pour des seuils inférieurs (par exemple, 0.2, 0.4 et 0.6). Cela permet de donner un retour plus significatif à l'utilisateur. Par exemple, les αXSS peuvent être utilisées pour donner les explications suivantes : (i) avec un degré de confiance supérieur à 0.6, il y a au moins un enseignant qui enseigne les BD ($t_2 \wedge t_4$), (ii) pour 0.4, il y a au moins un âge connu pour une personne (t_1) et (iii) pour 0.2, il y a au moins un enseignant intéressé par des recherches en Web sémantique dont l'âge est connu ($t_1 \wedge t_2 \wedge t_3$) et au moins un enseignant intéressé par des recherches en Web sémantique qui enseigne les BD ($t_2 \wedge t_3 \wedge t_4$).

Ces explications peuvent aider l'utilisateur à reformuler sa requête, à ajuster ses attentes sur le degré

de confiance voulu ou à avoir une meilleure compréhension du contenu de l'ontologie incertaine. Nous présentons, dans ce qui suit, les approches que nous proposons pour pouvoir retourner ces explications, c'est-à-dire calculer les αMFS et αXSS pour un ensemble de seuils de confiance. Nous commençons par considérer le problème du calcul des αMFS et αXSS pour un unique seuil α.

4.4.2.2 Découverte des αMFS et αXSS d'une requête RDF pour un seuil α

Nous avons montré dans [152] que l'approche LBA peut être directement adaptée au contexte des ontologies incertaines afin de calculer les αMFS et αXSS d'une requête RDF pour un unique seuil α. Cependant, cette approche repose sur l'hypothèse que, si une requête réussit, alors toutes ses sous-requêtes réussissent également. Dans le contexte des ontologies incertaines, selon la fonction d'agrégation choisie pour affecter un degré de confiance à un résultat, cette hypothèse n'est pas toujours vraie.

Exemple. La figure 4.9 illustre ce fait avec un exemple d'une requête Q et de sa sous-requête Q' exécutées sur les triplets RDF de la figure 4.7 (a) pour le seuil 0.8. Pour les fonctions d'agrégation *maximum* et *moyenne* (*avg*), la requête Q réussit alors que sa sous-requête Q' échoue. Ainsi, l'approche LBA ne peut pas être utilisée avec ces fonctions d'agrégation.

```
Q :   SELECT ?p WHERE {
      ?p type Professor . ?p researchInterest "SW" }
Q' :  SELECT ?p WHERE {
      ?p type Professor }
```

(a) La requête Q et sa sous-requête Q'

aggreg	$[[Q']]_D^{0.8}$	$[[Q]]_D^{0.8}$
min	$\emptyset$	$\emptyset$
max	$\emptyset$	$\{p_1\}$
$\prod$	$\emptyset$	$\emptyset$
avg	$\emptyset$	$\{p_1\}$

(b) Résultats de Q et Q'

FIGURE 4.9 – Résultats d'une requête Q et de sa sous-requête Q' avec différentes fonctions d'agrégation

Nous avons montré que l'approche LBA ne peut être utilisée pour les ontologies incertaines que si la fonction d'agrégation, choisie pour affecter un degré de confiance à un résultat, est *monotone décroissante* par rapport à la relation d'inclusion ensembliste. Comme exemples, nous pouvons citer le *minimum* ou le *produit* lorsqu'il s'applique à des valeurs réelles entre 0 et 1. Nous considérons dans les sections suivantes qu'une telle fonction est utilisée.

4.4.2.3 Découverte des αMFS et αXSS d'une requête RDF pour plusieurs seuils α

Pour trouver les αMFS et αXSS d'une requête RDF pour une ensemble de seuils $\alpha_i \in \{\alpha_1, \cdots \alpha_n\}$, la méthode directe consiste à utiliser l'approche LBA pour chaque seuil α_i. Nous appelons *NLBA* cette méthode de base. Nous présentons dans ce qui suit des approches plus efficaces. Elles se basent sur l'idée

que les αMFS et αXSS découvertes pour un seuil donné peuvent être utilisées pour déduire des αMFS et αXSS d'un seuil supérieur ou inférieur.

Approches Bottom-Up et Top-Down

L'approche *Bottom-Up* considère les seuils dans l'ordre croissant. Pour le premier seuil, l'approche LBA est utilisée pour calculer les αMFS et αXSS de ce seuil. Pour les seuils suivants, l'approche Bottom-Up se déroule en 2 étapes : (i) déduire des αMFS et αXSS pour le seuil considéré à partir de celles calculées précédemment et (ii) exécuter une version optimisée de LBA qui prend en paramètre les αMFS et αXSS identifiées précédemment. Nous ne détaillons ici que la première étape.

Soit α_i et α_j deux seuils tels que $\alpha_i < \alpha_j$, nous avons montré les propriétés suivantes.

- Les α_iXSS qui réussissent pour α_j sont des α_jXSS.
- Les α_iMFS *élémentaires*, c'est-à-dire ayant un seul patron de triplet, sont des α_jMFS.
- Les α_iMFS d'une taille > 1 échouent aussi pour α_j et contiennent donc une α_jMFS qui peut être trouvée avec l'algorithme *a_mel_fast* (voir section 4.3.2.2).

En se basant sur ces propriétés, un algorithme peut-être mis en place pour déduire des α_jMFS et α_jXSS à partir des α_iMFS et α_iXSS [152]. Nous illustrons cet algorithme sur un exemple.

Exemple. Pour illustrer l'approche Bottom-Up, nous considérons l'exemple de la figure 4.8. À partir des 0.6 αMFS et αXSS, nous montrons comment l'approche Bottom-Up calcule les 0.8 αMFS et αXSS. Comme t_1 est une 0.6 αMFS et ne contient qu'un seul patron de triplet, cette dernière est une 0.8 αMFS. La seconde 0.6 αMFS est $t_2 \wedge t_3$. Comme cette requête échoue nécessairement pour 0.8, nous utilisons l'algorithme *a_mel_fast* pour identifier la 0.8 αMFS t_2. Les 0.6 αXSS sont ensuite exécutées, et seule $t_3 \wedge t_4$ réussit pour 0.8. Ainsi, $t_3 \wedge t_4$ est une 0.8 αXSS. Les αMFS et αXSS découvertes sont respectivement : $\{t_1, t_2\}$ et $\{t_3 \wedge t_4\}$. Ainsi, dans cet exemple, toutes les 0.8 αMFS et αXSS ont été trouvées, sans avoir besoin d'exécuter l'algorithme LBA.

L'approche *Top-Down* suit les mêmes étapes que l'approche Bottom-Up, mais en considérant les valeurs de seuils dans l'ordre décroissant. Grâce à la relation de dualité entre les αMFS et αXSS, les propriétés et algorithmes utilisés dans cette approche sont similaires à ceux de l'approche Bottom-Up. Soit α_i et α_j deux seuils tels que $\alpha_i > \alpha_j$, nous avons montré les propriétés suivantes.

- Les α_iMFS qui échouent pour α_j sont des α_jMFS.
- Les α_iXSS d'une taille égale à $|Q| - 1$ sont des α_jXSS.
- Les α_iXSS d'une taille $< |Q| - 1$ réussissent aussi pour α_j et contiennent donc une α_jXSS qui peut être trouvée avec l'algorithme dual de *a_mel_fast*.

Exemple. Pour illustrer l'approche Top-Down, nous considérons à nouveau l'exemple de la figure 4.8. À partir des 0.8 αMFS et αXSS, nous montrons comment l'approche Top-Down calcule les 0.6 αMFS et αXSS. La 0.8 αMFS t_1 échoue pour 0.6, c'est donc une 0.6 αMFS. La 0.8 αXSS $t_3 \wedge t_4$ réussit nécessairement pour 0.6, cette dernière est utilisée comme paramètre de l'algorithme dual de *a_mel_fast* afin de trouver la 0.6 αXSS $t_3 \wedge t_4$. Ainsi, les 0.6 αMFS et αXSS découvertes sont respectivement $\{t_1\}$ et $\{t_3 \wedge t_4\}$. Celles-ci sont utilisées comme paramètre de la version optimisée de LBA qui identifie alors les autres 0.6 αMFS ($t_2 \wedge t_3$) et αXSS ($t_2 \wedge t_4$).

Approche Hybride

L'idée de l'approche *Hybride* est de combiner les propriétés utilisées par les approches Bottom-Up et Top-Down pour découvrir le plus grand nombre possible d'αMFS et αXSS. Cette approche parcourt les seuils en alternant entre le plus petit et le plus grand seuil restant. Dans notre exemple avec les seuils $\{0.2, 0.4, 0.6, 0.8\}$, l'approche Hybride considère ces derniers dans l'ordre suivant : $\{0.2, 0.8, 0.4, 0.6\}$. Grâce à cet ordre, lors de la recherche des αMFS et αXSS pour les seuils 0.4 et 0.6, l'approche Hybride a accès non seulement aux αMFS et αXSS d'un seuil inférieur mais aussi à celles d'un seuil supérieur. Elle peut ainsi bénéficier des propriétés utilisées dans les approches Bottom-Up et Top-Down pour découvrir des αMFS et αXSS. De plus, nous pouvons utiliser les propriétés suivantes où α_i, α_j et α_k trois seuils tels que $\alpha_i < \alpha_j < \alpha_k$.

- Si une requête Q^* est à la fois α_iXSS et α_kXSS, alors Q^* est une α_jXSS.
- Si une requête Q^* est à la fois α_iMFS et α_kMFS, alors Q^* est une α_jMFS.

Exemple. Nous illustrons l'approche Hybride sur l'exemple courant (voir figure 4.8). Nous montrons comment elle calcule les 0.6 αMFS et αXSS, connaissant celles pour 0.4 et 0.8. L'approche Hybride identifie $t_3 \wedge t_4$ comme une αXSS pour 0.4 et 0.8, c'est donc également une 0.6 αXSS. Puis, l'approche Hybride recherche les 0.8 αMFS qui échouent pour 0.6 (propriété de l'approche Top-Down). C'est le cas pour t_1, qui est une 0.6 αMFS. De même, il recherche les 0.4 αXSS qui réussissent pour 0.6 (propriété de l'approche Bottom-Up) et trouve la 0.6 αXSS $t_2 \wedge t_4$. L'algorithme *a_mel_fast* est ensuite utilisé avec les 0.4 αMFS qui échouent. Cela permet d'identifier que $t_2 \wedge t_3$ est une 0.6 αMFS. Ainsi, grâce aux propriétés des approches Bottom-Up et Top-Down, l'approche Hybride a calculé toutes les 0.6 αMFS et αXSS sans avoir eu besoin d'exécuter la version optimisée de LBA.

4.4.3 Résultats

Nous avons présenté nos approches d'identification des αMFS et αXSS dans le contexte des ontologies incertaines. Nous avons d'abord montré que l'approche LBA peut être utilisée dans ce contexte si et seulement si la fonction d'agrégation choisie pour affecter un degré de confiance à un résultat est *monotone décroissante* par rapport à la relation d'inclusion ensembliste. Des fonctions usuelles comme le minimum ou le produit, lorsqu'il s'applique à des valeurs réels entre 0 et 1, respectent cette condition.

Ensuite, pour pouvoir donner des explications sur ce qui se passerait si l'utilisateur décidait de diminuer le seuil de confiance voulu, nous avons proposé des approches d'identification des αMFS et αXSS pour plusieurs seuils. En comparaison avec la méthode de base NLBA, qui consiste à exécuter LBA pour chaque seuil, nos approches se basent sur les αMFS et αXSS découvertes pour un seuil donné afin d'en déduire pour un seuil supérieur ou inférieur. Nous avons proposé dans [153] une étude de complexité sur ces différentes approches en termes de nombre de requêtes exécutées. Nous avons montré que, dans le pire des cas, nos approches n'exécuteront jamais plus de requêtes que NLBA. Nous avons aussi identifié les conditions qui font que nos approches exécuteront moins de requêtes (par exemple, plus les αXSS auront de patrons de triplet, moins l'approche Top-Down exécutera de requêtes).

Notre implémentation Java des approches décrites dans cette section est disponible sur `https://forge.lias-lab.fr/projects/qars4ukb`. Nous avons mené des expérimentations sur le banc d'essai *WatDiv* [43] en générant des degrés de confiance aléatoirement. Nous les avons réalisées sur

les triplestores Jena TDB et Virtuoso en considérant plusieurs tailles de données et types de requêtes. Nous avons évalué les temps d'exécution et le nombre de requêtes exécutées par nos approches Bottom-Up, Top-Down et Hybride en comparaison avec NLBA (c'est-à-dire, exécuter LBA pour chaque seuil). Ces expérimentations nous ont permis de tirer les conclusions suivantes.

- Nos approches Bottom-Up, Top-Down et Hybride exécutent moins de requêtes que NLBA (respectivement 21%, 12% et 16% de moins pour les requêtes considérées) et ont, en conséquence, un plus faible temps d'exécution (une diminution de respectivement 30%, 42% et 33%).
- Aucune de nos approches n'est plus efficace que les autres pour toutes les requêtes. En moyenne, l'approche Top-Down exécute le plus de requêtes mais a le meilleur temps d'exécution. Cela vient du fait que les requêtes exécutées par Top-Down concernent généralement un seuil élevé. Elles sont ainsi sélectives et donc peu coûteuses en termes de temps d'exécution.
- L'approche Hybride n'obtient jamais les meilleures performances. Dans nos expérimentations, les αMFS et αXSS changent à chaque seuil exploré et, donc, les propriétés spécifiques de l'approche Hybride ne sont pas exploitées. Nous pensons que ce ne serait pas le cas avec des jeux de données réels où les degrés de confiance ne seraient pas définis aléatoirement.

Les approches proposées dans les sections précédentes identifient les MFS et XSS d'une requête RDF exprimée sur une ontologie, dont les données sont certaines ou incertaines. Les XSS fournissent à l'utilisateur des requêtes relaxées par suppression de patrons de triplet. Il est également possible de les rendre moins sélectifs. Nous abordons ce cas dans la section suivante. Pour simplifier, nous le faisons sans prendre en compte l'incertitude des données sémantiques même si les approches proposées pourraient s'appliquer aux ontologies incertaines.

4.5 Contribution 3 : approches de relaxation basées sur les causes d'échec

Nous commençons par expliquer les limites de l'état de l'art sur la relaxation de requêtes sémantiques qui ont motivé nos travaux.

4.5.1 Contexte et motivation : relaxation de requêtes sémantiques

Le problème des réponses vides étant apparu comme important dans le contexte des ontologies, de nombreux travaux ont été menés pour fournir des techniques de relaxation pour des requêtes sémantiques [17,19,20,138,154–158]. Ces propositions incluent différents opérateurs de relaxation qui peuvent être basés sur le schéma d'une ontologie (par exemple, remplacer une classe ou propriété utilisée dans une requête par une super-classe ou super-propriété) [17, 19, 20, 138, 154, 155], sur des mesures de similarité [156, 157] ou sur des préférences utilisateur [158]. Dans les approches proposées, ces opérateurs sont utilisés de différentes façons :

- pour retourner les *top-K réponses* [20] à l'utilisateur [17, 19, 20] ;
- pour être inclus directement dans des requêtes SPARQL [138, 154, 155] ;
- pour être appelés dans des règles de réécriture de requêtes [158].

20. K meilleures réponses, où K est un paramètre défini par l'utilisateur

Dans nos travaux, nous nous sommes intéressés aux approches retournant les top-K réponses à l'utilisateur et en particulier à l'approche de référence proposée par Huang *et al.* [17]. Dans ces approches, les opérateurs de relaxation sont appliqués à la requête initiale de l'utilisateur pour générer un ensemble de requêtes relaxées. Celles-ci sont ensuite triées, avec une mesure de similarité, puis exécutées par ordre décroissant (c'est-à-dire, en commençant par la plus similaire à la requête utilisateur) jusqu'à obtenir au moins K résultats. Chaque patron de triplet pouvant être relaxé plusieurs fois et de différentes façon, ce processus de relaxation peut exécuter de nombreuses requêtes relaxées (c'est-à-dire, un nombre exponentiel par rapport à la taille de la requête) avant d'obtenir les top-K réponses escomptées .

Ce problème vient du fait que ce processus de relaxation est fait *à l'aveugle*, c'est-à-dire sans savoir pourquoi la requête utilisateur a échoué. Ainsi, ce processus peut exécuter de nombreuses requêtes inutilement car celles-ci relaxent des patrons de triplet de la requête qui ne font pas partie des raisons pour lesquelles la requête utilisateur a échoué. Notre idée était donc que les MFS, que nos approches présentées précédemment calculent, pourraient être utiles pour limiter le nombre de requêtes relaxées exécutées et ainsi accélérer le processus de relaxation.

À notre connaissance, seuls les travaux de Bosc *et al.* [140], effectués dans le contexte des requêtes floues exprimées sur des BDR, ont également émis l'idée d'utiliser les MFS pour accélérer le processus de relaxation d'une requête. Cependant, l'approche proposée dans ces travaux n'est possible que lorsque l'ensemble des MFS de la requête relaxée est un sous-ensemble de celles de la requête utilisateur. Nous présentons dans ce qui suit plusieurs approches basées sur les MFS qui ne nécessitent pas cette hypothèse et qui sont définies pour le contexte des requêtes sémantiques exprimées sur des BDS.

4.5.2 Principe général de nos approches de relaxation basées sur les causes d'échec

Un patron de triplet RDF peut-être relaxé de plusieurs façons. Nous présentons dans ce qui suit le modèle de relaxation défini dans [17]. Nous avons utilisé ce modèle pour pouvoir comparer nos approches à celle proposée dans cet article. Dans cette section, nous ne donnons que les grands principes de nos approches. Le lecteur intéressé pourra consulter [142] pour avoir le détail des algorithmes et les preuves des propriétés utilisées.

4.5.2.1 Modèle de relaxation considéré

Soit t un patron de triplet, t' est un *patron de triplet relaxé* de t si $t' \neq t$ et, pour toutes BDS contenant des triplets RDF D, $[[t]]_D \subseteq [[t']]_D$. Nous utilisons la notation $t \prec t'$. Une *règle de relaxation* est une règle de réécriture qui transforme un patron de triplet t en un patron de triplet relaxé de t. Nous considérons les règles de relaxation suivantes :

- *relaxation en remplaçant une classe par une de ses super-classes (R1)* ;
- *relaxation en remplaçant une propriété par une de ses super-propriétés (R2)* ;
- *relaxation en remplaçant une constante par une variable (R3).*

Soit $Q = t_1 \wedge \cdots \wedge t_n$ et $Q' = t'_1 \wedge \cdots \wedge t'_n$ deux requêtes RDF, Q' est une *requête relaxée* de Q, si (i) pour chaque patron de triplet t_i, $t_i \preceq t'_i$ ($t_i = t'_i$ ou $t_i \prec t'_i$) et (ii) pour au moins un patron de triplet t_j, $t_j \prec t'_j$. Nous utilisons la notation $Q \prec Q'$.

Exemple. La figure 4.10 (c) présente une requête $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$ exécutée sur la BDS composée des triplets RDF donnés dans la figure 4.10(a). Cette requête recherche les enseignants (`Lecturer`) de 46 ans, avec leur nationalité, qui enseignent le Web sémantique (`SW`). Une requête relaxée de Q, nommée Q' est présentée dans la figure 4.10 (d). Dans cet exemple, $t_3^{(1)}$ est un patron de triplet relaxé de t_3 obtenu en appliquant la règle de relaxation R3. Cette requête recherche les enseignants de 46 ans, avec leur nationalité, qui enseignent au moins un cours. Les MFS de Q et Q' sont données dans la figure 4.10 (b).

Triplets		
sujet	**prédicat**	**objet**
p_1	type	Lecturer
p_1	teacherOf	SW
p_1	age	45
p_2	type	Lecturer
p_2	nationality	US
p_2	age	46
p_3	type	FullProfessor
p_3	teacherOf	DB
p_3	age	46

(a) Triplets RDF

$mfs(Q) = \{t_2 \wedge t_3, t_3 \wedge t_4\}$
$mfs(Q') = \{t_2 \wedge t_3^{(1)},\ t_1 \wedge t_3^{(1)} \wedge t_4\}$

(b) Les MFS de Q et Q'

```
SELECT ?p ?n WHERE {
?p type Lecturer        (t1)
?p nationality ?n       (t2)
?p teacherOf SW         (t3)
?p age 46 }             (t4)
```

(c) Une requête Q

```
SELECT ?p ?n WHERE {
?p type Lecturer        (t1)
?p nationality ?n       (t2)
?p teacherOf ?c         (t3(1))
?p age 46 }             (t4)
```

(d) Une requête relaxée Q' de Q

FIGURE 4.10 – Exemple d'une requête Q et d'une requête relaxée de Q

Nous utilisons les formules définies dans [17] pour calculer la similarité entre une requête Q et sa requête relaxée Q'. Soit D une BDS contenant des triplets RDF, si $\mu \in [[Q']]_D$ et $\mu \notin [[Q]]_D$, alors μ est une *réponse approximative* de Q. Un score est attribué à chaque réponse approximative en se basant sur la similarité entre Q et Q'. L'objectif des travaux présentés dans cette section est de proposer des approches les plus efficaces possibles pour trouver les top-K réponses approximatives de Q. Nous définissons d'abord les structures de données sur lesquelles s'appuient ces approches.

4.5.2.2 Structures de données pour la relaxation de requêtes

Un patron de triplet t peut être relaxé en appliquant plusieurs fois des règles de relaxation. Nous notons $t^{(0)}$ le patron de triplet initial et $t^{(i)}$ le patron de triplet qui est la i-ème meilleure relaxation de t en termes de similarité avec t. Enfin, nous notons $nbRel(t)$ le nombre de relaxations de t. La liste des patrons de triplet relaxés de t, ordonnée par similarité, peut être calculée par un algorithme qui applique récursivement les règles de relaxation sur le patron de triplet initial et ses patrons de triplet relaxés [142].

Exemple. Supposons que *FullProfessor* soit une sous-classe de *Professor*. Un exemple de relaxation du patron de triplet $t = (?X,\ type,\ FullProfessor)$, qui s'appuie sur nos trois règles de relaxation (*R1*, *R2* et *R3*), est présenté dans la figure 4.11. Dans cet exemple, $nbRel(t) = 5$.

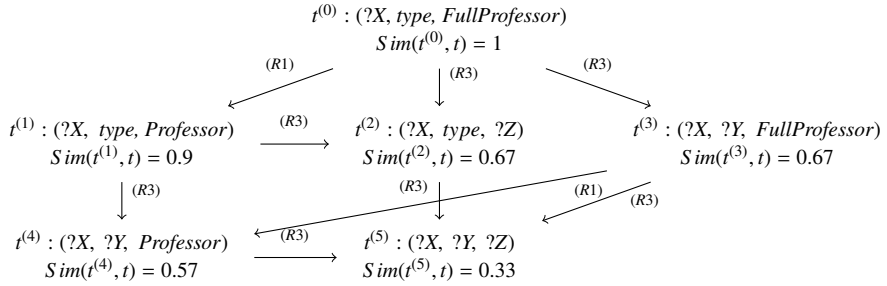


FIGURE 4.11 – Exemple de relaxation d'un patron de triplet

Les requêtes relaxées de la requête utilisateur sont organisées dans un graphe, appelé *graphe de relaxation*. Les nœuds de ce graphe sont la requête utilisateur et ses requêtes relaxées. Une arête allant du nœud $Q_i = t_1^{(i_1)} \wedge \cdots \wedge t_n^{(i_n)}$ à $Q_j = t_1^{(j_1)} \wedge \cdots \wedge t_n^{(j_n)}$ est présente si et seulement si (i) pour un patron de triplet t_l, $i_l = j_l - 1$ et (ii) pour les autres patrons de triplet t_k, $i_k = j_k$. Le nombre de requêtes relaxées dans le graphe de relaxation est : $\prod_{i=1}^{n}(nbRel(t_i) + 1)$.

Exemple. La figure 4.12 donne un exemple du graphe de relaxation de notre requête $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$. Pour des raisons de lisibilité, nous supposons dans cet exemple que chaque patron de triplet ne peut être relaxé qu'une seule fois.

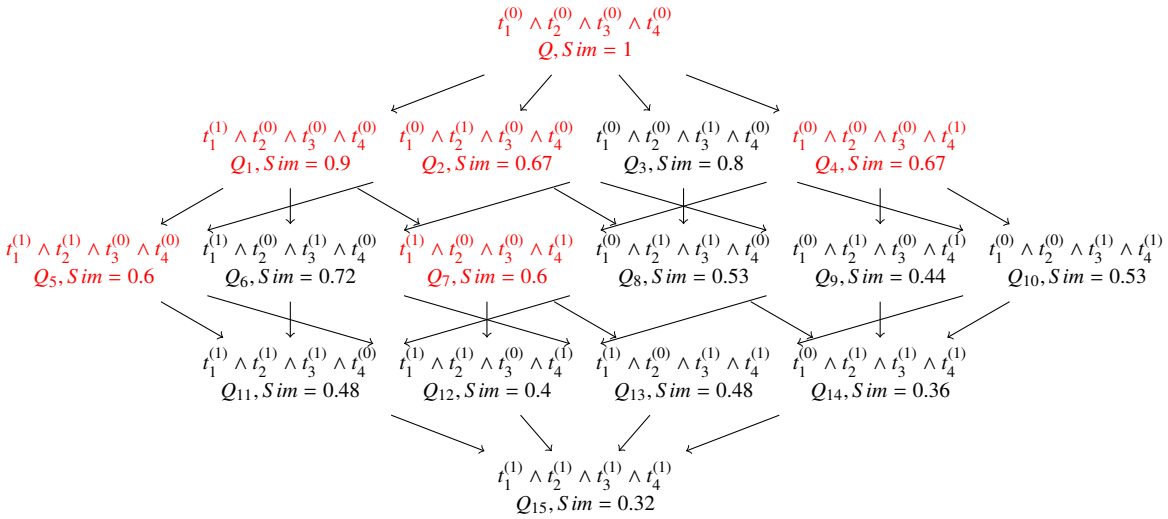


FIGURE 4.12 – Exemple de graphe de relaxation

4.5.2.3 Approches de relaxation

Nous présentons différentes approches de relaxation qui se basent sur les structures de données définies précédemment. Nous commençons par l'approche de référence proposée dans la littérature.

Méthode de l'état de l'art

L'approche *BFS (Best-First Search)* proposée par Huang *et al.* [17] consiste (i) à trier les requêtes

relaxées du graphe de relaxation selon leur similarité avec la requête utilisateur (de la plus à la moins similiaire) et (ii) à les exécuter dans cet ordre jusqu'à trouver le nombre de résultats attendus par l'utilisateur. Par exemple, BFS explore le graphe de relaxation présenté dans la figure 4.12 dans l'ordre suivant : $Q_1, Q_3, Q_6, Q_2, Q_4, Q_5, Q_7, Q_8, Q_{10}, Q_{11}, Q_{13}, Q_9, Q_{12}, Q_{14}, Q_{15}$. Dans le meilleur cas, où la requête Q_1 retourne le nombre de résultats attendus, BFS exécute une seule requête. Dans le pire cas, où le nombre de résultats attendus n'est pas obtenu avant d'exécuter Q_{15}, BFS exécute toutes les requêtes du graphe de relaxation. Comme celui-ci est composé d'un nombre exponentiel de requêtes relaxées par rapport à la taille de la requête ($\prod_{i=1}^{n}(nbRel(t_i)+1)$), BFS peut nécessiter un temps d'exécution important.

Dans l'approche BFS, les causes d'échec de la requête utilisateur sont inconnues. Ainsi, elle peut exécuter des requêtes relaxées qui échouent car contenant toujours une cause d'échec. Elle peut aussi relaxer des patrons de triplet qui n'ont pas besoin d'être modifiés, car ils ne font pas partie des causes d'échec. Aussi, nous avons proposé plusieurs approches basées sur les MFS pour éviter ces écueils.

Nos approches basées sur les MFS

MFS-Based Search (MBS). Nous avons montré que, si une requête relaxée Q' de Q ne relaxe pas au moins un patron de triplet de chaque MFS de Q, alors elle échoue nécessairement. Aussi, l'approche MBS suit le principe de BFS (c'est-à-dire, exécuter les requêtes relaxées dans l'ordre de similarité avec la requête utilisateur) mais évite d'exécuter des requêtes relaxées qui échouent nécessairement, car contenant toujours une MFS de la requête utilisateur.

Exemple. Si les MFS de notre exemple de requête $Q = t_1 \wedge t_2 \wedge t_3 \wedge t_4$ sont $t_2 \wedge t_3$ et $t_3 \wedge t_4$, alors toutes les requêtes en rouge dans la figure 4.12 (Q_1, Q_2, Q_4, Q_5, Q_7) échouent nécessairement et ne sont donc pas exécutées par MBS. Ainsi, cet algorithme exécute les requêtes du graphe de relaxation dans l'ordre suivant : $Q_3, Q_6, Q_8, Q_{10}, Q_{11}, Q_{13}, Q_9, Q_{12}, Q_{14}, Q_{15}$.

Optimized MFS-Based Search (O-MBS). Dans l'approche précédente, nous n'utilisons que les MFS de la requête utilisateur pour élaguer l'espace de recherche. L'idée sous-jacente à l'approche O-MBS est d'utiliser également les MFS des requêtes relaxées de Q qui échouent. Le calcul des MFS étant coûteux, nous avons montré que les MFS de la requête utilisateur Q peuvent être utilisées pour en déduire des MFS d'une requête relaxée Q' de Q. Une fois que des MFS d'une requête relaxée ont ainsi été identifiées, elles peuvent alors être utilisées pour élaguer à nouveau l'espace de recherche.

Précisons maintenant comment nous déduisons des MFS de Q' à partir de celles de Q. Soit M_Q une MFS de Q ; nous notons $M_Q^{\uparrow Q'}$ la requête qui correspond à M_Q dans Q'. Par extension, nous notons $mfs^{\uparrow Q'}(Q)$ les requêtes qui correspondent aux MFS de Q dans Q'. Dans notre exemple, présenté dans la figure 4.10, $mfs^{\uparrow Q'}(Q) = \{t_2 \wedge t_3^{(1)},\ t_3^{(1)} \wedge t_4\}$. Nous avons montré que toutes les requêtes de $mfs^{\uparrow Q'}(Q)$ qui échouent sont des MFS de Q'.

O-MBS étend MBS de la façon suivante. Pour chaque requête relaxée Q' explorée dans le graphe de relaxation, MBS exécute chaque requête $M_{Q'} \in mfs^{\uparrow Q'}(Q)$. Si la requête $M_{Q'}$ échoue, c'est une MFS de Q' et donc, toutes les requêtes qui contiennent $M_{Q'}$ peuvent être enlevées du graphe de relaxation.

Exemple. Reprenons notre exemple présenté dans la figure 4.12. Supposons que la requête Q_3 ne résout pas la MFS $t_2 \wedge t_3$ (c'est-à-dire que $t_2 \wedge t_3^{(1)}$ échoue). Alors, les requêtes Q_6, Q_{10} et Q_{13} ne sont pas exécutées par O-MBS. Si aucune autre MFS n'est trouvée par la suite, O-MBS exécute les requêtes du graphe de relaxation dans l'ordre suivant : $Q_3, Q_8, Q_{11}, Q_9, Q_{12}, Q_{14}, Q_{15}$.

Full MFS-Based Search (F-MBS). Dans l'approche O-MBS, toutes les MFS des requêtes relaxées explorées ne sont pas nécessairement découvertes. Nous avons donc proposé une approche, nommée F-MBS, qui recherche toutes les MFS des requêtes relaxées explorées. En proposant cette approche, nous souhaitions savoir s'il est judicieux de rechercher toutes les MFS. Autrement dit, si le temps de calcul de ces MFS est inférieur au gain que ces MFS permettent d'obtenir en évitant d'exécuter des requêtes relaxées du graphe de relaxation.

F-MBS étend l'approche O-MBS comme suit. Pour chaque requête relaxée Q', nous exécutons toutes les MFS de $mfs^{\uparrow Q'}(Q_0)$, où Q_0 est la dernière requête relaxée explorée telle que $Q_0 \prec Q'$. Nous avons montré que, si toutes ces requêtes échouent, alors $mfs(Q') = mfs^{\uparrow Q'}(Q_0)$. Sinon, nous exécutons une version de l'algorithme LBA (voir section 4.3.2.2), qui prend en paramètre les MFS déjà identifiées, afin de trouver toutes les MFS de Q'. Comme dans les approches précédentes, les requêtes qui incluent au moins une des MFS de Q' sont enlevées du graphe de relaxation.

Exemple. Revenons à notre exemple présenté dans la figure 4.12 et supposons que $mfs(Q_3) = \{t_2 \wedge t_3^{(1)},\ t_1 \wedge t_3^{(1)} \wedge t_4\}$. Alors, les requêtes Q_6, Q_{10}, Q_{13} et Q_8 sont enlevées du graphe de relaxation. Si les MFS des requêtes relaxées suivantes n'aident pas à élaguer davantage le graphe de relaxation, alors F-MBS exécute les requêtes relaxées dans l'ordre suivant : $Q_3, Q_{11}, Q_9, Q_{12}, Q_{14}, Q_{15}$.

4.5.3 Résultats

Les approches d'identification des MFS que nous avons développées dans nos travaux fournissent à l'utilisateur les causes d'échec de sa requête. Cependant, il peut être difficile d'interpréter ces MFS, surtout lorsqu'elles sont nombreuses. Cette étape est pourtant nécessaire pour pouvoir reformuler correctement la requête. Dans cette section, nous avons montré que les MFS peuvent aussi être utilisées pour relaxer automatiquement la requête et retourner top-K réponses à l'utilisateur. Nous avons ainsi proposé trois approches de relaxation d'une requête RDF basées sur les MFS. Leur originalité est d'utiliser plus ou moins d'information sur les MFS de la requête utilisateur et de ses requêtes relaxées qui échouent.

Ces approches ont été implémentées en Java. Elles prennent comme paramètres d'entrée une requête RDF qui échoue et un nombre minimum K de résultats voulus. Elles retournent au moins K réponses approximatives pour cette requête. Cette implémentation est disponible sur `https://forge.lias-lab.fr/projects/qars`. Nous avons réalisé des expérimentations sur le banc d'essai *LUBM* [42] avec différents types de requêtes, plusieurs tailles de données et sur les deux triplestores Jena TDB et Virtuoso. Dans celles-ci, nous avons analysé le nombre de requêtes exécutées et le temps d'exécution de nos approches MBS, O-MBS et F-MBS en comparaison avec l'algorithme de référence BFS. Ces expérimentations nous ont permis de tirer les conclusions suivantes [142].

- BFS exécute plus de requêtes que nos approches et cet écart augmente avec la taille des requêtes. Cela peut impliquer une différence de temps d'exécution plus ou moins importante selon les requêtes que nos approches évitent d'exécuter par rapport à BFS. En moyenne, notre approche la plus efficace, O-MBS, est plus rapide que BFS par un facteur supérieur à 2.
- O-MBS est meilleure que les autres approches pour la plupart des requêtes. En analysant les requêtes relaxées exécutées, nous avons observé que O-MBS, grâce aux MFS des requêtes relaxées qu'elle déduit, se focalise sur le patron de triplet qu'il est nécessaire de relaxer. Ce n'est pas tou-

jours le cas de MBS qui exécute alors de nombreuses requêtes relaxées inutilement car basées sur une relaxation non nécessaire d'un patron de triplet. De son côté, F-MBS exécute toujours plus de requêtes que O-MBS car le nombre de requêtes qu'elle exécute pour trouver toutes les MFS est trop important par rapport au nombre de requêtes relaxées qu'elle évite d'exécuter.

En conclusion, nous considérons que l'approche O-MBS est celle qui doit être utilisée en pratique. Elle représente un bon compromis entre MBS qui n'utilise pas d'information sur l'échec des requêtes relaxées et F-MBS qui en utilise trop.

4.6 Bilan sur les techniques coopératives proposées pour les BDS

Les travaux menés dans l'axe de recherche présenté dans ce chapitre nous ont permis d'apporter des contributions à l'état de l'art sur les aspects décrits dans la section suivante.

4.6.1 Contributions à l'état de l'art

Calcul des MFS et XSS. Le tableau 4.1 présente les principales approches proposées dans la littérature pour calculer les MFS ou XSS d'une requête qui échoue. À notre connaissance, nos travaux sont les premiers à calculer simultanément les MFS et XSS dans le contexte de SPARQL/RDF et a ainsi prendre en compte les spécificités de ces langages par rapport au contexte relationnel (par exemple, la différence de sémantique sur le traitement des valeurs *null* entre SPARQL et SQL). Par ailleurs, nos travaux nous ont amenés à considérer le caractère incertain des données RDF et à proposer des approches qui calculent les MFS et XSS pour différents degrés de confiance (*MFS et XSS graduelles*). Enfin, nous avons veillé à ce que l'implémentation des approches proposées dans nos travaux soit disponible en ligne et puisse être utilisée sur différentes BDS (par exemple, Jena TDB et Virtuoso).

Approches	Contexte	Données	Calcul	Type d'approche
ISHMAEL [124]	SQL/BDR	Certaines	MFS ou XSS	Parcours de treillis
Jannach [125]	SQL/BDR	Certaines	XSS	Approche matricielle
McSherry [139]	SQL/BDR	Certaines	MFS	Parcours de treillis
Pivert et Smits [150]	SQLf/BDR	Certaines	MFS et XSS graduelles	Parcours de treillis basé sur un résumé de la BDR
Matrix-BS [159]	SQL/BDR	Incertaines	MFS et XSS	Approche matricielle
LBA [142]	SPARQL/RDF	Certaines	MFS et XSS	Parcours de treillis
MBA [142]	SPARQL/RDF	Certaines	MFS et XSS	Approche matricielle
Bottom-Up, Top-Down, Hybride [152]	SPARQL/RDF	Incertaines	MFS et XSS graduelles	Parcours de plusieurs treillis

TABLE 4.1 – Principales approches de calcul des MFS et XSS (les nôtres sont indiquées en italique)

Utilisation des MFS et XSS. Le tableau 4.2 présente les principales approches proposées dans la littérature pour utiliser les MFS ou XSS. Dans la plupart des approches calculant les MFS ou XSS,

ces dernières sont présentées à l'utilisateur qui a la charge de les interpréter pour reformuler sa requête (par exemple, [124]). Certaines approches, comme celles de McSherry [139], Jannach [125] et Mottin *et al.* [160], proposent d'utiliser les MFS ou XSS pour relaxer interactivement une requête. Elles consistent à proposer à l'utilisateur des relaxations de certaines parties de sa requête. Celui-ci indique alors s'il souhaite faire ces relaxations. Dans nos travaux, nous avons montré que les MFS peuvent également être utiles à la construction d'approches automatiques de relaxation de requêtes [142]. Grâce à la connaissance des causes d'échec de la requête, les approches basées sur les MFS sont plus efficaces que l'approche standard de l'état de l'art qui consiste à exécuter des requêtes relaxées jusqu'à trouver le nombre de résultats attendus par l'utilisateur [17].

Approches	Contexte	MFS/XSS	Type d'approche
Godfrey [124]	SQL/BDR	MFS ou XSS	Approche manuelle
Jannach [125]	SQL/BDR	MFS et XSS	Approches interactive et automatique
McSherry [139]	SQL/BDR	MFS	Approche interactive
Mottin *et al.* [160]	SQL/BDR	MFS	Approche interactive
Bosc *et al.* [140]	SQLf/BDR	MFS	Approche automatique
BFS [17]	SPARQL/RDF	Aucune	Approche automatique
MBS, O-MBS, F-MBS [142]	SPARQL/RDF	MFS	Approches automatiques

TABLE 4.2 – Principales approches d'utilisation des MFS et XSS (les nôtres sont indiquées en italique)

4.6.2 Production scientifique

Le tableau 4.3 synthétise notre production scientifique liée aux travaux de recherche présentés dans ce chapitre. Ils ont débuté en 2012 et ont été réalisés dans le cadre des thèses soutenues de Géraud Fokou (co-encadrement avec Allel Hadjali) [161] et Ibrahim Dellal (co-encadrement avec Allel Hadjali et Brice Chardin) [162]. Je me suis également fortement investi dans la thèse de Chourouk Belheouanne [163] sans être encadrant officiel [21]. Enfin, la thèse de Louise Parkin est actuellement en cours sur les perspectives ouvertes par nos travaux.

Production	Description
Publications	3 RI, 1 RN, 6 CI, 4 CN, 1 API
Encadrements	Géraud Fokou (2016), Ibrahim Dellal (2019), Chourouk Belheouanne (2019)
doctoraux	Louise Parkin (en cours)
Projets et contrats	QDOSSI (projet CNRS), Good Morning Planet (prestation)
Logiciels	QaRS, QaRS4UKB et MFS4UDB

TABLE 4.3 – Production scientifique liée à nos travaux sur les techniques coopératives pour les BDS

Ces travaux, et plus généralement cet axe de recherche, ont donné lieu à 3 RI, 1 RN, 6 CI, 4 CN et 1

21. Référence Allel Hadjali, allel.hadjali@ensma.fr

API (démonstration). Parmi ces publications, les principales sont les suivantes.

- Un article de démonstration dans la conférence internationale *International Conference on Extending Database Technology (EDBT 2015)* [164] qui présente le logiciel QaRS (Query and Relax System) développé dans le cadre de nos travaux de recherche sur les MFS et XSS.
- Un article dans la conférence internationale *Extended Semantic Web Conference (ESWC 2015)* [165] qui a fait l'objet d'une extension dans la revue internationale *Knowledge and Information Systems (KAIS)* [144]. Ces articles présentent en détails les approches LBA et MBA (voir section 4.3).
- Un article dans la conférence internationale *International Conference on Database and Expert Systems Applications (DEXA 2017)* [152] qui a fait l'objet d'une extension dans la revue internationale KAIS [153]. Ces articles présentent l'adaptation de l'approche LBA dans le contexte des ontologies incertaines (voir section 4.4).
- Un article dans la conférence internationale ESWC 2016 [142] qui présente nos approches de relaxation basées sur les MFS (voir section 4.5). Cet article a obtenu le prix du *best research paper*.

Ces travaux ont impliqué le développement des logiciels QaRS [22] (7K lignes de code, environ 200 téléchargements), QaRS4UKB [23] (3K lignes de code, environ 100 téléchargements) et MFS4UDB [24] (1,5K lignes de code, environ 100 téléchargements) qui ont été utilisés dans le cadre du projet de recherche *QDOSSI* (projet CNRS) et d'une prestation avec l'entreprise *Good Morning Planet*.

4.7 Conclusion et perspectives de recherche

Nous avons présenté dans ce chapitre nos propositions de techniques coopératives pour les BDS. Le fil directeur de nos travaux a été d'essayer de comprendre en premier lieu pourquoi la requête utilisateur échoue. Ces travaux nous ont ainsi conduits à proposer des approches d'identification des MFS et XSS pour le contexte des requêtes sémantiques. Inspirés par les deux principales approches de la littérature, nous avons proposé les approches LBA, basée sur un parcours du treillis des sous-requêtes de la requête utilisateur, et MBA, basée sur le calcul d'une matrice. Dans le contexte relationnel, l'approche matricielle est celle qui est considérée comme étant la plus efficace [150, 159]. Nous avons montré que ce ne n'est pas le cas dans le contexte de SPARQL/RDF à cause de la différence de sémantique sur le traitement des valeurs *null* entre SPARQL et SQL. Le fait qu'en SPARQL une jointure ne rejette pas les valeurs *null*, contrairement à SQL, fait que la matrice d'une requête sémantique à une taille importante qui implique un coût de calcul rédhibitoire.

Nous avons ensuite considéré une caractéristique de plusieurs ontologies réelles qui est d'associer un degré de confiance à chacun de leurs faits. Cette caractéristique fait qu'un utilisateur peut exprimer une requête en exigeant que les résultats aient un degré de confiance supérieur à un seuil α. Si une telle requête échoue, cela peut venir de celle-ci mais, aussi, du seuil défini. Pour répondre à cette possibilité, nous avons d'abord établi la condition nécessaire pour que l'approche LBA puisse être utilisée dans ce nouveau contexte pour l'identification de ce que nous avons appelé les αMFS et αXSS. Puis, nous avons proposé des approches les calculant pour différents seuils α afin que l'utilisateur puisse savoir ce qui se

22. https://forge.lias-lab.fr/projects/qars
23. https://forge.lias-lab.fr/projects/qars4ukb
24. https://forge.lias-lab.fr/projects/mfs4udb

passerait s'il décidait de baisser le seuil initialement spécifié. Plutôt que d'exécuter naïvement l'approche LBA pour chaque seuil α, nos approches essaient de déduire des αMFS et αXSS d'un seuil donné à partir de celles calculées précédemment pour d'autres seuils .

Les MFS et XSS que nos approches calculent peuvent être nombreuses et donc difficilement exploitables par l'utilisateur. Aussi, nous nous sommes intéressés à leur utilisation dans des approches automatiques de relaxation visant à fournir top-K réponses approximatives à l'utilisateur. Nous avons montré que les MFS permettent d'éviter d'exécuter de nombreuses requêtes relaxées qui échoueront nécessairement car contenant toujours une cause d'échec de la requête utilisateur. Nos expérimentations ont montré que nos approches basées sur les MFS sont plus efficaces que l'approche de référence qui exécute des requêtes relaxées sans identifier d'abord les raisons de l'échec de la requête utilisateur.

Les travaux présentés dans cette section ouvrent de nombreuses perspectives de recherche. Nous décrivons quelques pistes qui nous semblent prometteuses.

Proposition d'une approche unifiée pour le problème des réponses vides. Comme nous l'avons indiqué dans la section 4.2, de nombreuses approches complémentaires ont été proposées pour le problème des réponses vides. Ces approches traitent différents aspects du même problème : incomplétude des sources de données, difficulté pour un utilisateur d'exprimer une requête reflétant son intention, manque d'explication sur le résultat retourné, etc. Cependant, il n'existe pas d'approche unifiée pour le traitement de ce problème. Notre idée est de proposer une approche qui identifie le ou les approches coopératives à utiliser en fonction du comportement utilisateur ou de la source de données interrogée. Selon que l'utilisateur souhaite être plus ou moins sollicité, cette approche pourrait proposer des méthodes plus ou moins automatiques. L'idée serait donc de s'appuyer sur la connaissance de l'utilisateur via ses préférences ou son historique de requêtes afin de proposer la ou les approches les plus adaptées pour arriver au résultat souhaité.

Considération d'autres problèmes liés à des résultats insatisfaisants. Les travaux présentés dans ce chapitre visent à répondre au problème des réponses vides. Cependant, d'autres résultats qu'une réponse vide peuvent être considérés comme insatisfaisants par l'utilisateur, menant à d'autres problèmes tels que les suivants.

- *Le problème des réponses insuffisantes*. L'utilisateur n'obtient qu'un nombre faible de résultats (inférieur à un seuil k défini par l'utilisateur) par rapport à ses attentes.
- *Le problème des réponses pléthoriques*. L'utilisateur obtient un nombre trop important de résultats (supérieur à un seuil k défini par l'utilisateur) qui l'empêche d'extraire l'information pertinente de cette masse de résultats.
- *Le problème de l'absence d'un résultat*. L'utilisateur s'attendait à obtenir un résultat qui n'apparaît pas dans la réponse à sa requête.

Pour répondre à ces problèmes, peu de travaux se sont intéressés, à notre connaissance, à identifier les causes d'échec de la requête utilisateur. Aussi, une perspective de nos travaux consiste à étudier comment adapter les notions de MFS et XSS à ces problèmes puis à proposer des approches les calculant en un temps raisonnable. Le principal challenge est que dans ces problèmes, une requête qui retourne un résultat satisfaisant peut en contenir une dont le retour est insatisfaisant. Aussi nos approches développées pour le problème des réponses vides ne peuvent pas être utilisées directement pour répondre à ces nouveaux problèmes.

Passage à l'échelle. Les expérimentations que nous avons menées sur les approches proposées dans ce chapitre ont montré que celles-ci ont des temps d'exécution raisonnables lorsqu'elles sont utilisées sur des ontologies contenant des millions de triplets RDF et pour des requêtes allant jusqu'à 15 patrons de triplet. Dans le contexte des données massives, des tailles de données et de requêtes plus importantes doivent être considérées. Pour répondre à ce besoin de passage à l'échelle, il serait nécessaire d'utiliser des solutions de stockage distribuées reposant sur plusieurs nœuds. Cela implique d'étudier comment les approches que nous avons proposées pourraient être parallélisées pour minimiser la communication entre les différents nœuds.

Par ailleurs, puisque nos approches exécutent de nombreuses requêtes similaires (des sous-requêtes d'une même requête), nous pensons que nos approches pourraient être optimisées en utilisant des techniques d'*optimisation multi-requêtes* (*multi-query optimization*) [166]. Le principe de cette technique est d'éviter d'évaluer une partie d'une requête qui a déjà été exécutée précédemment. La principale difficulté pour utiliser cette technique dans nos approches est d'identifier, a priori, l'ensemble des requêtes que celles-ci vont devoir exécuter. Nous pensons qu'il pourrait être judicieux pour cela de s'appuyer sur des techniques d'estimation du nombre de résultats d'une requête [116].

Chapitre 5

Conclusion et perspectives

5.1 Bilan général

Les travaux résumés dans ce mémoire donnent une vue globale de nos activités de recherche menées au sein de l'équipe IDD du laboratoire LIAS de l'ISAE-ENSMA. Ils ont débuté lors de ma thèse de doctorat avec la conception de la *Bases de Données Sémantiques (BDS)* OntoDB/OntoQL. Ils se sont poursuivis par le développement d'axes de recherche autour des BDS : (i) l'extension et la généralisation de ces *Bases de Données (BD)*, (ii) l'*ingénierie des BDS*, c'est-à-dire l'étude de leur processus de conception ou sélection, et (iii) le développement de techniques *coopératives* pour l'exploitation des BDS. Nos travaux sont ainsi à la frontière entre les domaines des BD et des ontologies. Le développement de nos trois axes de recherche nous a également conduits à explorer d'autres thématiques telles que l'*Ingénierie Dirigée par les Modèles* ou l'*Ingénierie des Besoins*.

La motivation initiale de nos travaux, qui est restée le fil directeur de nos activités de recherche, est que les ontologies sont utilisées dans différents domaines pour des problématiques variées. En conséquence, une diversité de BDS est nécessaire et doit être prise en compte. Cette motivation est issue des domaines d'application de l'ISAE-ENSMA, tels que l'aéronautique, le transport, la mécanique ou l'énergie pour lesquels des ontologies standardisées ont été développées dont les caractéristiques diffèrent de celles proposées dans le contexte du Web sémantique. Nous tirons à présent un bilan de chacun des axes de recherche développés.

5.1.1 Extension et généralisation des Bases de Données Sémantiques

Cet axe de recherche a débuté avec le développement de la BDS OntoDB/OntoQL. Notre objectif était différent de celui des autres BDS, développées dans le contexte du Web sémantique, qui visaient à stocker un maximum de triplets RDF et à répondre à des requêtes SPARQL en un temps minimum. En effet, nous souhaitions proposer une solution alternative lorsque toute la flexibilité du langage RDF n'est pas nécessaire et que les hypothèses et l'expressivité des modèles d'ontologies basés sur RDF (RDFS, OWL) ne sont pas adaptées aux besoins applicatifs. C'était le cas dans nos domaines cibles de l'ingénierie. La poursuite de cet objectif nous a conduits au développement d'une BDS et d'un langage d'exploitation opérationnels dont l'intérêt a été montré dans nos domaines d'applications cibles via des

projets académiques (notamment les projets ANR EWok-Hub et DAFOE 4APP) et industriels (avec l'Institut Français du pétrole, la Compagnie Française de Câblage, etc.). La plateforme OntoDB/OntoQL présente deux principales originalités, décrites dans ce qui suit, sur lesquelles nous nous sommes appuyés pour développer un axe de recherche sur l'extension et la généralisation des BDS.

L'explicitation de la structure des données sémantiques via le modèle de stockage horizontal

Pour supporter la flexibilité du langage RDF, les BDS conçues pour le Web sémantique s'appuient sur des modèles de stockage de type binaire ou vertical qui, d'une part, n'explicite pas la structure des données sémantiques et, d'autre part, sont peu efficaces lorsque les données présentent une structure régulière. La BDS OntoDB/OntoQL propose, comme alternative, le modèle de stockage horizontal qui consiste à garder une représentation proche du modèle relationnel en groupant, dans une table, les propriétés qui sont définies ensemble. Pour cela, elle s'appuie sur les hypothèses de typage du modèle PLIB qui sont plus restrictives que celles de RDF (par exemple, la multi-instanciation n'est pas autorisée). En contrepartie, le modèle de stockage horizontal offre de meilleures performances pour les requêtes spécifiant la classe des instances recherchées et utilisant plusieurs propriétés. De plus, il explicite la structure des données sémantiques afin de faciliter l'expression des requêtes utilisateur.

L'extensibilité de la BDS via l'architecture quatre parties

La seconde originalité de la BDS OntoDB/OntoQL est l'architecture utilisée. Celle-ci est inspirée de l'architecture de métamodélisation du MOF [14] et de celle des *Bases de Données Relationnelles (BDR)* qui inclut une partie métabase stockant la description des tables. En suivant ces approches, nous avons proposé d'ajouter une partie *métaschéma* à l'architecture des BDS. Elle est composée de tables stockant le modèle d'ontologies utilisé par la BDS. Elle joue donc, pour les ontologies, le même rôle que celui joué par la métabase pour les données.

Initialement, cette partie avait été définie dans OntoDB pour stocker le modèle d'ontologies PLIB et gérer ses évolutions. Nous avons généralisé cette approche lors du développement du langage OntoQL. Ainsi, celui-ci se base sur un modèle d'ontologies noyau, stocké dans le métaschéma, qui contient les principaux constructeurs des modèles usuels. Il autorise l'extension de ce modèle noyau via des instructions du langage. Cette capacité permet non seulement des évolutions structurelles du modèle d'ontologies utilisé (par exemple, ajouter une remarque pour décrire les classes des ontologies), mais aussi de créer d'autres modèles de données au sein de la BDS dont les instances pourront être liées aux ontologies stockées. Elle autorise ainsi la persistance dans un même environnement de données définies selon différents modèles et des ontologies qui en décrivent le sens.

Extensions de la BDS OntoDB/OntoQL pour différentes problématiques

Mon équipe de recherche a pu s'appuyer sur les capacités d'extension de la plateforme OntoDB/OntoQL et sur son implémentation opérationnelle lors de dix thèses de doctorat (Dilek Tapucu, Laura Silveira Mastella, Nabil Belaïd, Ilyès Boukhari, Chédlia Chakroun, Chimène Fankam, Bery Mbaiossoum, Kevin Royer, Henri Valéry Teguiak et Okba Barkat) auxquelles j'ai participé (publications communes pour la plupart). Je me suis particulièrement investi dans les thèses de Dilek Tapucu, Laura Silveira Mastella, Nabil Belaïd et Kevin Royer, ce qui m'a permis de voir l'intérêt des capacités d'extension de cette plateforme mais aussi leurs limites.

Nous avons présenté succinctement dans ce mémoire les travaux faits dans le cadre des thèses de Laura Silveira Mastella et Dilek Tapucu. Ils portent respectivement sur l'annotation de modèles métiers

issus de l'ingénierie par des ontologies et la prise en compte de préférences utilisateur dans les BDS. Dans ces travaux, nous avons suivi la même démarche :

1. définir un modèle des données devant être liées à des ontologies (par exemple, un modèle de préférences utilisateur) ;
2. concevoir une structure permettant de lier ce modèle avec le modèle d'ontologies de la BDS (par exemple, une structure qui lie une préférence utilisateur à une classe ou propriété d'une ontologie) ;
3. persister ce modèle et cette structure de données dans la BDS OntoDB et utiliser des instructions OntoQL pour les exploiter (par exemple, faire une requête avec préférences utilisateur).

Cependant, ces extensions de la plateforme n'étant que structurelles, nous avons dû modifier la langage OntoQL pour coder leur sémantique comportementale. Par exemple, nous avons étendu la syntaxe et la sémantique d'OntoQL pour pouvoir prendre en compte des préférences utilisateur dans ses requêtes et les interpréter afin de retourner les résultats adéquats.

Généralisation de la BDS OntoDB/OntoQL

Les extensions de la plateforme OntoDB/OntoQL réalisées nous ont ainsi fait prendre conscience de la nécessité de pouvoir définir la sémantique comportementale d'une extension du métaschéma. Nous avons aussi réalisé que cette plateforme pourrait avoir un usage plus général que la persistance d'ontologies. En conséquence, nous avons proposé une généralisation de la BDS OntoDB/OntoQL dans le cadre de la thèse de Youness Bazhar que j'ai co-encadrée.

Ayant pour objectif de généraliser la BDS OntoDB/OntoQL pour la persistance de différents types de données et modèles, nous avons comparé nos objectifs aux travaux faits dans le domaine de l'*Ingénierie Dirigée par les Modèles*. La plupart des systèmes de métamodélisation classiques chargent l'ensemble des données manipulées en mémoire centrale pour pouvoir ensuite les traiter. Peu de propositions d'environnements complets de BD ont été faites pour la persistance et manipulation de différents types de données et modèles. Ces solutions sont qualifiées de *PMMS* (*Persistent MetaModeling Systems*).

Nous avons donc proposé le PMMS BeMoRe comme une surcouche à la plateforme OntoDB/OntoQL. Par rapport aux autres PMMS, sa principale originalité est de permettre d'introduire des opérations utilisateur *à la volée* qui peuvent être définies aux niveaux modèle ou métamodèle et être implantées dans différents environnements de programmation. Son intérêt a été montré dans plusieurs cas d'usage tels que la conception d'une BDR en 3ème forme normale à partir d'une ontologie ou l'analyse des systèmes temps réel.

5.1.2 Ingénierie des Bases de Données Sémantiques

Au cours du développement de notre premier axe de recherche, nous avons constaté que les BDS peuvent répondre à différents objectifs et, qu'en conséquence, elles présentent une diversité sur différents aspects tels que le modèle d'ontologies, l'architecture ou le modèle de stockage utilisés. Lorsqu'une application basée sur des ontologies est construite et qu'elle nécessite la persistance de celles-ci, cette diversité rend la tâche d'un concepteur difficile puisqu'il doit choisir ou construire une BDS qui réponde au mieux à ses objectifs applicatifs. Nous nous sommes intéressés à ce processus de conception des BDS.

Dans les *Systèmes de Gestion de Données (SGD)* traditionnels, le processus de conception est com-

posé d'étapes bien identifiées : phases de recueil des besoins, conceptuelle, logique et physique. Historiquement, chaque nouveau type de SGD a eu un impact sur ce processus de manière *verticale*, par l'ajout de nouvelles phases, ou de manière *horizontale*, par la proposition de nouveaux outils ou formalismes pour une phase donnée. Nous nous sommes donc intéressés à l'impact des BDS sur ce processus. Cela nous a conduits à développer un axe de recherche dont l'objectif est de prendre en compte les spécificités des BDS, et en particulier leur diversité, dans le processus de conception habituellement suivi pour les SGD traditionnels. Cet axe a été mené avec Ladjel Bellatreche au cours de plusieurs thèses de doctorat co-encadrées.

Évolution verticale : l'intégration de besoins hétérogènes

Notre premier constat est que les ontologies sont souvent utilisées comme modèles de référence par différents partenaires pour construire une application. C'est en particulier le cas dans les domaines de l'ingénierie qui comprennent de nombreuses *entreprises étendues* ainsi que des ontologies standardisées. Chaque partenaire pouvant exprimer ses besoins applicatifs en utilisant un vocabulaire et formalisme particuliers, nous avons proposé l'ajout d'une *phase d'intégration de besoins hétérogènes* au processus de conception des BDS.

Nous avons confronté nos objectifs aux travaux menés dans le domaine de l'*Ingénierie des Besoins*. Les travaux qui se sont intéressés à l'hétérogénéité des besoins visent principalement à combiner plusieurs formalismes et à analyser les besoins produits. Peu de liens sont établis avec le processus de conception des SGD sous-jacents. Aussi, nous avons proposé, lors de la thèse d'Ilyès Boukhari, une approche d'intégration de besoins hétérogènes. Son originalité est d'une part, d'unifier les vocabulaires et formalismes utilisés pour la définition des besoins et, d'autre part, d'être liée à la phase physique de la conception d'une BDS en permettant la sélection de structures d'optimisation à partir des besoins intégrés.

Évolution horizontale : démarche de conception d'une BDS comme une Ligne de Produits

Notre second constat est que la conception des BDS repose sur des choix qui doivent être réalisés au cours des différentes phases. Cela est dû à la diversité des BDS. Les méthodes et outils classiques de conception des SGD, tels que les outils CASE, n'explicitent pas les différents choix possibles et les impacts qu'ils ont les uns sur les autres. C'est par contre le cas de ceux des *Lignes De Produits (LDP)* utilisées dans le contexte de l'*Ingénierie Logicielle*. Aussi, nous avons proposé lors de la thèse de Selma Bouarar, une démarche de conception d'une BDS s'appuyant sur ces méthodes et outils.

Cette démarche repose sur des *modèles de features* qui explicitent les différentes *variantes* de conception possibles à chaque phase ainsi que les liens entre ces différentes variantes. En allant plus loin, que ce qui est proposé dans le contexte des LDP, notre démarche inclut la possibilité d'identifier la *configuration* de BDS qui répond le mieux à des objectifs non fonctionnels. Cela a été concrétisé par l'outil OntoDBench qui identifie la conception logique d'une BDS qui fournit le meilleur temps de réponse pour un ensemble de requêtes exécutées sur un jeu de données spécifié.

Évolution horizontale : sélection de vues matérialisées prenant en compte la diversité des BDS

Notre troisième constat est qu'il est difficile d'implémenter l'ensemble des features de notre démarche de conception basée sur les LDP puisque ceux-ci doivent prendre en compte la diversité des BDS. Pour étudier cela, nous nous sommes intéressés à la phase physique et plus particulièrement à la phase de sélection de vues matérialisées pour les BDS. La plupart des approches proposées dans la litté-

rature pour ce problème sont définies pour une BDS implémentée par un table de triplets. De plus, celles pouvant s'appliquer à différentes BDS ne prennent pas en compte précisément les impacts de la diversité des BDS. Aussi, nous avons proposé dans la thèse de Bery Mbaiossoum deux nouvelles approches de sélection de vues matérialisées. Celles-ci sont basées sur un modèle de coût *générique* dans le sens où il peut s'appliquer à différents types de BDS tout en prenant en compte leurs spécificités. Nous avons montré expérimentalement que, malgré le fait qu'elles puissent s'appliquer à différents types de BDS, nos approches donnent des résultats proches de celles développées pour une BDS spécifique.

5.1.3 Techniques coopératives pour les Bases de Données Sémantiques

Dans les deux premiers axes de recherche développés, nos propositions ont pour objectifs d'étendre les fonctionnalités des BDS et de faciliter leur conception. Dans notre troisième axe de recherche, nous nous sommes intéressés à l'exploitation de ces BDS d'un point de vue utilisateur. Cet axe a débuté avec l'arrivée d'Allel Hadjali au laboratoire LIAS dont une thématique est le développement de *techniques d'interrogations coopératives*. Conçues jusque-là dans le contexte des BDR, nous avons d'abord constaté le besoin de ces techniques pour les BDS. En effet, comme nous avions pu le voir dans nos autres travaux, les BDS présentent des caractéristiques particulières telles que le fait que la structure des données ne soient pas explicitées dans la plupart d'entre elles. En conséquence, une requête utilisateur peut retourner une réponse vide, alors qu'un résultat était attendu, simplement parce qu'une propriété qu'elle utilise ne décrit pas les instances recherchées.

Les techniques coopératives visent à aider l'utilisateur à résoudre ce problème, qualifié de *problème des réponses vides*. Nous avons donc étudié les manières d'adapter et d'étendre ces techniques pour les BDS au cours de plusieurs thèses que j'ai co-encadrées avec Allel Hadjali. Pour que nos travaux puissent avoir le plus d'impact possible, nous les avons développés avec les technologies du Web sémantique qui sont celles majoritairement utilisées dans le contexte académique.

Approches de calcul des MFS et XSS pour les BDS

Nous avons commencé par étudier le problème du calcul des *MFS (Minimal Failing Subquery)* et *XSS (MaXimal Succeeding Subquery)*. Ces sous-requêtes de la requête utilisateur sont intéressantes car elles fournissent, d'une part, les causes d'échec de celle-ci et, d'autres part, des requêtes alternatives, comportant un nombre maximal d'éléments de la requête utilisateur, qui réussiront (c'est-à-dire, qui retourneront un résultat). L'énumération des MFS et XSS étant un problème $\mathcal{NP}$-Difficile, notre objectif était de développer des approches qui puissent être appliquées pour des requêtes de tailles raisonnables. Pour cela, nous avons adapté et étendu les deux approches faisant office de référence dans le contexte relationnel : l'approche de Godfrey [124] basée sur un parcours du treillis des sous-requêtes de la requête utilisateur et celle de Jannach [125] basée sur le calcul d'une matrice.

Le résultat de ces travaux, développés lors de la thèse de Géraud Fokou, sont les approches *LBA (Lattice-Based Approach)* et *MBA (Matrix-Based Approach)* qui prennent en compte les spécificités des langages RDF/SPARQL par rapport au contexte relationnel (par exemple, la différence de sémantique sur le traitement des valeurs *null* entre SPARQL et SQL). Alors que l'approche matricielle est celle considérée comme la plus efficace dans le contexte relationnel, nous avons montré qu'elle n'est applicable dans le contexte des BDS que sous des hypothèses restrictives. Aussi, nous considérons que l'approche LBA est celle qui doit être majoritairement utilisée en pratique. Nous avons montré expérimentalement qu'elle

calcule les MFS et XSS d'une requête RDF en un temps raisonnable tant que sa taille ne dépasse pas 15 patrons de triplet.

Prise en compte de l'incertitude des faits contenus dans les ontologies

Nous nous sommes ensuite intéressés, dans le cadre de la thèse d'Ibrahim Dellal, à prendre en compte le fait que de nombreuses ontologies réelles associent un *degré de confiance* aux faits représentés. Cette caractéristique permet à l'utilisateur d'exprimer des requêtes avec un seuil de confiance minimum voulu sur les résultats obtenus. Nous avons d'abord montré que l'algorithme LBA ne peut être utilisé dans ce contexte que si la fonction d'agrégation utilisée pour affecter un degré de confiance à un résultat d'une requête est *monotone décroissante* par rapport à la relation d'inclusion ensembliste. Puis, nous avons considéré le fait que l'utilisateur pourrait être enclin à diminuer le degré de confiance minimum voulu s'il n'obtient pas de résultat à sa requête initiale. Aussi, nous avons proposé des approches de calcul des MFS et XSS pour différents seuils α. Plutôt que d'exécuter l'algorithme LBA pour chacun de ces seuils, nos approches essaient de déduire des MFS et XSS à partir de celles calculées pour un autre seuil. Nous avons montré théoriquement et expérimentalement l'intérêt de ces approches.

Approches automatiques de relaxation de requêtes RDF basées sur les causes d'échec

Même si nos approches calculent les MFS et XSS d'une requête RDF, qui sont des informations utiles pour éviter le problème des réponses vides, leur exploitation par l'utilisateur final peut s'avérer délicate. Aussi, nous nous sommes intéressés aux approches automatiques de relaxation de requêtes RDF. Ces approches se basent sur les hiérarchies de classes et de propriétés du schéma des ontologies afin de généraliser la requête utilisateur. L'approche de référence [17] consistait à générer l'ensemble des requêtes relaxées possibles, les trier selon leur similarité avec la requête utilisateur (c'est-à-dire, de la plus à la moins similaire) et les exécuter dans cet ordre jusqu'à trouver suffisamment de résultats par rapport aux exigences de l'utilisateur. Le nombre de requêtes relaxées possibles étant exponentiel par rapport à la taille de la requête, cette approche pose rapidement des problèmes de performance dès que la taille de la requête dépasse quelques patrons de triplet.

Nous avons identifié que ce problème vient du fait que cette approche réalise le processus de relaxation *à l'aveugle*, c'est-à-dire sans savoir pourquoi la requête utilisateur échoue. Aussi, nous avons proposé d'utiliser les causes d'échec (MFS) pour élaguer l'espace de recherche formé par l'ensemble des requêtes relaxées possibles. Nous avons ainsi développé plusieurs approches qui utilisent plus ou moins d'information sur les causes d'échec de la requête initiale et de ses requêtes relaxées. Nous avons montré expérimentalement l'intérêt de ces approches par rapport à celle de référence. Nous avons aussi montré que, globalement, l'approche donnant les meilleures performances consiste à trouver un équilibre entre le coût de calcul des MFS et le gain que celles-ci permettent d'obtenir en évitant d'exécuter des requêtes relaxées.

5.2 Perspectives

Dans la conclusion des chapitres décrivant nos trois axes de recherche, nous avons évoqué des pistes potentiellement prometteuses qu'ils ouvrent. Dans cette section, nous commençons par décrire nos perspectives à court terme concrétisées par des travaux en cours. Nous évoquerons ensuite des perspectives à moyen ou long terme.

5.2.1 Travaux en cours

Framework de techniques coopératives pour les BDS

Je co-encadre actuellement la thèse de Louise Parkin avec Allel Hadjali et Brice Chardin qui vise à poursuivre nos travaux sur le développement de techniques coopératives pour les BDS. Dans le cadre de ces travaux nous souhaitons étudier s'il est possible de créer un *framework* de techniques coopératives pour les BDS. Il serait composé de deux parties : d'approches de calcul des MFS et XSS pour divers problèmes de réponses insatisfaisantes et de méthodes pour exploiter ces MFS et XSS.

Approches extensibles de calcul des MFS et XSS. Le framework proposerait ainsi des algorithmes génériques de calcul des MFS et XSS sur lesquels l'utilisateur du framework pourrait se baser pour les calculer pour différents problèmes tels que les réponses pléthoriques ou l'apparition d'un résultat inattendu. Pour cela, il devrait spécifier différentes fonctions prévues par le framework permettant, entre autres, de savoir si une requête échoue ou non pour le problème considéré.

Nos premiers résultats sur cette problématique montrent la nécessité de définir une notion plus générale que celle de MFS qui ne correspond pas forcément à une cause d'échec pour certains problèmes tels que celui des réponses pléthoriques. Nous avons aussi constaté que le framework devra fournir deux types d'algorithmes selon que la propriété de monotonie (c'est-à-dire, si le fait qu'une requête échoue implique forcément que ses super-requêtes échouent) est vérifiée ou non pour le problème considéré. Enfin, le framework devra permettre à l'utilisateur d'introduire des propriétés sur les requêtes du treillis afin d'élaguer cet espace de recherche.

Approches permettant d'utiliser les MFS et XSS. Le framework proposerait différentes manières de tirer profit des MFS et XSS.

- *Dans l'approche manuelle*, l'idée est de pouvoir expliquer en langage naturel les causes d'échecs correspondant aux MFS en s'appuyant sur les travaux de verbalisation des requêtes SPARQL [167]. Concernant les XSS, puisque ce sont des requêtes alternatives que l'utilisateur peut exécuter avec une garantie de succès, nous envisageons de définir des méthodes pour les ordonner afin de ne présenter à l'utilisateur que celles qui sont les plus pertinentes.
- *Dans l'approche interactive*, les MFS et XSS seraient utilisées pour aider l'utilisateur à corriger sa requête en évitant, autant que possible, d'obtenir de nouveaux échecs qui seraient frustrants pour l'utilisateur. Pour cela, nous envisageons de nous baser sur les travaux de Mottin *et al.* [160], définis dans le contexte des BDR, qui proposent à l'utilisateur de relâcher des conditions de sa requête. Le principal challenge dans le contexte de SPARQL est qu'un patron de triplet ne correspond pas forcément à une condition au sens usuel et qu'il peut être généralisé de plusieurs façons.
- *Dans l'approche automatique*, les MFS et XSS seraient utilisées pour corriger automatiquement le problème comme nous l'avons fait pour le problème des réponses vides [142]. L'utilisateur du framework devrait alors spécifier les manières de construire des requêtes alternatives selon le problème considéré et fournir un moyen d'ordonner ces requêtes par similarité avec la requête posant problème.

Ces travaux poursuivent un de nos axes de recherche. Dans ceux décrits dans ce qui suit, nous considérons de nouvelles thématiques. Nous y abordons des problématiques scientifiques rencontrées dans un contexte industriel. Nous envisageons de traiter ces problèmes en nous basant sur les grands principes

identifiés dans nos travaux de recherche.

Approche ontologique pour la qualité de données

Je co-encadre actuellement avec Ladjel Bellatreche la thèse de Reda Mekhezzem, co-financée par l'entreprise *Ayaline* [25]. Cette thèse aborde le thème de la qualité des données qui est un élément essentiel dans de nombreux projets industriels. Pour traiter les problèmes de qualité de données, il existe aujourd'hui de nombreuses techniques automatiques ou semi-automatiques intégrées à des outils [168]. Cette variété de possibilités pour traiter ce problème complexifie la tâche de nettoyage de données. En effet, il n'est pas simple d'identifier quelles sont les techniques à choisir et à appliquer pour améliorer la qualité des données utilisées dans un projet particulier. En effet, chaque technique traite un problème de qualité spécifique pour un contexte donné généralement sans garantie sur la complétude et justesse des corrections effectuées.

Un autre problème que nous souhaitons traiter concerne le processus amenant à améliorer la qualité des données. Aujourd'hui aucune solution ne garde une *traçabilité* complète de ce processus. Cette notion, qui est le sujet de nombreux travaux dans l'*Ingénierie Logicielle* [169], est encore peu mise en œuvre dans le contexte de la qualité des données. Cela pose un double problème. D'une part, il est difficile de revenir en arrière dans ce processus lorsqu'une erreur est commise. L'absence de *transparence* dans ce processus ne permet pas non plus de savoir qui a réalisé cette erreur et pourquoi. D'autre part, il n'y a pas de *capitalisation* sur l'expérience acquise lors d'une tentative pour améliorer la qualité des données. Cela nécessite de conserver les actions réalisées pour améliorer la qualité des données mais aussi les résultats obtenus.

Pour traiter ces problèmes, en s'inspirant des approches définies pour garantir la transparence sur le cycle de vie de la donnée [170], nous envisageons de concevoir une ontologie qui représentera et catégorisera les techniques permettant d'améliorer la qualité des données. En complément de cette ontologie, nous devrons définir des approches pour aider l'utilisateur à choisir la technique adaptée à un problème de qualité de données ciblé. Nous envisageons également que cette ontologie puisse servir à sauvegarder toutes manipulations faites sur les données et les résultats obtenus afin de conserver une traçabilité sur le processus suivi. Cela offrirait également la possibilité de le *certifier*. La plateforme OntoDB/OntoQL pourra être utilisée pour assurer la persistance de ces informations.

Établissement d'un réseau anonyme de confiance pour les applications communautaires

Je co-encadre actuellement avec Mickaël Baron et Allel Hadjali la thèse CIFRE de Chayma Sellami réalisée avec l'entreprise *O'Code* [26]. Cette société a développé une technologie permettant d'identifier des objets de manière unique et souhaite la valoriser par le développement d'*applications communautaires*. Celles-ci se basent sur un *réseau d'utilisateurs* qui collaborent pour une activité donnée. Ce type d'application a connu un développement important ces dernières années et est utilisé pour une grande variété d'activités (par exemple, le covoiturage ou l'échange de services). Cependant, la nature ouverte et décentralisée des communautés rend ces applications vulnérables à l'apparition d'utilisateurs malveillants. Une évaluation de la *confiance* que l'on peut accorder à un utilisateur dans ces systèmes est donc essentielle pour la prise de décision. Elle fournit des informations précieuses aux utilisateurs d'une application communautaire puisqu'elle permet de faire la différence entre ceux qui sont fiables et ceux

25. https://www.ayaline.com
26. https://www.ocode.team

qui ne le sont pas. Pour des raisons éthiques, ce travail s'inscrit dans le cadre des réseaux dits *anonymes* où l'anonymat des utilisateurs est complètement respecté.

Cette thèse s'intéresse ainsi aux *modèles de confiance* qui définissent une façon de représenter et d'évaluer la confiance au sein d'un réseau d'utilisateurs anonymes. Dans la littérature, de nombreux modèles ont été proposés [171]. Cependant, ils se basent sur différentes définitions de la notion de confiance et des concepts qui s'y rapportent. De plus, ils utilisent des modèles de calcul variés (par exemple, les réseaux bayésiens ou chaînes de Markov) pour quantifier la notion de confiance au sein d'un réseau. En conséquence, il est difficile d'identifier ou de concevoir un modèle de confiance qui soit adapté à une application communautaire particulière pour laquelle des exigences doivent être respectées (par exemple, l'anonymat). Les états de l'art récents sur ces modèles [171, 172] ne permettent pas de résoudre ce problème. D'une part, même s'ils essaient de donner des définitions générales sur la notion de confiance, ils ne proposent pas de formalisation de ces notions. D'autre part, ils ne comparent les modèles de confiance que sur des aspects statiques liés à des critères techniques tels que les données d'entrée et sortie de ces modèles. Ainsi, ils n'explicitent pas leur intérêt dans le cadre de scénarios concrets qui pourraient arriver dans le contexte des applications communautaires.

Aussi, en suivant les principes des ontologies, nous souhaitons tout d'abord formaliser les notions liées à la confiance et proposer une représentation graphique, basée sur des graphes, pour en faciliter l'utilisation. Puis, nous souhaitons pouvoir comparer le comportement *dynamique* des modèles de confiance en s'appuyant sur des scénarios qui permettent de caractériser leurs comportements à l'aide d'un vocabulaire métier (par exemple, déterminer si un modèle est *tolérant* à une mauvaise action dans l'application ou non). Enfin, nous souhaitons proposer une démarche complète qui permette, d'une part, de définir des exigences sur l'application communautaire en cours de conception et, d'autre part, de déterminer le modèle de confiance, ou une combinaison de ceux-ci, qui les satisfasse autant que possible.

5.2.2 Perspectives à moyen ou long terme

Nous décrivons à présent deux axes de recherche que nous souhaitons développer à plus ou moins long terme.

Traitement coopératif des flux de données sémantiques

Avec l'utilisation croissante de réseaux de capteurs dans de nombreux domaines et en particulier dans ceux de l'ingénierie, de nombreuses données sont produites sous la forme de flux. Par exemple, la détection et prévision des séismes s'appuient sur des réseaux de capteurs tels que les sismomètres ou accéléromètres pour étudier les ondes sismiques. Les données de flux sont reçues en continu et en temps réel. Elles sont ordonnées selon leur ordre d'arrivée et sont utilisées à des fins de surveillance, de déclenchement d'alarmes ou d'aide à la prise de décision.

Des systèmes de gestion de flux de données (*DSMS* pour *Data Stream Management Systems*) et de traitement d'évènements complexes (*CEP* pour *Complex Event Processing*) ont été conçus pour évaluer des requêtes sur les données caractérisées par des *fenêtres temporelles* [173]. Cependant, les DSMS et CEP ne prennent pas en compte l'hétérogénéité des flux de données. Le Web sémantique proposant des langages pour faciliter l'interopérabilité des systèmes, des solutions de traitement de flux de données sémantiques, qualifiées de *RSP*, pour *RDF Stream Processing*, ont été proposées. Elles s'ap-

puient sur le langage RDF qui décompose les flux de données en des triplets (sujet, prédicat, objet) auxquels une estampille temporelle est ajoutée. Ces flux de données sont interrogés avec une extension de SPARQL [174].

De nombreux travaux se sont intéressés aux traitements des flux de données sémantiques, notamment sur les aspects suivants : (i) *échantillonnage* du flux pour le passage à l'échelle [175] (ii) *raisonnement* efficace sur le flux pour déduire de nouvelles informations à partir de celles présentes [176], (iii) *enrichissement sémantique* du flux avec de nouvelles informations issues d'ontologies externes [177] (iv) *fouille de données* sur le flux [178]. Par contre, peu de travaux ont proposé des approches pour aider l'expert qui obtient des résultats insatisfaisants en réponse à ses requêtes posées sur des flux de données sémantiques.

Nous proposons ainsi d'étendre et d'adapter les techniques coopératives, que nous avons développées dans un contexte statique, pour aider l'expert à exploiter les flux de données sémantiques. Nous pensons que cette problématique est complémentaire aux travaux cités précédemment car ces derniers n'abordent pas l'interaction avec l'expert. Pourtant l'exploitation des flux de données est souvent faite dans un contexte où l'expertise humaine est nécessaire. C'est le cas de la détection des séismes où l'expert doit analyser le flux de données ayant fait remonter une alerte pour prendre des décisions telles que l'évacuation d'une zone géographique.

Les caractéristiques propres aux flux de données sémantiques amènent de nouvelles problématiques par rapport à nos précédents travaux.

- L'absence d'une *sémantique opérationnelle* pour les langages de requêtes conçus pour les flux RDF. Actuellement, ces langages s'appuient sur la sémantique déclarative de SPARQL [141] qui n'explicite pas le calcul incrémental de solutions au fur et à mesure que de nouvelles données arrivent dans le flux. Notre idée est de proposer une sémantique basée sur des systèmes de transitions afin d'expliciter ce calcul incrémental. Les techniques coopératives proposées pourraient alors se baser sur cette sémantique pour fournir une assistance à l'expert *au cours* de l'exécution d'une requête.
- Les techniques coopératives existantes sont généralement associées à des algorithmes coûteux dont les temps de réponses peuvent être acceptables dans un contexte statique mais ne pas l'être lorsqu'un flux de données sémantiques doit être analysé en temps réel. Il est donc nécessaire de proposer des heuristiques, éventuellement basées sur le domaine applicatif, pour optimiser ces techniques et ainsi les rendre opérationnelles dans un contexte de flux de données.
- Les techniques coopératives s'appuient souvent sur l'identification des raisons pour lesquelles le résultat n'est pas satisfaisant. Dans le contexte classique, ces raisons sont souvent liées au fait que l'utilisateur a exprimé des contraintes trop fortes dans sa requête. Dans le contexte de flux de données sémantiques, cela peut aussi venir du fait que la fenêtre temporelle est mal choisie ou que le flux de données doit être enrichi par des données sémantiques externes pour que la requête retourne un résultat intéressant. Ces caractéristiques offrent de nouvelles possibilités pour définir la notion de cause d'échec dans le contexte des requêtes exprimées sur un flux.

Visualisation et ontologie

Dans cette perspective, nous nous intéressons au domaine de la préservation du patrimoine, thématique pour laquelle nous avons participé à la mise en place de l'atelier *Semantic Web for Cultural*

Heritage[27] *(SW4CH)* et d'un numéro spécial pour la revue *Semantic Web Journal*[28]. Il sera mené en collaboration avec des collègues des laboratoires *XLIM*[29] et *CESM*[30] *(Centre d'études supérieures de civilisation médiévale)* pour combiner nos compétences sur la caractérisation sémantique et la visualisation 3D d'objets à partir de photographies (c'est-à-dire, le *rendu à base d'images*), afin de répondre à des besoins d'études de monuments du patrimoine tels que le *Palais des ducs d'Aquitaine* [179] ou l'*Hypogée des Dunes* [180].

Les représentations 2D et 3D sont souvent utilisées par les historiens pour décrire, comprendre et analyser des objets ou monuments du patrimoine. Elles sont *annotées* afin d'enregistrer des observations sur l'objet étudié. Ces représentations sont généralement insuffisantes pour les historiens qui s'appuient également sur d'autres sources d'information (par exemple, des ressources documentaires ou des données analytiques de différentes natures) pour mener leur analyse. Ainsi, de nombreuses données *hétérogènes* sont créées et manipulées dans ces activités. Même si de nombreux outils numériques permettent d'aider les historiens dans ces tâches, ils présentent des limitations concernant la gestion des données [181, 182].

- Les annotations sont souvent définies comme de simples mots clés, textes libres ou à l'aide de couples *(attribut, valeur)* ce qui rend difficile leur exploitation.
- L'ensemble des informations utiles à l'analyse d'un objet du patrimoine reste disséminé et difficilement accessible pour les historiens.
- Les annotations sont souvent liées à une unique représentation de l'objet étudié (par exemple, une image) alors qu'elles ont du sens pour d'autres représentations (par exemple, un modèle 3D).

Récemment, plusieurs travaux ont été menés pour répondre à ces limitations [181, 182]. Ils se focalisent sur différents aspects de ces problèmes. Ainsi, les travaux menés en lien avec le projet Aïoli[31] [181, 183] proposent une plateforme collaborative pour l'annotation d'objets du patrimoine. Ils se focalisent sur la proposition d'une approche semi-automatique pour la création des annotations et à leur propagation entre les représentations 2D et 3D. Par contre, la structuration des annotations reste à la charge de l'utilisateur et ne se base sur aucun modèle de référence. Les travaux de Messaoudi *et al.* [182] ont proposé une ontologie de domaine dédiée à l'annotation d'images spatialisées pour le suivi de la conservation du patrimoine culturel bâti. Cette ontologie s'appuie sur le modèle de référence CIDOC-CRM [184] pour faciliter le partage des annotations produites par les historiens. L'intérêt de cette ontologie pour d'autres tâches, telles que faciliter la recherche d'information, n'est cependant pas abordé.

Ces travaux sont dans la lignée de ce que nous souhaitons faire, c'est-à-dire proposer une représentation formelle et partagée des informations liées à l'étude d'objets du patrimoine. Nous souhaitons aller plus loin que ces travaux sur la gestion des données, en répondant aux problématiques suivantes.

- *Construire une ontologie pour l'annotation des objets considérés*. Celle-ci sera dédiée à la description d'objets du patrimoine et contiendra différents types d'information incluant du texte, des images, etc. Notre objectif est que cette ontologie se base autant que possible sur des modèles de références tels que CIDOC-CRM [184] ou le Dublin Core[32] et qu'elle soit liée à d'autres ontolo-

27. https://sw4ch2018.ensma.fr
28. http://www.semantic-web-journal.net/blog/special-issue-semantic-web-cultural-heritage
29. https://www.xlim.fr
30. https://cescm.labo.univ-poitiers.fr
31. http://www.aioli.cloud/
32. http://dublincore.org

gies existantes dans ce domaine [185]. Pour aider l'historien dans le processus d'annotation, nous envisageons de lui proposer des annotations en nous basant sur les faits contenus dans d'autres ontologies généralistes telles que DBPedia [22] ou YAGO [21].

– *Établir des liens entre les représentations 2D/3D et les annotations*. Dans les approches de l'état de l'art, les annotations sont généralement utilisées pour aider l'historien à trouver des informations sur l'objet étudié. Cependant, nous pensons qu'elles peuvent aussi servir à améliorer ou vérifier ses représentations 2D/3D (par exemple, si elles définissent les dimensions de l'objet étudié, il est possible de vérifier ou ajuster les représentations 2D/3D pour qu'elles soient conformes à ces données). Et inversement, les représentations 2D/3D pourraient servir à compléter ou vérifier les annotations produites par les historiens.

– *Aider l'historien à exploiter l'ontologie et les représentations 2D/3D fournies*. Celui-ci peut, en effet, vouloir exprimer des requêtes sur l'ontologie ou les représentations 2D/3D. Il est donc nécessaire de fournir un langage d'interrogation *unifié* permettant d'interroger ces différents types de données. Pour qu'il puisse être utilisé par des historiens, il devra probablement être basé sur le langage naturel. Pour cela, nous envisageons de nous baser sur les approches développées dans les domaines de la *recherche sémantique (Semantic Search)* [186] et des *requêtes visuelles (Visual Question Answering)* [187]. Un challenge sera également de transposer nos méthodes d'interrogation coopératives à ce type d'approches.

Pour ces deux dernières problématiques, nous nous intéresserons aux approches développées pour la reconnaissance de formes dans des images. En effet, des ontologies d'images annotées telles que *ImageNet* [188] ou *VisualGenome* [189], qui sont qualifiées d'*ontologies visuelles*, ont été conçues pour cette problématique. Elles sont utilisées pour mettre en œuvre des techniques d'apprentissage afin de répondre à des questions visuelles sur les images. Même si ces approches peuvent être difficilement appliquées au domaine du patrimoine pour lequel il n'existe pas de collections d'images conséquentes et parfaitement annotées, elles proposent des techniques pour combiner des ontologies visuelles avec des connaissances externes et interroger cet ensemble d'information [187]. Elles peuvent donc être utiles à l'atteinte de nos objectifs.

Bibliographie

[1] Pierra, G.: Context Representation in Domain Ontologies and its Use for Semantic Integration of Data. Journal Of Data Semantics (JODS) **X** (2007) 34–43

[2] Dehainsala, H.: Explicitation de la sémantique dans les bases de données : le modèle OntoDB de bases de données à base ontologique. PhD thesis, LISI/ENSMA et Université de Poitiers (2007)

[3] Gandon, F.: A Survey of the First 20 Years of Research on Semantic Web and Linked Data. Ingénierie des Systèmes d Information **23**(3-4) (2018) 11–38

[4] Hull, R., King, R.: Semantic Database Modeling: Survey, Applications, and Research Issues. ACM Computing Surveys **19**(3) (1987) 201–260

[5] Abrial, J.R.: The B-book: Assigning Programs to Meanings. Cambridge University Press (1996)

[6] Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition **5**(2) (1993) 199–220

[7] Jean, S.: OntoQL, un langage d'exploitation des bases de données à base ontologique. PhD thesis, LISI/ENSMA et Université de Poitiers (2007)

[8] Cullot, N., Parent, C., Spaccapietra, S., Vangenot, C.: Ontologies: A Contribution to the DL/DB Debate. In: Proceedings of the 1st International Workshop on Semantic Web and Databases (SWDB'03). (2003) 109–129

[9] Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human-Computer Studies (IJHCS) **43**(5-6) (1995) 907–928

[10] Bellatreche, L., Pierra, G., Xuan, D.N., Dehainsala, H., Ameur, Y.A.: An a Priori Approach for Automatic Integration of Heterogeneous and Autonomous Databases. In: Proceedings of the 15th International Conference on Database and Expert Systems Applications (DEXA'04). (2004) 475–485

[11] Brickley, D., Guha, R.V.: RDF Schema 1.1. World Wide Web Consortium. (2014) http://www.w3.org/TR/rdf-schema.

[12] Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. World Wide Web Consortium. (2004) http://www.w3.org/TR/owl-ref.

[13] De Bruijn, J., Lara, R., Polleres, A., Fensel, D.: OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In: Proceedings of the 14th international conference on World Wide Web (WWW'05). (2005) 623–632

[14] Object Management Group: Meta Object Facility (MOF) https://www.omg.org/spec/MOF/. Technical report (2016)

[15] Pham, M., Passing, L., Erling, O., Boncz, P.A.: Deriving an Emergent Relational Schema from RDF Data. In: Proceedings of the 24th International Conference on World Wide Web (WWW'15). (2015) 864–874

[16] Saleem, M., Ali, M.I., Hogan, A., Mehmood, Q., Ngomo, A.N.: LSQ: The Linked SPARQL Queries Dataset. In: Proceedings of the 14th International Semantic

Web Conference (ISWC'15). (2015) 261–269

[17] Huang, H., Liu, C., Zhou, X.: Approximating Query Answering on RDF Databases. Journal of the World Wide Web: Internet and Web Information Systems (WWW) **15**(1) (2012) 89–114

[18] Frosini, R., Calì, A., Poulovassilis, A., Wood, P.T.: Flexible Query Processing for SPARQL. Semantic Web **8**(4) (2017) 533–563

[19] Huang, H., Liu, C., Zhou, X.: Computing Relaxed Answers on RDF Databases. In: Proceedings of the 9th International Conference on Web Information Systems Engineering (WISE'08). (2008) 163–175

[20] Hurtado, C.A., Poulovassilis, A., Wood, P.T.: Ranking Approximate Answers to Semantic Web Queries. In: Proceeding of the 6th Extended Semantic Web Conference (ESWC'09). (2009) 263–277

[21] Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Artificial Intelligence **194** (2013) 28–61

[22] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web **6**(2) (2015) 167–195

[23] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14). (2014) 601–610

[24] Deshpande, O., Lamba, D.S., Tourn, M., Das, S., Subramaniam, S., Rajaraman, A., Harinarayan, V., Doan, A.: Building, Maintaining, and Using Knowledge Bases: A Report from the Trenches. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD'13). (2013) 1209–1220

[25] Harris, S., Gibbins, N.: 3store: Efficient Bulk RDF Storage. In: Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03). (2003) 1–15

[26] Wilkinson, K.: Jena Property Table Implementation. In: Proceedings of the 2nd International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS'06). (2006) 35–46

[27] Özsu, M.T.: A Survey of RDF Data Management Systems. Frontiers of Computer Science **10**(3) (2016) 418–432

[28] Abdelaziz, I., Harbi, R., Khayyat, Z., Kalnis, P.: A Survey and Experimental Comparison of Distributed SPARQL Engines for Very Large RDF Data. VLDB Journal **10**(13) (2017) 2049–2060

[29] Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J., Özsu, M.T.: The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing: Extended Survey. VLDB Journal **29**(2-3) (2020) 595–618

[30] Bigerl, A., Conrads, F., Behning, C., Sherif, M.A., Saleem, M., Ngomo, A.N.: Tentris - A Tensor-Based Triple Store. In: Proceedings of the 19th International Semantic Web Conference (ISWC'20). (2020) 56–73

[31] Khelil, A., Mesmoudi, A., Galicia, J., Bellatreche, L., Hacid, M.S., Coquery, E.: Combining Graph Exploration and Fragmentation for Scalable RDF Query Processing. Information Systems Frontiers (2020) 1–19

[32] Levandoski, J.J., Mokbel, M.F.: RDF Data-Centric Storage. In: Proceedings of the 7th IEEE International Conference on Web Services (ICWS'09). (2009) 911–918

[33] Kopena, J., Regli, W.: DAMLJessKB: A Tool for Reasoning with the Semantic Web. IEEE Intelligent Systems **18**(3) (2003) 74–77

[34] Corby, O., Dieng-Kuntz, R., Faron-Zucker, C.: Querying the Semantic Web with Corese Search Engine. In: Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI'04). (2004) 705–709

[35] Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: SW-Store: A Vertically Partitioned DBMS for Semantic Web Data Management. VLDB Journal **18**(2) (2009) 385–406

[36] Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Proceedings of the 1st Internation Semantic Web Conference (ISWC'02). (2002) 54–68

[37] Park, M.J., Lee, J.H., Lee, C.H., Lin, J., Serres, O., Chung, C.W.: An Efficient and Scalable Management of Ontology. In: Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07). Volume 4443. (2007) 975–980

[38] Zou, L., Mo, J., Chen, L., Özsu, M.T., Zhao, D.: gStore: Answering SPARQL Queries via Subgraph Matching. VLDB Journal **4**(8) (2011) 482–493

[39] Neumann, T., Weikum, G.: RDF-3X: A RISC-style Engine for RDF. VLDB Journal **1**(1) (2008) 647–659

[40] Weiss, C., Karras, P., Bernstein, A.: Hexastore: Sextuple Indexing for Semantic Web Data Management. VLDB Journal **1**(1) (2008) 1008–1019

[41] Yuan, P., Liu, P., Wu, B., Jin, H., Zhang, W., Liu, L.: TripleBit: A Fast and Compact System for Large Scale RDF Data. VLDB Journal **6**(7) (2013) 517–528

[42] Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Web Semantics **3**(2-3) (2005) 158–182

[43] Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: Diversified Stress Testing of RDF Data Management Systems. In: Proceedings of the 13th International Semantic Web Conference (ISWC'14). (2014) 197–212

[44] Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD'11). (2011) 145–156

[45] Aït-Ameur, Y., Baron, M., Belaid, N., Jean, S., Mastella, L.S.: Ontology Integration and Management within Data Intensive Engineering Systems. In: Shared Earth Modeling: Knowledge Driven Solutions for Building and Managing Subsurface 3D Geological models, Chapter 13. (2013) 281–305

[46] Mastella, L.S., Aït-Ameur, Y., Jean, S., Perrin, M., Rainaud, J.F.: Semantic Exploitation of Persistent Metadata in Engineering Models: Application to Geological Models. In: Proceedings of the 3rd International Conference on Research Challenges in Information Science (RCIS'09). (2009) 129–138

[47] Bergamaschi, S., Beneventano, D., Guerra, F., Orsini, M.: Data Integration. In: Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges. (2011) 441–476

[48] Ludäscher, B., Lin, K., Bowers, S., Jaeger-Frank, E., Brodaric, B., Baru, C.: Managing Scientific Data: From Data Integration

to Scientific Workflows. In: GeoInformatics, Data to Knowledge. (2006) 109–129

[49] Heflin, J., Hendler, J.: Searching the Web with SHOE. In: Proceedings of the AAAI-2000 Workshop on Artificial Intelligence for Web Search (WS'00). (2000) 35–40

[50] Farell, J., Lausen, H.: Semantic Annotations for WSDL - W3C Working Draft http://www.w3.org/2002/ws/sawsdl/ (2007)

[51] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. Journal of Web Semantics **4**(1) (2006) 14–28

[52] Chomicki, J.: Preference Formulas in Relations Queries. ACM Transactions on Database Systems **28**(4) (2003) 1–39

[53] Bellatreche, L., Giacometti, A., Marcel, P., Mouloudi, H., Laurent, D.: A Personalization Framework for OLAP Queries. In: Proceedings of the 8th ACM International Workshop on Data warehousing and OLAP (DOLAP'05). (2005) 9–18

[54] Siberski, W., Pan, J.Z., Thaden, U.: Querying the Semantic Web with Preferences. In: Proceedings of the 5th International Semantic Web Conference (ISWC'06). (2006) 612–624

[55] Gurský, P., Horváth, T., Vojtás, P., Jirásek, J., Krajci, S., Novotny, R., Pribolová, J., Vaneková, V.: User Preference Web Search – Experiments with a System Connecting Web and User. Computing and Informatics Journal **28**(4) (2009) 515–553

[56] Toninelli, A., Corradi, A., Montanari, R.: Semantic-based Discovery to Support Mobile Context-aware Service Access. Computer Communications **31**(5) (2008) 935–949

[57] Roy, S.B., Thirumuruganathan, S., Amer-Yahia, S., Das, G., Yu, C.: Exploiting Group Recommendation Functions for Flexible Preferences. In: Proceedings of the 30th International Conference on Data Engineering (ICDE'14). (2014) 412–423

[58] Tapucu, D.: A Generic Model for Handling Preferences in Ontology Based Databases. PhD thesis, LISI/ENSMA et Université de Poitiers (2010)

[59] Tapucu, D., Jean, S., Aït-Ameur, Y., Ünalir, M.O.: An Extension of Ontology Based Databases to Handle Preferences. In: Proceedingss of the 11th International Conference on Enterprise Information Systems (ICEIS'09). (2009) 208–213

[60] Bazhar, Y.: Extension des systèmes de métamodélisation persistant avec la sémantique comportementale. PhD thesis, LIAS/ENSMA et Université de Poitiers (2013)

[61] Jarke, M., Jeusfeld, M.A., Nissen, H.W., Quix, C., Staudt, M.: Metamodelling with Datalog and Classes: ConceptBase at the Age of 21. In: Proceedings of the 2nd International Conference on Objects and Databases (ICOODB'09). (2009) 95–112

[62] Melnik, S., Rahm, E., Bernstein, P.A.: Rondo: A Programming Platform for Generic Model Management. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03). (2003) 193–204

[63] Hick, J., Hainaut, J.: Strategy for Database Application Evolution: The DB-MAIN Approach. In: Proceedings of the 22nd International Conference on Conceptual Modeling (ER'03). (2003) 291–306

[64] Miller, R.J., Hernández, M.A., Haas, L.M., Yan, L., Ho, C.T.H., Fagin, R., Popa, L.: The Clio Project: Managing Heterogeneity. SIGMOD Record **30**(1) (2001) 78–83

[65] The Eclipse Foundation: The CDO Model Repository (CDO) http://www.eclipse.org/cdo/. Technical report (2013)

[66] Koegel, M., Helming, J.: EMFStore : A Model Repository for EMF Models. In: Proceedings of the 32nd International Conference on Software Engineering (ICSE'10). (2010) 307–308

[67] Bazhar, Y., Aït-Ameur, Y., Jean, S.: BeMoRe: A Repository for Handling Models Behaviors. In: Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE'13). (2013) 262–267

[68] Bazhar, Y., Aït-Ameur, Y., Jean, S., Baron, M.: A Flexible Support of Non Canonical Concepts in Ontology-Based Databases. In: Proceedings of the 9th International Conference on Web Information Systems and Technologies (WEBIST'12). (2012) 393–398

[69] Bazhar, Y., Chakroun, C., Aït-Ameur, Y., Bellatreche, L., Jean, S.: Handling Non Canonical Concepts in Ontology-Based Database Design. In: Proceedings of 11th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE'12). (2012) 879–896

[70] Bazhar, Y., Ouhammou, Y., Aït-Ameur, Y., Grolleau, E., Jean, S.: Persistent Meta-Modeling Systems as Heterogeneous Model Repositories. In: Proceedings of the 3rd International Conference on Model and Data Engineering (MEDI'13). (2013) 25–37

[71] Aït-Ameur, Y., Baron, M., Bellatreche, L., Jean, S., Sardet, E.: Ontologies in Engineering: The OntoDB/OntoQL Platform. Soft Computing **21**(2) (2017) 369–389

[72] Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. W3C Recommendation 21 March 2013 (2013) https://www.w3.org/TR/sparql11-query/.

[73] Karvounarakis, G., Magkanaraki, A., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M., Tolle, K.: RQL: A Functional Query Language for RDF. In: The Functional Approach to Data Management: Modelling, Analyzing and Integrating Heterogeneous Data. (2004) 435–465

[74] Mastella, L.S.: Semantic Exploitation of Engineering Models: Application to Petroleum Reservoir Models. PhD thesis, Ecole Nationale Supérieure des Mines De Paris (2010)

[75] Belaid, N.: Modélisation de services et de workflows sémantiques à base d'ontologies de services et d'indexations. Application à la modélisation géologique. PhD thesis, LISI/ENSMA et Université de Poitiers (2011)

[76] Royer, K.: Vers un entrepôt de données et des processus : le cas de la mobilité électrique chez EDF. PhD thesis, LIAS/ENSMA et Université de Poitiers (2015)

[77] Jean, S., Aït-Ameur, Y., Pierra, G.: OntoQL: An Alternative to Semantic Web Query Languages. In International Journal of Semantic Computing (IJSC) **9**(1) (2015) 105–137

[78] Bouarar, S.: Vers une conception logique et physique des bases de données avancées dirigée par la variabilité. PhD thesis, LIAS/ENSMA et Université de Poitiers (2016)

[79] Silverston, L.: The Data Model Resource Book: A Library of Universal Data Models for All Enterprises. Data modeling and design. Wiley (2001)

[80] Tropashko, V.: SQL Design Patterns: Expert Guide to SQL Programming. Rampant Techpress (2008)

[81] Stathopoulou, E., Vassiliadis, P.: Design Patterns for Relational Databases. Technical report (2009) 1–38

[82] Bruno, N., Chaudhuri, S.: To Tune or Not to Tune?: A Lightweight Physical Design Alerter. In: Proceedings of the 32nd Internatio-

nal Conference on Very Large Data Bases (VLDB'06). (2006) 499–510

[83] Goasdoué, F., Karanasos, K., Leblay, J., Manolescu, I.: View Selection in Semantic Web Databases. VLDB Journal **5**(2) (2011) 97–108

[84] Castillo, R., Leser, U.: Selecting Materialized Views for RDF Data. In: Proceedings of the 10th International Conference on Web Engineering (ICWE'10 Workshops). (2010) 126–137

[85] Ibragimov, D., Hose, K., Pedersen, T.B., Zimányi, E.: Optimizing Aggregate SPARQL Queries Using Materialized RDF Views. In: Proceedings of the 15th International Semantic Web Conference (ISWC'16). (2016) 341–359

[86] Khouri, S.: Cycle de vie sémantique de conception de systèmes de stockage et de manipulation de données. PhD thesis, LIAS/ENSMA, Université de Poitiers et Ecole nationale Supérieure d'Informatique (ESI), Algérie (2013)

[87] Tueno Fotso, S.J., Frappier, M., Laleau, R., Mammar, A., Leuschel, M.: Formalisation of SysML/KAOS Goal Assignments with B System Component Decompositions. In: Proceedings of the 14th International Conference on Integrated Formal Methods (IFM'18). (2018) 377–397

[88] Vicente-Chicote, C., Moros, B., Toval, A.: REMM-Studio: An Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting. Journal of Object Technology **6(9)** (2007) 437–454

[89] Brottier, E., Baudry, B., Y.Traon, Touzet, D., Nicolas, B.: Producing a Global Requirement Model from Multiple Requirement Specifications. In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, (EDOC'07). (2007) 390–404

[90] Nguyen, T.H., Grundy, J., Almorsy, M.: Integrating Goal-oriented and Use Case-based Requirements Engineering: The Missing Link. In: Proceedings of the 18th International Conference on Model Driven Engineering Languages and Systems (MODELS'15). (2015) 328–337

[91] Mirbel, I., Villata, S.: An Argumentation-based Support System for Requirements Reconciliation. In Parsons, S., Oren, N., Reed, C., Cerutti, F., eds.: Proceedings of Computational Models of Argument (COMMA'14). (2014) 467–468

[92] Doan, A., Halevy, A.Y., Ives, Z.G.: Principles of Data Integration. Morgan Kaufmann (2012)

[93] Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-based Integration of Information - A Survey of Existing Approaches. In: Proceedings of the International Workshop on Ontologies and Information Sharing. (2001) 108–117

[94] Bourque, P., Fairley, R.E.: Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0. IEEE Computer Society (2014)

[95] Nuseibeh, B., Easterbrook, S.: Requirements Engineering: A Roadmap. In: Proceedings of the 22nd International Conference on Software Engineering (ICSE'00). (2000) 35–46

[96] Rochfeld, A., Tardieu, H.: MERISE: An Information System Design and Development Methodology. Information & Management **6**(3) (1983) 143–159

[97] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Me-

thodology. Autonomous Agents and Multi-Agent Systems **8** (2004) 203–236

[98] Boukhari, I.: Intégration et exploitation de besoins en entreprise étendue fondées sur la sémantique. PhD thesis, LIAS/ENSMA et Université de Poitiers (2014)

[99] Lapeña, R., Pérez, F., Cetina, C., Pastor, O.: Improving Traceability Links Recovery in Process Models Through an Ontological Expansion of Requirements. In: Proceedings of the 31st International Conference on Advanced Information Systems Engineering (CAISE'19). (2019) 261–275

[100] Boukhari, I., Jean, S., Ait-Sadoune, I., Bellatreche, L.: The Role of User Requirements in Data Repository Design. In International Journal on Software Tools for Technology Transfer (STTT) (2018) 19–34

[101] Davoudian, A., Chen, L., Liu, M.: A Survey on NoSQL Stores. ACM Computer Surveys **51**(2) (2018) 40:1–40:43

[102] Almassabi, A., Bawazeer, O., Adam, S.: Top NewSQL Databases and Features Classification. International Journal of Database Management Systems **10** (2018) 11–31

[103] Golfarelli, M., Rizzi, S.: Data Warehouse Testing: A Prototype-based Methodology. Information & Software Technology **53**(11) (2011) 1183–1198

[104] Bosch, J.: Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach. ACM Press/Addison-Wesley Publishing (2000)

[105] Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical report, Carnegie-Mellon University Software Engineering Institute (1990)

[106] Jean, S., Bellatreche, L., Fokou, G., Baron, M., Khouri, S.: OntoDBench: Novel Benchmarking System for Ontology-Based Databases. In: Proceedings of the 11th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE'12). (2012) 897–914

[107] Yang, J., Karlapalem, K., Li, Q.: Algorithms for Materialized View Design in Data Warehousing Environment. In: Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB 1997). (1997) 136–145

[108] Gupta, H., Mumick, I.S.: Selection of Views to Materialize Under a Maintenance Cost Constraint. In: Proceedings of the 7th International Conference on Database Theory (ICDT 1999). (1999) 453–470

[109] Boukorca, A., Bellatreche, L., Senouci, S.B., Faget, Z.: Coupling Materialized View Selection to Multi Query Optimization: Hyper Graph Approach. International Journal of Data Warehousing and Mining **11**(2) (2015) 62–84

[110] Dritsou, V., Constantopoulos, P., Deligiannakis, A., Kotidis, Y.: Optimizing Query Shortcuts in RDF Databases. In: Proceedings of the 8th Extended Semantic Web Conference (ESWC'11). (2011) 77–92

[111] Gallego, M.A., Fernández, J.D., Martínez-Prieto, M.A., de la Fuente, P.: An Empirical Study of Real-World SPARQL Queries. In: Proceedings of the 1st International Workshop on Usage Analysis and the Web of Data (USEWOD'11). (2011)

[112] Mbaiossoum, B.L., Bellatreche, L., Jean, S.: Materialized View Selection Considering the Diversity of Semantic Web Databases. In: Proceedings of the 18th East-European Conference on Advances in Databases and Information Systems (ADBIS'14). (2014) 163–176

[113] Mbaiossoum, B.L.: Conception physique des bases de données à base ontologique : le

cas des vues matérialisées. PhD thesis, LIAS/ENSMA et Université de Poitiers (2014)

[114] Ramakrishnan, R., Gehrke, J.: Database Management Systems. McGraw-Hill (2003)

[115] Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL Basic Graph Pattern Optimization Using Selectivity Estimation. In: Proceedings of the 17th International Conference on World Wide Web (WWW'08). (2008) 595–604

[116] Neumann, T., Moerkotte, G.: Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins. In: Proceedings of the 27th International Conference on Data Engineering (ICDE'11). (2011) 984–994

[117] Chong, E.I., Das, S., Eadon, G., Srinivasan, J.: An Efficient SQL-based RDF Querying Scheme. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05). (2005) 1216–1227

[118] Lu, J., Ma, L., Zhang, L., Brunner, J., Wang, C., Pan, Y., Yu, Y.: SOR: A Practical System for Ontology Storage, Reasoning and Search. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07). (2007) 1402–1405

[119] Bornea, M.A., Dolby, J., Kementsietsidis, A., Srinivas, K., Dantressangle, P., Udrea, O., Bhattacharjee, B.: Building an Efficient RDF Store Over a Relational Database. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD'13). (2013) 121–132

[120] Khouri, S., Boukhari, I., Bellatreche, L., Jean, S., Sardet, E., Baron, M.: Ontology-based Structured Web Data Warehouses for Sustainable Interoperability: Requirement Modeling, Design Methodology and Tool. In Computers in Industry Journal **63**(8) (2012) 799–812

[121] Bouarar, S., Bellatreche, L., Jean, S., Baron, M.: Do Rule-based Approaches Still Make Sense in Logical Data Warehouse Design? In: Proceedings of the 18th East-European Conference on Advances in Databases and Information Systems (ADBIS'14). (2014) 83–96

[122] Rosenmüller, M., Siegmund, N., Thüm, T., Saake, G.: Multi-dimensional Variability Modeling. In: Fifth International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS'11). (2011) 11–20

[123] Gaasterland, T., Godfrey, P., Minker, J.: An Overview of Cooperative Answering. Journal of Intelligent Information Systems (JIIS) **1**(2) (1992) 123–157

[124] Godfrey, P.: Minimization in Cooperative Response to Failing Database Queries. International Journal of Cooperative Information Systems **6**(2) (1997) 95–149

[125] Jannach, D.: Fast Computation of Query Relaxations for Knowledge-based Recommenders. AI Communications **22**(4) (2009) 235–248

[126] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14). (2014) 601–610

[127] Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast Rule Mining in Ontological Knowledge Bases with AMIE+. VLDB Journal **24**(6) (2015) 707–730

[128] Jayaram, N., Khan, A., Li, C., Yan, X., Elmasri, R.: Towards a Query-by-Example System for Knowledge Graphs. In: Proceedings of Workshop on GRAph Data Ma-

nagement Experiences and Systems (GRADES'14). (2014) 1–6

[129] Campinas, S.: Live SPARQL Auto-Completion. In: Proceedings of the ISWC 2014 Posters & Demonstrations Track. (2014) 477–480

[130] Jiang, J.Y., Ke, Y.Y., Chien, P.Y., Cheng, P.J.: Learning User Reformulation Behavior for Query Auto-completion. In: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'14). (2014) 445–454

[131] Xiao, C., Qin, J., Wang, W., Ishikawa, Y., Tsuda, K., Sadakane, K.: Efficient Error-tolerant Query Autocompletion. VLDB Journal **6**(6) (2013) 373–384

[132] Nandi, A., Jagadish, H.V.: Assisted Querying Using Instant-response Interfaces. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD'07). (2007) 1156–1158

[133] De Virgilio, R., Maccioni, A., Torlone, R.: A Similarity Measure for Approximate Querying Over RDF data. In: Proceedings of the Joint EDBT/ICDT 2013 Workshops. (2013) 205–213

[134] Carpio, G.V.G., Abrouk, L., Cullot, N.: A Query Expansion Methodology in a Cooperation of Information Systems Based on Ontologies. In: Proceedings of the 5th International Conference on Web Information Systems and Technologies (WEBIST'09). (2009) 256–262

[135] Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Gandon, F.: Searching the Semantic Web: Approximate Query Processing Based on Ontologies. IEEE Intelligent Systems **21**(1) (2006) 20–27

[136] Bidoit, N., Herschel, M., Tzompanaki, A.: Efficient Computation of Polynomial Explanations of Why-Not Questions. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM'15). (2015) 713–722

[137] Song, Q., Namaki, M.H., Wu, Y.: Answering Why-Questions for Subgraph Queries in Multi-attributed Graphs. In: Proceedings of the 35th International Conference on Data Engineering (ICDE'19). (2019) 40–51

[138] Calí, A., Frosini, R., Poulovassilis, A., Wood, P.: Flexible Querying for SPARQL. In: Proceedings of the 13th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE'14). (2014) 473–490

[139] McSherry, D.: Incremental Relaxation of Unsuccessful Queries. In: Advances in Case-Based Reasoning. Volume 3155. (2004) 131–148

[140] Bosc, P., Hadjali, A., Pivert, O.: Incremental Controlled Relaxation of Failing Flexible Queries. Journal of Intelligent Information Systems **33**(3) (2009) 261–283

[141] Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. ACM Transactions on Database Systems (TODS) **34**(3) (2009) 16:1–16:45

[142] Fokou, G., Jean, S., Hadjali, A., Baron, M.: RDF Query Relaxation Strategies Based on Failure Causes. In: Proceedings of the 13th Extended Semantic Web Conference (ESWC'16). (2016) 439–454

[143] Hose, K., Vlachou, A.: A Survey of Skyline Processing in Highly Distributed Environments. VLDB Journal **21**(3) (2012) 359–384

[144] Fokou, G., Jean, S., Hadjali, A., Baron, M.: Handling Failing RDF Queries: From Diagnosis to Relaxation. Knowledge and Information Systems (KAIS) **50**(1) (2017) 167–195

[145] Mannila, H., Toivonen, H.: Levelwise Search and Borders of Theories in Know-

ledge Discovery. Data Mining and Knowledge Discovery **1**(3) (1997) 241–258

[146] Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., Sharm, R.S.: Discovering All Most Specific Sentences. ACM Transactions on. Database Systems **28**(2) (2003) 140–174

[147] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E.R., Mitchell, T.M.: Toward an Architecture for Never-Ending Language Learning. In: Association for the Advancement of Artificial Intelligence (AAAI). Volume 5. (2010) 1306–1313

[148] Hartig, O.: Querying Trust in RDF Data with tSPARQL. In: Proceedings of the 6th European Semantic Web Conference (ESWC'09). (2009) 5–20

[149] Tomaszuk, D., Pak, K., Rybiński, H.: Trust in RDF graphs. In: Proceedings of the 16th East European Conference (ADBIS'12). (2012) 273–283

[150] Pivert, O., Smits, G.: How to Efficiently Diagnose and Repair Fuzzy Database Queries that Fail. In: Fifty Years of Fuzzy Logic and its Applications, Studies in Fuzziness and Soft Computing. (2015) 499–517

[151] Smits, G.: Personnalisation et enrichissement des méthodes d'accès aux données. HDR, Université Bretagne Loire (2018)

[152] Dellal, I., Jean, S., Hadjali, A., Chardin, B., Baron, M.: On Addressing the Empty Answer Problem in Uncertain Knowledge Bases. In: Proceedings of the 28th International Conference on Database and Expert Systems Applications (DEXA'17). (2017) 120–129

[153] Dellal, I., Jean, S., Hadjali, A., Chardin, B., Baron, M.: Query Answering Over Uncertain RDF Knowledge Bases: Explain and Obviate Unsuccessful Query Results. Knowledge and Information Systems (KAIS) **61**(3) (2019) 1633–1665

[154] Fokou, G., Jean, S., Hadjali, A.: Endowing Semantic Query Languages with Advanced Relaxation Capabilities. In: Proceeding of the 21st International Symposium on Methodologies for Intelligent Systems (ISMIS'14). (2014) 512–517

[155] Poulovassilis, A., Wood, P.T.: Combining Approximation and Relaxation in Semantic Web Path Queries. In: Proceedings of the 9th International Semantic Web Conference (ISWC'10). (2010) 631–646

[156] Hogan, A., Mellotte, M., Powell, G., Stampouli, D.: Towards Fuzzy Query-relaxation for RDF. In: Proceeding of the 9th Extended Semantic Web Conference (ESWC'12). (2012) 687–702

[157] Elbassuoni, S., Ramanath, M., Weikum, G.: Query Relaxation for Entity-Relationship Search. In: Proceeding of the 8th Extended Semantic Web Conference (ESWC'11). (2011) 62–76

[158] Dolog, P., Stuckenschmidt, H., Wache, H., Diederich, J.: Relaxing RDF Queries Based on User and Domain Preferences. Journal of Intelligent Information Systems **33**(3) (2009) 239–260

[159] Belheouane, C., Jean, S., Chardin, B., Hadjali, A., Azzoune, H.: Cooperative Treatment of Failing Queries over Uncertain Databases: A Matrix-Computation-Based Approach. Journal of Intelligent Information Systems (JIIS) **51**(1) (2019) 211–238

[160] Mottin, D., Marascu, A., Roy, S.B., Das, G., Palpanas, T., Velegrakis, Y.: A Holistic and Principled Approach for the Empty-Answer Problem. VLDB Journal **25**(4) (2016) 597–622

[161] Fokou, G.: Conception d'un framework pour la relaxation des requêtes SPARQL. PhD thesis, LIAS/ENSMA et Université de Poitiers (2016)

[162] Dellal, I.: Gestion et Exploitation de Larges Bases de Connaissances en Présence de Données Incomplètes et Incertaines. PhD thesis, LIAS/ENSMA et Université de Poitiers (2019)

[163] Belheouane, C.: Bases de données coopératives et intelligentes. PhD thesis, LIAS/ENSMA, Université de Poitiers et Université des Sciences et de la Technologie Houari Boumedienne (USTHB), Algérie (2019)

[164] Fokou, G., Jean, S., Hadjali, A.: QaRS: A User-Friendly Graphical Tool for Semantic Query Design and Relaxation. In: Proceedings of the 18th International Conference on Extending Database Technology (EDBT'15). (2015) 553–556

[165] Fokou, G., Jean, S., Hadjali, A., Baron, M.: Cooperative Techniques for SPARQL Query Relaxation in RDF Databases. In: Proceedings of the 12th Extended Semantic Web Conference (ESWC'15). (2015) 237–252

[166] Le, W., Kementsietsidis, A., Duan, S., Li, F.: Scalable Multi-query Optimization for SPARQL. In: Proceedings of the 28th International Conference on Data Engineering (ICDE'12). (2012) 666–677

[167] Ell, B., Harth, A., Simperl, E.: SPARQL Query Verbalization for Explaining Semantic Search Engine Queries. In: Proceedings of the 11th Extended Semantic Web Conference (ESWC'14). (2014) 426–441

[168] Taleb, I., Serhani, M.A., Dssouli, R.: Big Data Quality: A Survey. In: Proceedings oof the IEEE International Congress on Big Data (BigData Congress'18). (2018) 166–173

[169] Marcén, A.C., Lapeña, R., Pastor, O., Cetina, C.: Traceability Link Recovery Between Requirements and Models Using an Evolutionary Algorithm Guided by a Learning to Rank Algorithm: Train Control and Management Case. Journal of Systems and Software **163** (2020) 110519

[170] Oppold, S., Herschel, M.: Accountable Data Analytics Start with Accountable Data: The LiQuID Metadata Model. In: Proceedings of the ER Forum, Demo and Posters (ER'20). (2020) 59–72

[171] De Siqueira Braga, D., Niemann, M., Hellingrath, B., De Lima Neto, F.B.: Survey on Computational Trust and Reputation Models. ACM Computing Surveys **51**(5) (2019) 101:1–101:40

[172] Pinyol, I., Sabater-Mir, J.: Computational Trust and Reputation Models for Open Multi-agent Systems: A Review. Artificial Intelligence Review **40**(1) (2013) 1–25

[173] Cugola, G., Margara, A.: Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Computing Surveys **44**(3) (2012) 15:1–15:62

[174] Hirzel, M., Baudart, G., Bonifati, A., Valle, E.D., Sakr, S., Vlachou, A.: Stream Processing Languages in the Big Data Era. SIGMOD Record **47**(2) (2018) 29–40

[175] Haas, P.J.: Data-Stream Sampling: Basic Techniques and Results. In: Data Stream Management: Processing High-Speed Data Streams. (2016) 13–44

[176] Dell'Aglio, D., Valle, E.D., van Harmelen, F., Bernstein, A.: Stream Reasoning: A Survey and Outlook. Data Science **1**(1-2) (2017) 59–83

[177] Teymourian, K., Paschke, A.: Semantic Enrichment of Event Stream for Semantic Situation Awareness. In: Semantic Web: Implications for Technologies and Business Practices. (2016) 185–212

[178] Ramírez-Gallego, S., Krawczyk, B., García, S., Wozniak, M., Herrera, F.: A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions. Neurocomputing **239** (2017) 39–57

[179] Vigier, F., Andrault, C., Chauvaud, F., Veillon, D.: Le Palais des comtes de Poitiers du Moyen Age à nos jours. Revue Historique du Centre-Ouest (2014) 349–381

[180] Palazzo-Bertholon, B., Treffort, C.: Pour une relecture de l'hypogée des Dunes à Poitiers. Approche méthodologie et interdisciplinaire. In: Actes des XXVIIIe journées internationales d'archéologie mérovingienne. (2007) 151–170

[181] Manuel, A., M'Darhri, A.A., Abergel, V., Rozar, F., De Luca, L.: A Semi-Automatic 2D/3D Annotation Framework for the Geometric Analysis of Heritage Artefacts. In: Proceedings of the 3rd Digital Heritage International Congress. (2018) 1–7

[182] Messaoudi, T., Véron, P., Halin, G., De Luca, L.: An Ontological Model for the Reality-Based 3D Annotation of Heritage Building Conservation State. Journal of Cultural Heritage **29** (2018) 100–112

[183] Abergel, V.: Relevé numérique d'art pariétal : définition d'une approche innovante combinant propriétés géométriques, visuelles et sémantiques au sein d'un environnement de réalité mixte. PhD thesis, HESAM Université (2020)

[184] Doerr, M.: The CIDOC CRM, an Ontological Approach to Schema Heterogeneity. In: Semantic Interoperability and Integration. (2005)

[185] Marden, J., Li-Madeo, C., Whysel, N., Edelstein, J.: Linked Open Data for Cultural Heritage: Evolution of an Information Technology. In: Proceedings of the 31st ACM International Conference on Design of Communication (SIGDOC'13). (2013) 107–112

[186] Diefenbach, D., López, V., Singh, K.D., Maret, P.: Core Techniques of Question Answering Systems Over Knowledge Bases: A Survey. Knowledge and Information Systems **55**(3) (2018) 529–569

[187] Wu, Q., Teney, D., Wang, P., Shen, C., Dick, A.R., van den Hengel, A.: Visual Question Answering: A Survey of Methods and Datasets. Computer Vision and Image Understanding **163** (2017) 21–40

[188] Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: ImageNet: A Large-Scale Hierarchical Image Database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09). (2009) 248–255

[189] Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., Bernstein, M.S., Fei-Fei, L.: Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. International Journal of Computer Vision **123**(1) (2017) 32–73

La sémantique : une valeur ajoutée pour la conception et l'exploitation des données

Présenté par :

Stéphane JEAN

Résumé. Avec le développement des nouvelles technologies et du Web, de plus en plus de données sont produites. Généralement, celles-ci sont conçues pour un contexte d'utilisation particulier qui n'est pas explicité. En conséquence, la connaissance associée à l'information représentée est souvent partiellement définie et basée sur des règles d'interprétation externes. Cela pose des problèmes lorsqu'il s'agit d'utiliser ces données dans un autre contexte que celui prévu par leurs concepteurs. C'est par exemple le cas lorsque l'on souhaite partager, rechercher ou analyser ces données.

Pour répondre à cette problématique, la notion *d'ontologie* s'est imposée comme modèle de référence pour représenter la sémantique des concepts d'un domaine d'étude. Ainsi, de nombreuses ontologies ont été conçues dans le contexte du Web sémantique pour améliorer la recherche d'information ou la traduction automatique de textes. C'est également le cas dans les domaines de l'ingénierie où le standard PLIB est utilisé afin d'expliciter la sémantique des données techniques manipulées dans ces domaines. Cette diversité d'ontologies est le fil directeur de nos travaux de recherche. Ces derniers portent sur les bases de données permettant la persistance de ces ontologies, nommées *Bases de Données Sémantiques (BDS)*. Nous avons développé trois axes de recherche autour de ces BDS.

Notre premier axe de recherche est une extension de nos travaux de thèse sur la BDS OntoDB/OntoQL. Sa particularité est de permettre de prendre en compte différents types d'ontologies. Nous proposons d'abord des extensions de cette BDS pour diverses problématiques telles que la prise en compte des préférences utilisateur ou l'annotation sémantique de modèles métiers utilisés en ingénierie. Puis, nous présentons une généralisation de cette plateforme pour qu'elle puisse être utilisée pour manipuler des modèles plus généraux que les ontologies.

Dans notre second axe de recherche sur les BDS, nous avons constaté que les particularités des ontologies nécessitaient de revoir le processus de conception classique mis en œuvre dans les bases de données usuelles. Les applications basées sur des ontologies impliquant généralement différents partenaires ayant des besoins hétérogènes, nous avons proposé d'ajouter une phase d'intégration des besoins à ce processus. Par ailleurs, la diversité des ontologies et en conséquence des BDS, nous a amenés à proposer une démarche générale de conception d'un tel système qui explicite les différents choix possibles lors de ce processus.

Enfin, notre troisième axe de recherche concerne l'exploitation des BDS via des requêtes. Les spécificités des ontologies rendent ces systèmes difficiles à utiliser. Aussi, nos travaux visent à proposer des techniques pour aider l'utilisateur lorsqu'une requête exprimée sur une BDS ne retourne pas de résultat. L'originalité de nos travaux est d'essayer d'identifier les raisons pour lesquelles la requête utilisateur retourne ce résultat insatisfaisant. Grâce à l'identification de ces causes d'échec, nous proposons des techniques de relaxation de requêtes sémantiques plus efficaces que celles de l'état de l'art.

Mots-clés : Ontologie, Base de données, Web sémantique, Langage de requêtes, Optimisation de requêtes, Métamodélisation.