REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET
DE LA RECHERCHE SCIENTIFIQUE

Université Mouloud MAMMERI de Tizi-Ouzou
Faculté de génie électrique et de l'informatique
Département d'informatique

Mémoire de Fin d'Etudes

En vue de l'obtention du diplôme de Master en informatique
***Option** : Systèmes Informatiques (SI)*

Thème

Technologies du Web Sémantique pour manipuler une ontologie de domaine de compétences

Proposé et dirigé par :

M^me^ BOUARAB. F

Réalisé par :

M^elle^ MEDROUH Kahina

Promotion 2014/2015

Remerciements

Je remercie tout d'abord Dieu qui m'a donné la foi et le courage pour accomplir ce projet.

Je tiens à exprimer mes plus vifs remerciements et ma profonde gratitude à ma promotrice ***M^{me} BOUARAB.F*** *pour le temps précieux qu'elle m'a consacré, pour son suivi attentif et ses conseils inestimables qu'elle m'a donné, qu'elle trouve ici l'expression de ma haute considération et mon profond respect ;*

Mes plus vifs remerciements vont également aux membres du jury pour avoir accepté d'honorer par leur jugement mon travail ;

Je tiens aussi à remercier ***M^{me} BERKANE.T*** *pour leur aide précieuse ;*

En fin je tiens à exprimer mes plus sincères sentiments à ma très chère famille, à tous mes amis (es) et à toute personne qui m'a aidé de près ou de loin pour la réalisation de ce modeste travail.

Merci beaucoup

Liste des figures

Liste des tableaux

Sommaire

Chapitre I

Le Web Sémantique et les Ontologies

Chapitre II

L'approche par compétences

Chapitre III

Analyse et Conception

Chapitre IV

Réalisation

Introduction générale :

Internet est un réseau informatique reliant un grand nombre d'ordinateurs dans le monde et proposant à ses utilisateurs des services tels que l'email, la messagerie et le World Wide Web. Ce dernier service, plus communément appelé le Web, permet d'accéder à divers documents reliés les uns aux autres au travers de liens HyperText. De tels documents sont le plus souvent de nature textuelle et formatés dans le HyperText Markup Language (HTML), lequel permet de décrire la structure et le contenu de documents afin de les rendre interprétables par un navigateur Web. Une version plus récente de ce langage, le eXtensible HTML (XHTML), est basée sur la syntaxe définie par le eXtensible Markup Language (XML) et a été réalisée par le World Wide Web Consortium (W3C) afin de disposer d'un langage plus extensible que le HTML et d'augmenter l'interopérabilité avec d'autres formats de données.

De par la grande quantité de documents qu'il fournit, le World Wide Web est devenue une source de données incontournable lors de la recherche d'informations. Toutefois, les moteurs de recherche actuels tels Google et Yahoo fournissent des milliers de réponses dont une grande partie se révèle le plus souvent non pertinente tandis que les informations les plus intéressantes ne sont pas toujours présentées. Cette infructuosité des recherches provient du fait que les ressources présentes dans le Web ne sont compréhensibles que par les humains, de même que les liens HyperText reliant les divers documents. Ceci est une conséquence directe du langage utilisé pour présenter les pages Web, le XHTML n'étant en effet destiné qu'à formater les données pour les rendre compréhensibles par les utilisateurs.

Le Web Sémantique, présenté par T. Berners-Lee, J. Hendler et O. Lassila, est une évolution du Web consistant à insérer des métadonnées aux ressources Web afin de leur ajouter du sens et de les rendre compréhensibles par des systèmes informatiques. Ces derniers seraient alors en mesure d'interpréter les ressources Web et de répondre de manière plus adaptée aux requêtes des utilisateurs.

Une recherche automatique du Web est rendue possible dans le Web Sémantique par le fait que les ressources Web ne contiennent plus uniquement des éléments textuels compréhensibles par les utilisateurs mais également les noms des concepts associés à ces éléments sont identifiés univoquement. Ces concepts font partie d'un vocabulaire commun définit dans des ontologies et partagés à travers le Web Sémantique. Une ontologie est un ensemble structuré de termes permettant la représentation et la description de concepts dans

un domaine de connaissance. Les relations entre ces concepts sont également représentées et il est possible de réaliser des raisonnements sur les objets du domaine grâce à la définition de règles. Ces dernières permettent aux agents du Web Sémantique d'inférer de l'information sur base des ressources Web et de ces règles et, dès lors, de réaliser des raisonnements automatisés. **[1]**

Ce mémoire de fin d'études présente une ontologie de domaine de compétences à intégrer dans un système adaptatif éducatif. Le méta-modèle obtenu est formalisé dans un langage cohérent qui permet de montrer les compétences à acquérir dans une discipline donnée, notre travail consiste à montrer comment les techniques du web sémantique peuvent être utilisées pour exploiter et manipuler une base de connaissances générée à partir de cette ontologie de domaine. Ce document est découpé de la manière suivante :

Nous introduisons tout d'abord, dans le chapitre I, les différents langages et outils proposés par le W3C dans le cadre du Web Sémantique permettant d'ajouter de la sémantique au Web ainsi que d'interroger ses ressources, ensuite nous présentons les ontologies. Le chapitre II étudie les principales théories de l'apprentissage sous les deux approches pédagogiques (par objectif et par compétences), et l'intégration de l'approche par compétences dans les systèmes adaptatifs éducatifs. La construction et l'opérationnalisation de l'ontologie de domaine de compétences seront présentées dans le chapitre III (Analyse et Conception), en mettant en application l'outil d'édition adapté selon nos besoins. Le chapitre IV (Réalisation) est consacré pour la manipulation et l'exploitation du fichier OWL édité à partir de l'ontologie de domaine de compétences, il exposera aussi le choix des outils utilisés ainsi que les phases de réalisation de notre application.

Introduction :

Le web est constitué par un ensemble de documents, principalement textuels, formatés dans un langage particulier (HTML) permettant d'exprimer des liens entre un objet dans le document source (l'ancre) et un objet du document cible. Ce web est exploité par des dispositifs logiciels (navigateurs ou robots de recherche) qui traversent ces liens lorsqu'ils les rencontrent (ou lorsque l'utilisateur clique sur une ancre). Le travail d'exploitation de ce web est donc principalement dévolu aux utilisateurs humains qui doivent analyser le contenu des pages pour déterminer sur quel lien cliquer. Des dispositifs logiciels peuvent les aider en analysant ce contenu, mais leur aide, bien que remarquable, reste limitée car le contenu des documents du web s'adresse aux utilisateurs humains.

Le principe du web sémantique, comme celui du web, est son ouverture : la possibilité de faire référence à des ressources distantes, non maîtrisées et la possibilité d'enrichir ces ressources sans être entravé par des barrières physiques, techniques ou même réglementaires. Pour hériter de ces particularités du web, le web sémantique tirera parti d'un espace d'adressage global des ressources permettant d'ajouter en permanence de l'information à ce web et d'y avoir accès sans entrave. Il offrira des langages permettant de partager et d'échanger la description de ces ressources. Une application du web sémantique est donc une application qui se nourrit des descriptions des ressources du web à l'aide des outils du web sémantique et du langage permettant de le faire : RDF. Cependant, il est nécessaire de préciser le vocabulaire utilisé pour décrire les documents. D'autre part, la recherche de documents dépend d'informations sortant du domaine des documents, par exemple des informations sur les personnes ou sur les bases de données. Pour cela, on développe un modèle conceptuel des objets considérés qui circonscrit le vocabulaire propre au domaine et contraint le sens des concepts et de leurs relations. Ce modèle, qualifié d'ontologie, dispose d'un langage adapté pour s'exprimer dans le web sémantique : OWL. Enfin, il sera nécessaire de tirer parti de ces descriptions et de ces ontologies pour retrouver les documents. **[2]**

I. Le web sémantique :

L'idée de web sémantique est de permettre une recherche intelligente sur le Web, faite par des ordinateurs et basée sur des définitions qu'ils puissent comprendre, des définitions données pour le monde entier. En faisant une requête sur un moteur proposant de la recherche en langage naturel, vous l'interrogerez comme vous parlez, et il transformera cette demande en langage compréhensible et cohérent pour la machine. **[3]**

I.1 Définition du web sémantique :

La prochaine évolution du Web est d'arriver à un Web intelligent, où les informations ne seraient plus stockées mais comprises par les ordinateurs afin d'apporter à l'utilisateur ce qu'il cherche vraiment. D'après la définition de Tim Berners-Lee, le Web sémantique permettra contrairement au Web actuel qui est vu comme un Web syntaxique, de rendre de contenu sémantique des ressources Web interprétables non seulement par l'homme mais aussi par la machine.

Le terme de web sémantique a été proposé par Tim Berners-Lee en 2001 (« The Semantic Web », *Scientific American Magazine*, May 17, 2001) pour désigner une évolution du web qui permettrait aux données disponibles (contenus, liens) d'être plus facilement utilisables et interprétables automatiquement, par des agents logiciels. Pour permettre cette évolution, un certain nombre de standards et de technologies ont été développés par le W3C, avec pour objectif de sortir les données des silos fermés que constituent les bases de données en ligne. **[5]**

Le Web sémantique désigne un ensemble de technologies visant à rendre le contenu des ressources du World Wide Web accessible et utilisable par les programmes et agents logiciels, grâce à un système de métadonnées formelles, utilisant notamment la famille de langages développés par le W3C**.** Il est construit sur la capacité de XML de définir des schémas de balisage personnalisés, sur la flexibilité de l'approche RDF pour représenter les ressources et sur la modélisation sémantique de ressources de langage d'ontologies OWL. **[6]**

I.2 Le web actuel et le web sémantique :

Le web actuel	Le web sémantique
Ensemble de documents	Ensemble de connaissances
Basé essentiellement sur HTML	Basé sur XML et RDF(S)
Recherche par mots clé	Recherche par concepts
Utilisable par l'humain	Utilisable par la machine

Tableau 1 : Comparaison entre le web actuel et le web sémantique. **[4]**

I.3 L'architecture du web sémantique :

L'architecture du web sémantique proposée par le W3C s'appuie sur une pyramide de langages comme le montre la figure ci-dessous.

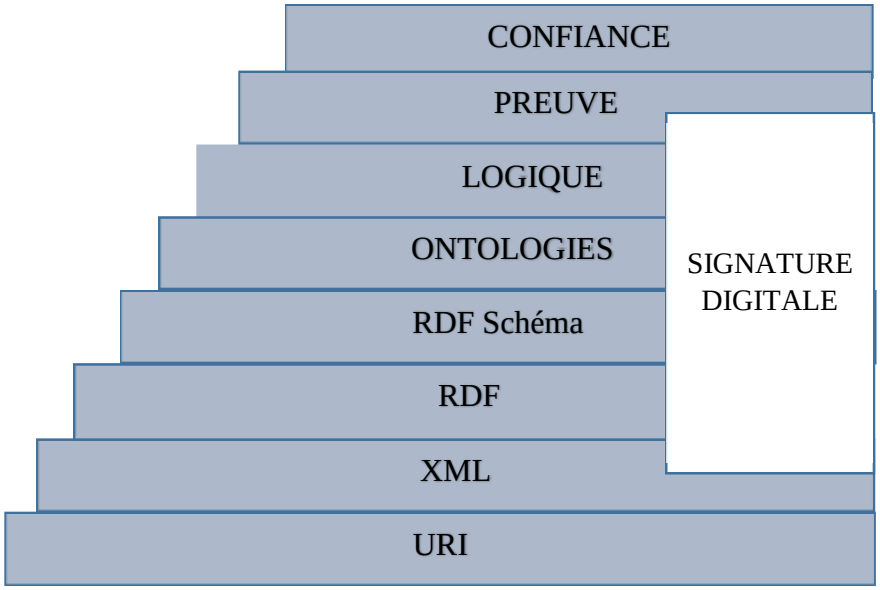

Figure 1 : Les couches du web sémantique. **[4]**

- Au niveau le plus bas se trouve les données les plus brutes, elles possèdent une adresse URI (*Uniforme Ressource Identifier)* qui permet d'attribuer un identifiant unique à chaque ressource.
- Ces données peuvent être structurées grâce à un langage de balises tel que XML (eXtensible Markup Language). La syntaxe XML peut être considérée comme un premier niveau de sémantique, elle permet aux utilisateurs de structurer des données en fonction de leur contenu sans rien dire de la signification des structures.
- Pour attribuer une signification à cette structure et relier d'une façon pertinente les différents éléments, Tim Berners-Lee propose le standard RDF (*Resource Description Framework)*, langage qui définit un cadre général pour la standardisation des métadonnées des ressources Web. Ce langage à pour but de donner une organisation plus structurée des informations présentées sur le Web à travers une description sémantique des données fournies par XML.
- Pour assurer une bonne distinction des différents concepts, il faut définir les uns par rapport aux autres, ce qui est possible avec RDFS et OWL. **[6]**
- Le niveau ontologie est une spécification explicite et formelle de la conceptualisation.
- La logique est un langage permettant d'exprimer des «règles» de raisonnement ces règles permettent de déduire de nouveaux faits à partir des faits existants et la preuve est en quelque sorte une suite d'applications de règles qui permettent de déduire un nouveau fait.
- Si on utilise des agents pour prendre des décisions à notre place, il faudrait qu'on puisse avoir «confiance» aux résultats pour ce faire, il faut que l'agent utilisé puisse expliquer

clairement comment il arrive à ses conclusion (preuve) et garantir la fiabilité et l'origine des informations utilisées (signature digitale). **[4]**

I.4 Les technologies du web sémantique :

L´évolution du Web vers le Web Sémantique nécessite le développement de langages permettant de représenter l'information d'une manière structurée et compréhensible par les systèmes informatiques. Un raisonnement automatisé des ressources Web doit également être rendu possible à travers l'établissement de règles d'inférence de même qu'une interrogation de ces ressources. Une collection de langages a été développée par le W3C à cette fin.

Nous présentons tout d'abord le Resource Description Framework, le langage de base du Web Sémantique permettant la représentation de l'information sous forme de triplets. Deux extensions de ce langage sont ensuite détaillées, le RDF Schéma et le Web Ontoloy Language, lesquels apportent les règles d'inférence nécessaires. Nous introduisons également le langage SPARQL qui permet d'interroger l'information représentée dans le langage RDF. Enfin, nous présentons plusieurs outils développés en vue de réaliser des applications basées sur les technologies du Web Sémantique. **[1]**

I.4.1 RDF :

RDF (Resource Description Framework) est un langage développé par le W3C, permettant la représentation de l'information dans le Web Sémantique en encodant de manière formelle les liens entre les ressources Web. Ce langage permet ainsi de d´écrire le contenu d'une ressource au travers de métadonnées telles que le nom de la ressource et son type. Ceci permet le traitement automatisé des ressources par des agents du Web Sémantique et l'échange de ces dernières entre diverses applications.

L'information est représentée sous forme de triplets (**sujet**, **prédicat**, **objet**). Le **sujet** est la ressource d´écrite par le triplet, le **prédicat** est une propriété de cette ressource et l'**objet** est la valeur prise par la propriété. L'objet peut être une valeur littérale, permettant de représenter des valeurs numériques ou des dates, ou une autre ressource. Le triplet (Belgique, type, pays) indique que la ressource « Belgique » possède une propriété « type » ayant la valeur « pays », représentant ainsi le fait que la Belgique est un pays. Dans ce cas, l'objet du triplet est une ressource. Dans le triplet (Hergé, hasBirthDay, 22-05-1907), représentant la

date de naissance d'Hergé, l'objet est un littéral étant donné qu'il s'agit d'une date de naissance.

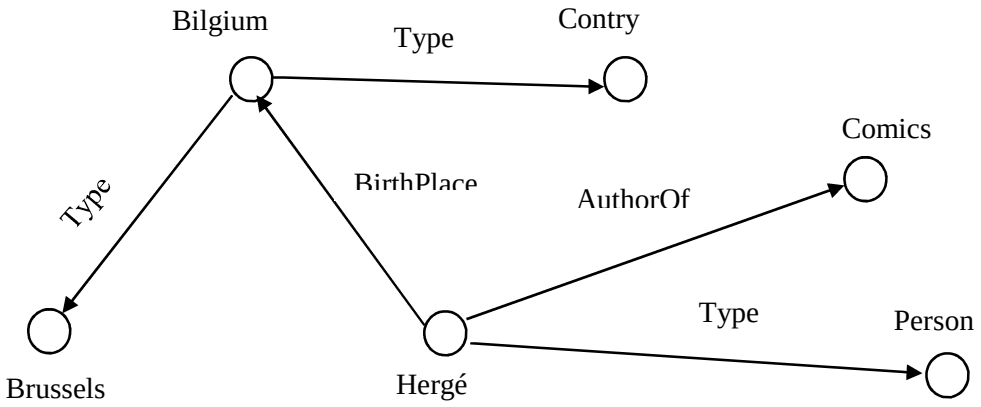

Figure 2 : Exemple de graphe RDF représentant un auteur ainsi que des informations sur son pays de naissance.

Un tel triplet peut également être représenté dans un graphe. Le sujet et l'objet sont des nœuds du graphe tandis que le prédicat est le label d'un arc directionnel, orienté du sujet vers l'objet, indiquant ainsi une relation entre ces derniers. Un ensemble de triplets constitue un graphe RDF. La Figure 2 représente un exemple de graphe RDF indiquant que la Belgique est de type « pays » et a pour capitale Bruxelles. L'auteur de bandes dessinées Hergé est également représenté et a la Belgique pour lieu de naissance.

Le langage RDF utilise des URIs (Uniform Resource Identifiers) afin d'identifier les ressources et les propriétés utilisées dans les triplets. Un identifiant uniforme de ressource (URI) est une chaîne de caractères permettant d'identifier une ressource, aussi bien abstraite que physique. Le sujet et le prédicat sont toujours des URIs tandis que l'objet peut être un URI ou un littéral, l'URI étant utilisé pour identifier une autre ressource. La Belgique peut ainsi, par exemple, être identifier par l'URI « http://dbpedia.org/resource/Belgium ».

Un sujet et un objet peuvent également être des nœuds anonymes (blank nodes), c'est-à-dire une ressource qui n'est pas identifiée par une URI et qui n'est pas un littéral. Il est toutefois possible d'identifier ce nœud à l'intérieur d'un graphe, de manière locale, en lui donnant un attribut rdf:nodeID, ce qui permet son utilisation dans plusieurs triplets d'un même graphe RDF. De tels nœuds peuvent par exemple être utilisés pour représenter le fait que deux personnes travaillent pour la même société, sans que cette dernière ne soit connue.

Les références URI utilisées dans les triplets pouvant atteindre une taille conséquente, un raccourci d'écriture a été défini dans la spécification RDF. Ce raccourci substitue les références URI par un nom XML qualifié, QName, contenant un préfixe qui est associé à un espace de noms URI et est suivi du caractère : ainsi qu'un nom local.

La référence URI « http://www.w3.org/1999/02/22-rdf-syntax-ns#Description » sera dès lors remplacée par le QName rdf:Description, où rdf: est un préfixe associé à l'espace de noms « http://www.w3.org/1999/02/22-rdf-syntax-ns# » et Description est le nom local.

L'utilisation des références URI pour identifier les ressources permet le développement de vocabulaires, des ensembles de références URI relatives à un même sujet spécifique et pouvant être partagés, menant ainsi à la réutilisation de ces vocabulaires par différentes personnes pour définir les mêmes concepts. Les références URI d'un même vocabulaire possèdent généralement le même espace de noms, et donc le même préfixe de QName, et diffèrent par le nom local ajouté à ce préfixe. Un tel exemple de vocabulaire est Friend of a Friend (FOAF), un vocabulaire RDF utilisé pour d'écrire des personnes et les relations entre ces dernières, comme le fait qu'une personne en connaisse une autre par exemple. Le préfixe commun à toutes les références URI de ce vocabulaire est foaf: et la propriété de connaissance a pour nom local knows.

Enfin, une syntaxe XML (eXtensible Markup Language) spécifique, appelée RDF/XML, a été spécifiée par le W3C afin de représenter les graphes RDF et de les rendre compréhensibles par des machines et échangeables entre applications. **[1]**

I.4.2 RDF Schema :

RDFS (RDF Schema) est un langage de description de vocabulaire RDF permettant la description formelle de hiérarchies et de relations entre les ressources. Toutes les références URI de RDFS sont regroupées dans l'espace de noms « http://www.w3.org/2000/01/rdf-schema# » auquel le préfixe rdfs: est conventionnellement associé. RDFS a été développé par le W3C.

Les ressources sont groupées en classes, la classe la plus générale étant rdfs:Resource. Cette dernière possède deux sous-classes, rdfs:Class qui représente l'ensemble des concepts définissables (Country et Region sont des instances de rdfs:Classe par exemple) et rdfs:Property dont toutes les propriétés définies dans un graphe RDF sont des instances. Il est

également possible de construire une hiérarchie de classes grâce à la propriété rdfs:subClassOf indiquant qu'une classe est une sous-classe d'une autre, par exemple Country subClassOf Region. Il en va de même pour les propriétés en utilisant rdfs:subPropertyOf.

RDFS permet également de définir des restrictions sur le domaine et l'espace d'arrivée des propriétés. La propriété rdfs:domain permet de définir la classe des sujets d'une propriété tandis que la propriété rdfs:range permet de définir la classe de ses objets. Par exemple, la propriété « date de naissance » ne peut s'appliquer qu'à des personnes (il s'agit de son domaine) et sa valeur doit être un littéral, comme indiqué ci-dessous. Notons également l'existence de collections et de containers dans RDFS permettant de rassembler des ressources.

ex: birthDay rdfs:domain foaf:Person

ex: birthDay rdfs:range rdf:Literal

Grâce à ce langage, il est possible de réaliser des raisonnements et de définir des ontologies simples. Connaissant les deux premières déclarations indiquées ci-dessous, la troisième peut être inférée par un moteur de raisonnement.

ex: Belgium rdf:type ex: Country

ex: Country rdfs:subClassOf ex: Region

ex: Belgium rdf:type ex: Region **[1]**

I.4.3 OWL :

OWL (Web Ontology Language) est un langage permettant la représentation d'ontologies dans le Web, donnant ainsi davantage de sens aux classes et relations définies entre celles-ci, de manière plus expressive que ne le permettait RDFS. OWL définit en effet de nombreuses relations supplémentaires et a été développé par le W3C afin de permettre aux applications de réaliser des raisonnements plus complexes sur les données ainsi représentées. Une extension de ce langage, nommée OWL2, est parue fin octobre 2009. Dans ce qui suit, OWL fera référence à la première version du langage OWL et OWL2 à son extension.

OWL définit les trois sous-langages OWL Lite, OWL-DL et OWL Full, allant du moins au plus expressif. OWL Lite permet d'établir une hiérarchie de classes et d'appliquer des contraintes simples comme par exemple une contrainte de cardinalité (minCardinality, maxCardinality) qui ne peut prendre que les valeurs 0 et 1. OWL-DL est destiné à permettre

une plus grande expressivité tout en restant un langage décidable. Enfin, OWL Full est encore plus expressif et permet de traiter une classe comme étant à la fois une collection d'individus et un individu à part entière, mais n'est pas un langage décidable.

Ce langage est basé sur la logique de description qui est un formalisme logique permettant la représentation de la connaissance. Une base de connaissance dans la logique de description est répartie en une TBox (Terminological Box) et une ABox (Assertionnal Box). La TBox contient des déclarations définissant des concepts, c'est-à-dire des classes d'éléments, et les relations, appelés rôles, entre ceux-ci. Un nouveau concept peut être défini sur base d'autres concepts de la TBox et il existe des concepts atomiques, c'est-à-dire qui ne sont pas définis sur base d'autres concepts. Ainsi, on peut définir le concept de femme comme une personne de sexe féminin avec la déclaration ci-dessous, utilisant les concepts Person et Female précédemment définis dans la TBox. L'ABox contient des informations sur les individus, appelles assertions. Par exemple, le fait Woman(Clara) fait partie de la ABox et permet d'indiquer que Clara est une femme.

Woman ≡ Person Female

OWL est construit comme une extension de RDFS et ajoute de nombreux prédicats, augmentant ainsi l'expressivité. Il est par exemple possible de définir deux classes comme étant équivalentes (prédicat owl:equivalentClass symbolisé par =) ou disjointes (owl:disjointWith), ou qu'une classe est l'union (unionOf), l'intersection (intersectionOf) ou même le complément (complementOf) d'une autre classe. La classe des rois belges peut être construite comme l'intersection de la classe des belges et de la classe des rois, comme indiqué ci-dessous.

BelgianKing = intersectionOf (Belgian, King)

De nouveaux prédicats sont également définis afin de caractériser les propriétés utilisées dans les ontologies. Une propriété peut par exemple être équivalente à une autre (equivalent- Property) ou l'inverse d'une autre propriété (inverseOf). L'exemple ci-dessous permet d'indiquer que la propriété kingOf est l'inverse de hasKing. De plus, il est possible de spécifier qu'une propriété est transitive (transitiveProperty), symétrique (SymmetricProperty) ou fonctionnelle (FunctionalProperty). Une propriété fonctionnelle est une propriété ne pouvant prendre qu'une seule valeur pour un sujet donné. Une telle propriété est par exemple la relation hasBirthDay, une personne n'ayant qu'une et une seule date de naissance.

kingOf inverseOf hasKing

hasbirthDay type FunctionnalProperty

OWL offre également la possibilité de restreindre le domaine et l'espace d'arrivée d'une propriété de manière locale, c'est-à-dire pour des instances d'une classe spécifique ayant cette propriété et non pour toutes les classes possédant la propriété concernée. L'exemple suivant permet d'indiquer que le prédicat hasResidence, appliqué à la classe BelgiumKing, ne peut prendre comme valeur que des instances de la classe Palace, imposant ainsi que les rois belges résident dans un palais. Cette restriction est locale dans le sens où elle n'impose pas que toutes les relations hasResidence aient pour valeur un palais et ne tient que lorsqu'elle est appliquée à un roi belge. Le prédicat someValuesFrom permet d'indiquer que le domaine ou l'espace d'arrivée d'une propriété doit avoir au moins une valeur appartenant à la classe ainsi spécifiée.

BelgiumKing hasResidence allValuesFrom(Palace)

Enfin, nous utiliserons par la suite la syntaxe OWL abstraite pour définir des ontologies. L'exemple suivant illustre une ontologie simple définissant une classe de rois belges comme étant l'intersection de la classe des belges et de celle des rois, que l'on suppose définies par ailleurs. Une restriction est appliquée à la relation kingOf, supposée définie comme ayant la relation hasKing pour inverse, pour spécifier qu'un individu de cette classe ne peut pas être roi de plus d'un pays. Cette classe possède également une relation hasResidence ne pouvant prendre comme valeur qu'une instance de la classe Palace, imposant qu'un roi belge doit vivre dans un palais. Enfin, deux individus sont définis, Albert II et la Belgique, que nous pourrions réécrire sous forme de triplets (AlbertII, type, BelgianKing), (Belgium, type, Country) et enfin (Belgium, hasKing, AlbertII).

Class (BelgianKing complete
    intersectionOf (Belgian King)
    restriction (kingOf maxCardinality(1))
    restriction (hasResidence allValues From(Palace)))
Individual (AlbertII type (BelgianKing))
Individual (Belgium type(Country)
    Value (hasKing AlbertII))

OWL2 est une extension du langage OWL publiée par le W3C fin octobre 2009. Cette extension ajoute de nouvelles fonctionnalités à OWL, parmi lesquelles la possibilité de définir des propriétés asymétriques, réflexives et disjointes. OWL2 permet également d'associer des clés à des instances de classes pour identifier ces individus de manière univoque. Deux individus ne peuvent alors posséder la même valeur de clé que s'ils sont égaux.

Trois profils sont également définis dans OWL2, à savoir OWL2 EL, OWL2 QL et OWL2 RL. Il s'agit de sous-langages d'OWL2 adaptés à certains types d'applications et consistant en différentes restrictions d'OWL2. OWL2 EL est adapté pour l'utilisation d'ontologies de grande taille mais relativement simples. OWL2 QL est prévu pour des applications visant à interroger un grand volume de données. Enfin, OWL2 RL est destiné aux applications nécessitant de grandes capacités de raisonnement tout en restant suffisamment expressif. Ces profils se distinguent par leur complexité de raisonnement et d'interrogation des ontologies réalisées dans ces sous-langages d'OWL2. **[1]**

I.4.4 SPARQL :

Les documents RDF peuvent être interrogés grâce au langage SPARQL (SPARQL Protocol and RDF Query Language) qui constitue la recommandation officielle du W3C. Ce langage permet l'interrogation des triplets des documents RDF et ne réalise aucune inférence.

Les requêtes de ce langage contiennent généralement un ensemble de modèles de triplets appelé modèle de graphe élémentaire (basic graph pattern en anglais). Ces modèles de triplets ressemblent à des triplets RDF, à ceci près que le sujet, le prédicat et l'objet peuvent être des variables. Un modèle de graphe élémentaire correspond à un sous-graphe du graphe RDF sur lequel s'applique la requête SPARQL lorsque les termes RDF de ce sous-graphe peuvent être substitués aux variables et le résultat est un graphe RDF équivalent à ce sous-graphe. Les variables de SPARQL sont identifiées par le préfixe ? ou $. Le modèle de graphe élémentaire est présent dans une clause WHERE et peut ainsi être vu comme une condition à satisfaire.

L'exemple suivant est une requête SPARQL permettant de retrouver tous les noms des personnes présentes dans les données RDF fournies. Le préfixe foaf est utilisé pour identifier la propriété de nom et est spécifié au début de la requête au moyen du mot clé PREFIX. Le résultat d'une telle requête est une séquence de solutions, comprenant de zéro à n solutions en fonction des données.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {?x foaf:name ?name. }
```

Ce type de requête, avec l'opérateur SELECT, est une requête dite interrogative dans la mesure où elle extrait du graphe RDF un sous-graphe correspondant à un modèle de graphe

élémentaire. Il existe également des requêtes constructives, exprimées avec l'opérateur CONSTRUCT, qui créent un nouveau graphe construit sur base d'un template et des triplets résultats de la correspondance avec le modèle de graphe élémentaire. La requête suivante construit un nouveau graphe par extraction des noms des employés et en utilisant la propriété foaf:name au lieu de la propriété org:employeeName. Ainsi, si le document RDF sur lequel la requête est exécutée comprend les triplets (A, org:employeeName, Alice) et (A, org:employeeId, 12345), le graphe RDF construit par la requête comprendra le nouveau triplet (A, foaf:name, Alice).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX org: <http://example.com/ns#>
CONSTRUCT {?x foaf:name ?name }
WHERE {?x org:employee Name ?name. }
```

Le langage SPARQL permet également de réaliser des correspondances en utilisant des littéraux ou des valeurs numériques dans le modèle de graphe élémentaire. Il est également possible de restreindre les résultats à certaines valeurs en utilisant des filtres. La requête suivante réalise un filtre sur le salaire d'un employé et renvoie un graphe RDF comprenant le nom et le salaire des employés.

```
PREFIX org: <http://example.com/ns#>
SELECT ?name ?salary
WHERE { ?x org:employee Name ?name. ?x org:salary ?salary.
FILTER (?salary > 10000) }
```

Enfin, notons qu'il est possible d'appliquer des modificateurs à la séquence de solutions obtenues. Cette séquence étant non ordonnée, le modificateur Order By permet de l'ordonner suivant une expression donnée. L'exemple ci-dessous ordonne les résultats suivant le nom des personnes. Le modificateur Distinct permet de s'assurer que les solutions de la séquence sont uniques tandis que le modificateur Limit est utilisé afin de restreindre le nombre de solutions.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
 SELECT  ?name
WHERE {?x foaf:name ?name. }
ORDER BY  ?name
```

I.5 Les outils:

De nombreux outils ont été développés afin d'utiliser les technologies du Web Sémantique et réaliser des applications se basant sur celles-ci. Une première collection de ces outils consiste en différents Framework destinés à faciliter le développement de telles applications à travers des librairies écrites en divers langages et réalisant la lecture et écriture de documents RDF, entre autres. Nous présentons ensuite les RDF stores, des bases de données spécialisées dans le stockage de triplets RDF. Nous détaillons ensuite le principe des raisonneurs indispensables à l'inférence de connaissance dans le Web Sémantique. **[1]**

I.5.1 Frameworks :

Afin de permettre un développement plus aisé d'applications liées au Web Sémantique, de nombreux frameworks ont été développés, tels que Jena, JRDF ou encore CubicWeb. Des librairies sont ainsi fournies afin de permettre la lecture de documents RDF, d'en extraire des triplets associés à une certaine propriété par exemple et également d'ajouter des triplets aux graphes RDF établis. Ces frameworks supportent également l'interrogation de ces documents au travers du langage SPARQL. Le framework Jena est réalisé en Java et fournit des raisonneurs, permettant ainsi de réaliser de l'inférence à partir de données RDF. Le JRDF a été développé dans ce même langage et ces deux frameworks supportent la sérialisation RDF/XML.

CubicWeb permet le développement d'applications dans le langage Python et utilise le langage RQL, similaire au langage SPARQL, afin de réaliser l'interrogation de graphes RDF.

I.5.2 RDF store :

Les RDF stores sont des bases de données de graphes RDF optimisées pour le stockage et la récupération de triplets RDF. Un RDF store peut ainsi contenir plusieurs millions de triplets RDF et permet d'extraire ces triplets de la base de données RDF, de réaliser des requêtes SPARQL sur le contenu de celles-ci ainsi que d'y ajouter des triplets. Parmi les RDF stores existants, citons AllegroGraph, Mulgara et Virtuoso. L'utilisation de bases de données permet de traiter plusieurs millions de triplets RDF, ce qui n'est pas possible avec le framework Jena par exemple qui conserve en mémoire l'ensemble des triplets RDF d'un graphe et se retrouve ainsi limité par la mémoire de la machine virtuelle Java. **[1]**

I.5.3 Raisonneurs :

Comme nous l'avons vu dans ce chapitre, les langages RDF Schema et OWL fournissent du support pour le raisonnement en définissant des hiérarchies de classes ainsi que des relations entre celles-ci et entre des propriétés. Il est par exemple possible de définir une propriété comme étant symétrique par exemple. De telles définition consistent en des règles d'inférence pouvant être appliquées par des logiciels appelés raisonneurs à la base de connaissance afin d'inférer automatiquement de la connaissance. Ainsi, si le prédicat kingOf est défini comme étant l'inverse de hasKing et que la base de connaissance contient le triplet (Baudouin, kingOf, Belgium), un raisonneur est capable d'enrichir cette base de connaissance avec le triplet (Belgium, hasKing, Baudouin).

Les principaux services d'inférence proposés par les raisonneurs sont repris ci-dessous.

- Vérification de la consistance : Un raisonneur est capable de s'assurer de la consistance d'une base de connaissance, autrement dit de vérifier qu'il n'y a pas de faits contradictoires dans la base de connaissance.
- Satisfaisabilité de concept : Il s'agit de déterminer si une classe peut posséder une instance. Une classe insatisfaisable entraîne une inconsistance si une instance d'une telle classe est définie. Un exemple de classe insatisfaisable est une classe définie comme l'intersection d'une classe Woman et d'une classe Father étant donné qu'une personne ne peut être à la fois une femme et un père.
- Classification : La classification consiste à établir les liens hiérarchiques entre toutes les classes d'une ontologie, c'est-à-dire identifier les différentes sous-classes de classes. Ceci permet par exemple de fournir toutes les sous-classes d'une classe donnée.
- Réalisation : La réalisation consiste à établir la classe la plus spécifique d'un individu, ce qui nécessite d'avoir réalisé une classification auparavant. Par exemple, s'il existe une classe Person et une classe Woman qui est une sous-classe de la classe Person, la classe la plus spécifique d'une femme sera la classe Woman et non la classe Person qui est plus générale.

Parmi les raisonneurs existants, citons RacerPro9, FaCt++10 et Pellet. Ce dernier, écrit dans le langage Java, est un raisonneur OWL-DL complet, c'est-à-dire qu'il supporte toute l'expressivité du langage OWL-DL. Il a de plus été étendu afin de supporter certaines propriétés apportées par OWL2, entre autres les assertions de négation de propriétés, les

propriétés disjointes et le punning qui est une technique permettant à un individu et une classe de partager la même URI. **[1]**

I.6 Exemples d'application du web sémantique :

Les technologies du Web sémantique sont de plus en plus appliquées à un large spectre d'applications au sein desquelles une connaissance de domaine est modélisée et formalisée (ontologie) afin de servir de support à des traitements très diversifiés (raisonnements) effectués par des machines. En outre, ces représentations peuvent être rendues compréhensibles par l'homme pour assurer un couplage optimal entre raisonnements humains (cognitifs) et mécaniques (sémantique formelle) confiant à l'homme et à la machine des tâches complémentaires. Pour citer quelques- unes de ces applications :

Application médicale :

La médecine est un des domaines d'applications privilégiés du Web sémantique comme elle l'a été, à une autre époque, des techniques de l'Intelligence artificielle, en particulier les systèmes experts. C'est en effet un domaine complexe où les informations à partager sont nombreuses et où il n'y a pas ou peu de solutions algorithmiques à ce partage comme à l'usage des connaissances, en particulier cliniques. Ainsi, un des principaux mécanismes du Web sémantique qui est la description de ressources via des annotations est de la plus grande importance en bio-informatique, plus particulièrement autour des questions de partage des ressources génomiques. Dans le contexte, plus ancien, de la recherche d'information, la médecine a une longue tradition de développement de thésaurus comme le MeSH (Medical Subject Heading) ou UMLS (Unified Medical Language System) et les utilise maintenant dans le cadre des mécanismes du Web sémantique. Enfin, et plus récemment, les services Web proposent des solutions à la problématique récurrente et non résolue de l'interopérabilité en médecine, en particulier dans le contexte des systèmes d'information hospitaliers (SIH).

Plusieurs d'autres applications ont été développées tel que ; Portails d'entreprises et Mémoire d'entreprises, E-Commerce, E-Work, Traitement Automatique des Langues et Traduction Automatique, Recherche d'Information, Intégration d'Entreprises et E-Work, Communautés d'Intérêts, Data Mining, etc.

Au carrefour d'une maturité technologie émergente et d'une pression économique pressentant des gains potentiels et l'élargissement ou la création de nouveaux marchés, se manifeste un intérêt croissant pour l'évaluation des technologies du Web sémantique sous l'angle des coûts et bénéfices mesurables qu'offre cette nouvelle technologie. Une première étape dans la mesure objective de l'intérêt de cette nouvelle technologie est d'en présenter simplement de premiers résultats pré-industriels pour des applications prototypes les plus prometteuses. **[7]**

II. les ontologies :

L'exploitation de connaissances en informatique a pour objectif de ne plus faire manipuler en aveugle des informations à la machine mais de permettre un dialogue (une coopération) entre le système et les utilisateurs. Alors, le système doit avoir accès non seulement aux termes utilisés par l'être humain mais aussi à la sémantique qui leur est associée, afin qu'une communication efficace soit possible. Actuellement, la connaissance visée par les ontologies est un sujet de recherche populaire dans diverses communautés (l'ingénierie des connaissances, la recherche d'information, le traitement du langage naturel, les systèmes d'information coopératifs, l'intégration intelligente d'information et la gestion des connaissances). Elles offrent une connaissance partagée sur un domaine qui peut être échangée entre des personnes et des systèmes hétérogènes. Elles ont été définies en intelligence artificielle afin de faciliter le partage des connaissances et leur réutilisation. La définition explicite du concept ontologie soulève un questionnement qui est tout à la fois d'ordre philosophique, épistémiologique, cognitif et technique. **[8]**

II.1 Notion d'ontologie :

L'ontologie est une branche de la métaphysique qui s'intéresse à l'existence, à l'être en tant qu'être et aux catégories fondamentales de l'existant. En effet, ce terme est construit à partir des racines grecques « ontos » qui veut dire ce qui existe, l'être, l'existant, et « logos » qui veut dire l'étude, le discours, d'où sa traduction par « l'étude de l'être » et par extension de l'existence.

Dans la philosophie classique, l'ontologie correspond à ce qu'**Aristote** appelait la Philosophie première (protè philosopha), c'est-à-dire la science de l'être en tant qu'être, par opposition aux philosophies secondes qui s'intéressaient, elles, à l'étude des manifestations de l'être (les existants). L'ontologie est une partie de la métaphysique qui s'attache à l'étude ou à la théorie de l'être dans son essence, indépendamment des phénomènes de son existence.

Dans le cadre de l'intelligence artificielle, **Neeches** et ses collègues furent les premiers à proposer une définition à savoir : « Une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire ».

En 1993, **Gruber** propose la définition suivante : « Spécification explicite d'une conceptualisation ». Cette définition a été modifiée légèrement par **Borst** comme « spécification formelle d'une conceptualisation partagée ». Ces deux dernières définitions sont regroupées dans celle de **Studer** comme « spécification formelle et explicite d'une conceptualisation partagée ».

- **Formelle** : l'ontologie doit être lisible par une machine, ce qui exclut le langage naturel.
- **Explicite** : la définition explicite des concepts utilisés et des contraintes de leurs utilisations.
- **Conceptualisation** : le modèle abstrait d'un phénomène du monde réel par identification des concepts clefs de ce phénomène.
- **Partagée** : l'ontologie n'est pas la propriété d'un individu, mais elle représente un consensus accepté par une communauté d'utilisateurs.

Pour **Guarino** & **Giaretta** « une ontologie est une spécification rendant partiellement compte d'une conceptualisation ». **Swartout** et ses collègues la définissent comme suit : « une ontologie est un ensemble de termes structurés de façon hiérarchique, conçue afin de décrire un domaine et qui peut servir de charpente à une base de connaissances ». La même notion est également développée par **Gomez** comme : « une ontologie fournit les moyens de décrire de façon explicite la conceptualisation des connaissances représentées dans une base de connaissances ». **[8]**

Gruber: « In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members) ».

Une ontologie est un ensemble structuré sous la forme d'un graphe orienté qui contient des nœuds représentant le vocabulaire d'un domaine particulier et des arcs représentant les relations nommées entre les concepts. **[9]**

II.2 Le cycle de vie d'une ontologie :

Puisque les ontologies sont destinées à être utilisées comme des composants logiciels dans des systèmes répondant à des objectifs opérationnels différents, leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. Ainsi, les ontologies doivent être considérées comme des objets techniques évolutifs et possédants un cycle de vie qui nécessite d'être précisé. Dans ce contexte, les activités liées aux ontologies sont, d'une part, des activités de gestion de projet (planification, contrôle, assurance qualité), et d'autre part, des activités de développement (spécification, conceptualisation, formalisation) ; s'y ajoutent des activités transversales de support telles que l'évaluation, la documentation, la gestion de la configuration. Un cycle de vie inspiré du génie logiciel est proposé dans la figure ci-dessous. Il comprend une étape initiale de détection et de spécification des besoins qui permet notamment de circonscrire précisément le domaine de connaissances, une étape de conception qui se subdivise en trois phases, une étape de déploiement et de diffusion, une étape d'utilisation, une étape incontournable, d'évaluation, et enfin, une sixième étape consacrée à l'évolution et à la maintenance du modèle. Après chaque utilisation significative, l'ontologie et les besoins doivent être réévalués et l'ontologie peut être étendue et, si nécessaire, en partie reconstruite. La validation du modèle de connaissances est au centre du processus et se fait de manière itérative.

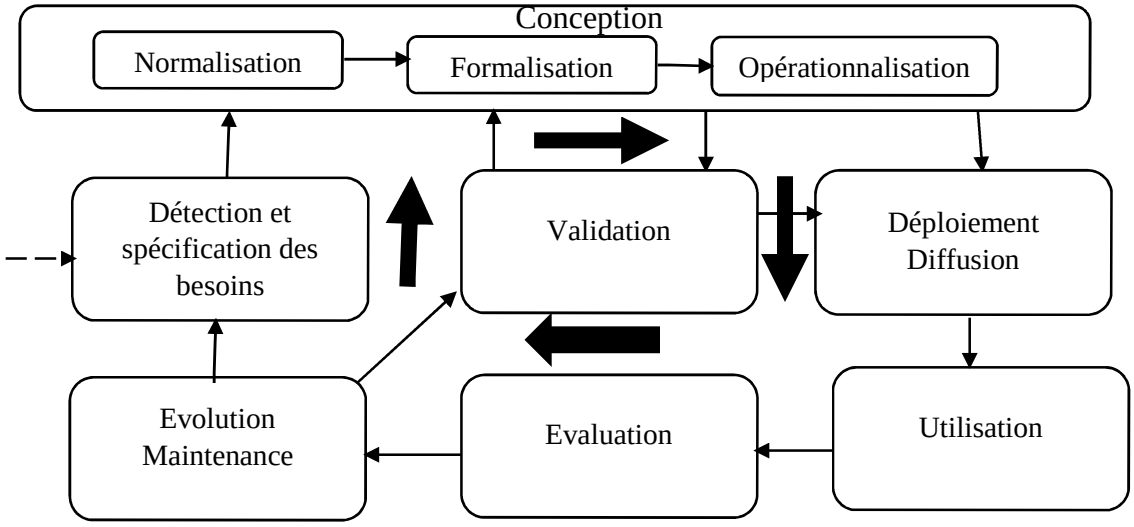

Figure 3 : Le cycle de vie d'une ontologie.

M. Fernandez insiste sur le fait que les activités de documentation et d'évaluation sont nécessaires à chaque étape du processus de construction, l'évaluation précoce permettant de limiter la propagation d'erreurs. Le processus de construction peut et doit être intégré au cycle de vie d'une ontologie. **[8]**

II.3 Types d'ontologie selon l'objet de conceptualisation :

On distingue six types d'ontologie :

II.3.1 Ontologie de représentation de connaissances :

Modélise les représentations primitives utilisées pour la formalisation des connaissances sous un paradigme donné. Par exemple, une ontologie sur le formalisme des Topic Maps comportera les concepts : Topic, Type de Topic, Association, Occurrence, Type Occurrence.

II.3.2 Ontologie de haut niveau / supérieure (Top-level / Upper-model) :

Elle exprime des conceptualisations valables dans différents domaines. Elle décrit des concepts très généraux comme l'espace, le temps, la matière, les objets, les évènements, les actions, etc. ces concepts ne dépendent pas d'un problème ou d'un domaine particulier, et doivent être, du moins en théorie, consensuels à de grandes communautés d'utilisateurs. Ce type d'ontologies est fondé sur la théorie de la dépendance. Son sujet est l'étude des catégories des choses qui existent dans le monde. Comme les concepts de haute abstraction tels que les entités, les évènements, les états, les processus, les actions, le temps, l'espace, les relations, les propriétés…

II.3.3 Ontologie Générique (Generic ontology) :

Elle est appelée également noyau ontologique, modélise des connaissances moins abstraites que celles véhiculées par l'ontologie de haut niveau mais assez générales néanmoins pour être réutilisées à travers différents domaines. Cette ontologie inclut un vocabulaire relatif aux choses, évènements, temps, espace, causalité, comportement, fonction, etc.

II.3.4 Ontologie du domaine (Domain ontology) :

Cette ontologie exprime des conceptualisations spécifiques à un domaine, elle est pour plusieurs applications de ce domaine. Elle fournit les concepts et les relations permettant de couvrir les vocabulaires, activités et théories de ces domaines. Selon **Mzoguchi**, l'ontologie du domaine caractérise la connaissance du domaine où la tâche est réalisée. Par exemple, dans le contexte du e-Learning, le domaine peut être celui de formation. C'est ce type d'ontologie qui nous sera utile pour définir les objectifs ciblés de notre projet.

II.3.5 Ontologie de Tâches (Task ontology) :

L'ontologie de tâches fournit un vocabulaire systématisé des termes employés pour résoudre des problèmes liés aux tâches qui peuvent être ou non du même domaine. Elle fournit un ensemble de termes au moyen desquelles nous pouvons décrire généralement comment résoudre un type de problèmes. Elle inclut des noms, des verbes et des adjectifs génériques dans les descriptions de tâches.

II.3.6 Ontologie d'application (Application ontology) :

C'est l'ontologie la plus spécifique, elle contient des concepts dépendants d'un domaine et d'une tâche particuliers, elle est spécifique et non réutilisable. Ces concepts correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine activité. Il s'agit donc ici de mettre en relation les concepts liés à une tâche particulière de manière à en décrire l'exécution. **[8]**

II.4 Les composantes d'une ontologie :

Comme nous l'avons abordé, les ontologies fournissent un vocabulaire commun d'un domaine et définissent la signification des termes et des relations entre elles. La connaissance dans les ontologies est principalement formalisée en utilisant les cinq types de composants à savoir : **concepts** (ou classes), **relations** (ou propriétés), **fonctions**, **axiomes** (ou règles) et **instances** (ou individus).

- Les concepts, aussi appelés termes ou classe de l'ontologie, correspondent aux abstractions pertinentes d'un segment de la réalité (le domaine du problème) retenus en fonction des objectifs qu'on se donne et de l'application envisagée pour l'ontologie ;
- Les relations traduisent les associations (pertinentes) existant entre les concepts présents dans le segment analysé de la réalité. Ces relations incluent les associations suivantes :
  - Sous classes de (généralisation-spécialisation) ;
  - Partie de (agrégation ou composition) ;
  - Associe à ;
  - Instance de, etc.

  Ces relations nous permettent d'apercevoir la structuration et l'interrelation des concepts, les uns par rapport aux autres ;
- Les fonctions constituent des cas particuliers de relations, dans laquelle un élément de la relation, (le nième) est défini en fonction des N-1 éléments précédents ;

- Les axiomes constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie.
- Les instances constituant la définition extensionnelle de l'ontologie ; ces objets véhiculent les connaissances (statiques, factuelles) à propos du domaine du problème. **[10]**

II.5 Méthodologies de construction :

Il n'existe pas qu'un seul mode ou qu'une seule méthode correctes pour développer des ontologies. Il existe plus de 33 méthodes de l'ingénierie ontologique. Il n'y a évidemment pas une méthode qui soit la meilleure. La conception des ontologies est un processus créatif et il ne peut pas y avoir d'ontologies identiques faites par des personnes différentes. Les applications potentielles d'une ontologie et la compréhension du concepteur, ainsi que le point de vue qu'il a du domaine traité, affecteront indubitablement les choix de conception de l'ontologie. « C'est à l'usage que l'on juge ».

Nous pouvons tester la qualité de notre ontologie uniquement en l'utilisant dans les applications pour lesquelles elle a été conçue, le développement d'une ontologie est nécessairement un processus itératif et les concepts dans une ontologie doivent être très proches des objets (physiques ou logiques) et des relations de domaine d'intérêt. Fort probablement ils sont des noms (objets) ou des verbes (relations) dans des phrases qui décrivent le domaine. Plusieurs méthodologies ont été décrites dans la littérature comme par exemple : la méthode Methontology, la méthode de Bernaras et al «1996 »**,** La méthode de Bachimont, La méthode de Uschold et King, etc….

II.5.1 La Méthode d'Uschold et King :

Uschold et King ont proposé la première méthode de construction d'ontologie qui est basé sur l'expérience acquise lors du développement de l'ontologie d'Entreprise « the enterprise ontology », cette méthode est composée de quatre étapes :

1. Identification des objectifs et du contexte de l'ontologie : Le but de cet étape est de clarifier le pourquoi de la construction de l'ontologie, les utilisations prévues et la finalité (réutiliser, partager, etc.) et les utilisateurs potentiels de l'ontologie.

2. Construction de l'ontologie : Cette étape est divisée en trois activités :

- **Capture de l'ontologie :** se fait indépendamment d'un langage de représentation (conceptualisation). Elle commence par l'identification des concepts et des relations clés ensuite produire en langage naturel les définitions précises et non ambiguës pour les concepts et les relations et se termine par l'identification des termes dénotant les concepts

et les relations pour ainsi essayer d'arriver à un agrément. Pour la catégorisation et la capture de l'ontologie, Uschold et Grüninger proposent trois approches :

a. Approche descendante *(Top Down)* : l'ontologie est construite par généralisation en partant des concepts des basses couches taxinomiques. Cette approche encourage la création d'ontologies spécifiques et adaptées.

b. Approche ascendante *(Buttom Up)* : l'ontologie est construite par spécialisation en partant de concepts des hautes couches taxinomiques. Cette approche encourage la réutilisation d'ontologies.

c. Approche intermédiaire *(Middle Out)* : l'ontologie est construite à partir de concepts centraux puis les généralise ou les spécialise pour former les couches hautes et les couches basses de l'ontologie. Cette approche encourage l'émergence de domaines thématiques dans l'ontologie et favorise la modularité.

- **Codage de l'ontologie :** Cette étape implique deux tâches :

a. Représentation explicite de la conceptualisation (ex. classe, entité, relation).
b. Ecriture du code dans un langage formel : Prolog, KL-One, OIL, CG, OWL…

- **Intégration d'ontologies existantes :** Cette étape fait référence à comment utiliser les ontologies qui existent. Elle peut être faite en parallèle avec les étapes de capture et/ou de codage. Par exemple ; The Frame Ontology peut être réutilisée pour modéliser les ontologies de domaine qui utilise l'approche basée sur les frames.

3. Evaluation de l'ontologie : c'est la mise à l'épreuve de l'ontologie en la faisant confronter aux objectifs pour lesquels elle a été conçue et aux utilisateurs.

4. Documentation de l'ontologie : cette étape est essentielle pour l'acceptation de l'ontologie. **[9]**

II.5.2 La méthode de Bachimont :

Cette méthode contraint l'utilisateur à un engagement sémantique en introduisant une normalisation sémantique des termes manipulés dans l'ontologie. La méthode de normalisation suit trois étapes :

1. **Normalisation sémantique** : l'utilisateur doit choisir les termes du domaine et les normaliser en explicitant leurs propriétés et en exprimant les identités et les différences dans leur voisinage proche. La place d'une notion dans l'ontologie doit être justifiée par rapport à la communauté et la différence avec le père et la fratrie.

2. **Formalisation des connaissances** : Cette étape consiste à désambiguïser les notions de l'ontologie référentielle obtenue par l'étape précédente et choisir leurs sens pour un domaine spécifique. Cela peut nécessiter la création de nouveaux concepts, l'ajout de propriétés et d'axiomes.

3. **Opérationnalisation des connaissances** : Le système utilise un langage opérationnel de représentation de connaissances qui possède les caractéristiques nécessaires pour répondre aux besoins exprimés lors de la spécification du système. **[8]**

II.6 Les outils de construction d'ontologie :

On distingue deux familles d'outils : les outils de construction d'ontologie dépendants de formalisme de représentation et les outils de construction d'ontologie indépendants de formalisme de représentation.

II.6.1 Les outils dépendants de formalisme de représentation :

II.6.1.1 Ontolingua : Ontolingua est un serveur d'édition d'ontologies. Il utilise des classes, des relations, des fonctions, des instances et des axiomes pour décrire une ontologie. Une relation peut contenir des propriétés nécessaires (contraintes) ou nécessaires et suffisantes qui définissent la relation. En plus le serveur Ontolingua offre la possibilité d'intégrer les ontologies Ontolingua, ce qui permet la construction modulaire des ontologies.

II.6.1.2 OntoSaurus : OntoSaurus de l'Information Science Institute de l'Université de Southern California est composé de deux modules : un serveur utilisant LOOM comme langage de représentation des connaissances, et un serveur de navigation créant dynamiquement des pages HTML qui affichent la hiérarchie de l'ontologie ; le serveur utilise des formulaires HTML pour permettre à l'usager d'éditer l'ontologie.

II.6.1.3 WebOnto : WebOnto du Knowledge Media Institute de l'Open University, est une application Web pour naviguer et développer collaborativement les ontologies. Il supporte la navigation collaborative, la création et l'édition d'ontologies sur le Web. Les ontologies WebOnto sont implémentées dans le langage OCML. WebOnto distingue quatre types d'ontologies : ontologie de domaine, ontologie de tâche, ontologie de méthode, et ontologie d'application.

II.6.1.4 OilEd : OilEd (Oil Editor) est un éditeur d'ontologies utilisant le formalisme OIL. Il est essentiellement dédié à la construction de petites ontologies dont on peut ensuite tester la cohérence à l'aide de FACT, un moteur d'inférences bâti sur OIL.

II.6.2 Les outils indépendants de formalisme de représentation :

II.6.2.1 Protégé 2000 : est une interface modulaire permettant l'édition, la visualisation, le contrôle d'ontologie, l'extraction d'ontologies à partir de sources textuelles, et la fusion semi-automatique d'ontologies. Le modèle de connaissances sous-jacent à protégé 2000 est issu du modèle des frames et contient des classes, des slots (propriétés) et des facets (valeurs des propriétés et contraintes), ainsi que des instances des classes et des propriétés. Il autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie.

II.6.2.2 ODE et WebOde : L'outil ODE (Ontology Design Environment) permet de construire des ontologies au niveau connaissance, comme le préconise la méthodologie Methontology. L'utilisateur construit son ontologie dans un modèle de type frame, en spécifiant les concepts du domaine, les termes associés, les attributs et leurs valeurs, les relations de subsomption.

II.6.2.3 OntoEdit : OntoEdit (Ontology Editor) est également un environnement de construction d'ontologies indépendant de tout formalisme. Il permet l'édition des hiérarchies de concepts et de relations et l'expression d'axiomes algébriques portant sur les relations, et de propriétés telles que la généricité d'un concept. Des outils graphiques dédiés à la visualisation d'ontologies sont inclus dans l'environnement. OntoEdit intègre un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs. Un contrôle de la cohérence de l'ontologie est assuré à travers la gestion des ordres d'édition. **[10]**

II.7 Les domaines d'applications des ontologies :

II.7.1 Système d'information :

L'ontologie retrouve maintenant dans une large famille de systèmes d'information. Elle est utilisée pour :

- Décrire et traiter des ressources multimédia ;
- Assurer l'interopérabilité d'applications en réseaux ;
- Piloter des traitements automatiques de la langue naturelle ;
- Construire des solutions multilingues et interculturelles ;
- Permettre l'intégration des ressources hétérogènes d'information ;
- Vérifier la cohérence de modèles ;
- Permettre les raisonnements temporel et spatial ;
- Faire des approximations logiques ; etc.

Ces utilisations des ontologies se retrouvent dans de nombreux domaines d'applications tel que :

- Intégration d'information géographique ;
- Gestion de ressource humaine ;
- Aide à l'analyse en biologie, suivi médicale informatisé ;
- Commerce électronique ;
- Enseignement assisté par ordinateur ;
- Bibliothèque numériques ;
- Recherche d'informations.

II.7.2 Web sémantique :

Un courant particulièrement prometteur pour l'expansion des systèmes à base d'ontologies est celui du Web sémantique. Grace à ce dernier, l'ontologie a trouvé un jeu de formalismes standards à l'échelle mondiale, et s'intègre dans de plus en plus d'applications Web, sans même que les utilisateurs ne le sachent. Cela se fait au profit des logiciels qui à travers les ontologies et les descriptions qu'elles permettent, peuvent proposer de nouvelles fonctionnalités exploitant les effets d'échelles du Web pour en améliorer les effets. **[10]**

II. 8 Le passage du modèle UML vers une ontologie OWL :

Un diagramme de classes UML modélise une vue de conception statique d'un système représentant principalement le vocabulaire du domaine étudié. Le passage vers une ontologie OWL permet de formaliser des connaissances exprimées de façon semi-formelle en une ontologie OWL formelle, le modèle UML est traduisible vers une ontologie OWL, pour le transformer nous allons suivre le processus ACM décrit ci-dessous.

II.8.1 La transformation d'une ontologie semi formelle en une ontologie formelle :

Il existe un ensemble de critères et de principes qui ont fait leurs preuves dans la transformation des ontologies d'un modèle à un autre par exemple La transformation selon l'Architecture Conduite par les Modèles (ACM).

II.8.1.1 Architecture conduite par les modèles (ACM) :

Proposée par l'*Object Management Group* (OMG), l'Architecture Conduite par les Modèles (ACM) *"Model Driven Architecture*" offre un cadre méthodologique et architectural de développement de logiciels qui vise à découpler le développement des spécificités métiers

des contraintes technologiques liées à l'implantation de la solution. Les concepts de *modèle*, *méta-modèle* et *méta-méta-modèle* composent la structure de l'ACM :

- ***Un modèle :*** est une abstraction qui, selon un point de vue spécifique, représente une partie de la réalité. Le modèle se construit à partir d'éléments qui répondent à une syntaxe et à une sémantique cohérente formant un système de représentation des connaissances.
- ***Un méta-modèle :*** est une abstraction qui sert à définir la syntaxe et la sémantique d'un système ou langage de modélisation.
- ***Un méta-méta-modèle :*** est une abstraction qui sert à définir la syntaxe et la sémantique d'un système de représentation de méta-modèles. Le trait particulier d'un méta-méta-modèle est qu'il assure la comparaison et l'interopérabilité entre les différents méta-modèles, facilitant ainsi la transposition entre des modèles qui n'ont pas le même système de représentation. C'est le cas lorsqu'on désire transposer un modèle semi-formel représenté en langage semi formel, en un modèle formel représenté en OWL. **[20]**

II.8.1.2 Les niveaux d'abstraction de l'ACM :

Le cadre de l'ACM, tel que défini par l'OMG, est divisé en quatre niveaux :

- Le **niveau M0 :** contient les instances du modèle de niveau M1. Chacune des instances est le représentant d'un objet de la réalité observable.
- Le **niveau M1 :** est le modèle de la couche M0 exprimé dans le système de représentation UML. Chaque élément du langage UML est une instance du méta-modèle du niveau M2.
- Le **niveau M2 :** ou méta-modèle exprime la syntaxe et la sémantique du système de représentation de la couche M1. Chacun des éléments du méta-modèle est une instance du niveau M3.
- Le **niveau M3 :** ou méta-méta-modèle est le modèle du niveau M2. Il permet de définir la structure syntaxique et sémantique de tous les méta-modèles du niveau M2 et donc de tout langage ou système de représentation. **[20]**

II.8.3 Principe de processus de transformation d'un model semi-formel en une ontologie formelle selon ACM :

- **Principe de transformation parallèle :** Le principe de transformation parallèle a comme pré-condition que le modèle source et le modèle cible possèdent une structure de méta-modèle similaire permettant la conception de règles de transformation de type "un à un". Par exemple, la méta-classe *Concept* du méta-modèle UML peut être transposée en méta-

classe *Class* dans le méta-modèle OWL ou encore la méta-classe *LienS,* de type relation, est transposée en méta-classe *subClassOf*. **[20]**

- **Principe de transformation orthogonale :**

Dans le but spécifique de considérer la sémantique du modèle source lors de la transformation, l'idée maîtresse du principe de la transformation orthogonale est de considérer les niveaux architecturaux du modèle semi-formel en tant que constituant de la réalité observable. Le *modèle du méta-modèle source* (l'ontologie du méta-modèle semi-formel) est un *modèle du modèle source* (l'ontologie du modèle semi-formel), ainsi que le modèle semi-formel (le modèle source du domaine à représenter), servent d'intrants au processus de transformation. Une particularité intéressante de ce principe de transformation est que le modèle semi-formel du domaine est représenté en tant qu'élément factuel de l'ontologie du modèle semi-formel et que la sémantique du modèle semi-formel (le modèle source) est représentée par l'ontologie du méta-modèle semi-formel. Cette particularité de combinaison des éléments sémantiques et factuels permet de produire des règles de transformation qui sont appliquées à partir de déductions et d'inférences conceptualisées par le modèle source, permettant la conception d'un système expert à la transformation.

Dans notre cas nous allons suivre le processus de transformation parallèle selon ACM et le tableau suivant présente quelques règles de transformation :

UML	RFD(S), OWL
Class	owl : Class.
Generalization	rdfs:subClassOf.
Association	owl : ObjectProperty.
Attributes	owl:DatatypeProperty
InstanceOf	rdf : type.
Multiplicity	owl:minCardinality, owl :maxCardinality.
Attribute type : String	rdfs:Literal
Attribute value	rdf:value

Tableau 2 : Correspondances utilisées entre les méta-modèles UML et RDF Schema, OWL.**[19]**

Le lien de composition appartenant à la sémantique de l'UML, n'a pas de représentation directe en OWL. La solution choisie est de créer la propriété *estComposéDe (*ComposedOf) au concept source et au concept destinataire désigné par le lien de composition.

Conclusion :

Nous avons présenté, dans cette section, en premier lieu ; les principaux langages développés par le W3C dans le cadre du Web Sémantique. Le Resource Description Framework permet de structurer l'information en la représentant sous forme de triplets (sujet, prédicat, objet), les ressources ainsi décrites étant de plus identifiées univoquement par un Uniform Resource Identifier. Le RDF Schema est une extension du RDF apportant du raisonnement en permettant de décrire une hiérarchie de ressources et de propriétés. Ce langage permet par exemple d'inférer le fait que la Belgique est une région, sachant qu'il s'agit d'un pays et qu'un pays est une sous-classe d'une région. Le Web Ontology Langage va plus loin dans cette capacité de raisonnement en ajoutant de nombreux prédicats tels que le fait que deux propriétés soient inverses l'une de l'outre ou qu'une classe corresponde à l'intersection de deux autres classes. Nous avons également présenté le langage SPARQL qui a été développé par le W3C afin de pouvoir interroger les documents RDF. Des outils destinés au développement d'applications liées au Web Sémantique ont également été détaillés, tels que les librairies utilisant ces langages du Web Sémantique, les RDF store ou encore les raisonneurs destinés à inférer de l'information structurée dans ces langages. En deuxième lieu nous avons rappelé la notion des ontologies, leurs différents types et composantes, et quelques outils de manipulation. A cet effet, nous allons présenter dans le chapitre suivant l'approche par compétences, qui permet de représenter les compétences à acquérir dans une discipline donnée qui est modélisée par une ontologie de domaine.

Introduction :

Les sciences de l'éducation puisent leurs fondements théoriques, entre autres, dans la psychologie, la sociologie, la philosophie et les sciences cognitives. Cette diversité de champs théoriques à la base des différentes approches de l'enseignement et de l'apprentissage peut parfois être confondante dans la mesure où certains auteurs peuvent se retrouver à l'intérieur de plus d'un courant théorique. Actuellement, une majorité de théoriciens en éducation s'accordent pour regrouper les modèles de l'enseignement et de l'apprentissage selon quatre courants: le courant béhavioriste, le courant cognitiviste, le courant constructiviste et le courant socioconstructiviste.

Le tableau ci-dessous présente un résumé schématique des quatre principaux courants en les reliant aux conceptions de l'acte d'enseigner et d'apprendre qui leur correspondent.

Socio-constructiviste	**Constructiviste**	**Cognitiviste**	**Béhavioriste**
Enseigner c'est…			
Organiser des situations d'apprentissage propices au dialogue en vue de provoquer et de résoudre des conflits sociocognitifs.	Offrir des situations obstacles qui permettent l'élaboration de représentations adéquates du monde.	Présenter l'information de façon structurée, hiérarchique, déductive.	Stimuler, créer et renforcer des comportements observables appropriés.
Apprendre c'est…			
Co-construire ses connaissances en confrontant ses représentations à celles d'autrui.	Construire et organiser ses connaissances par son action propre.	Traiter et emmagasiner de nouvelles informations de façon organisée.	Associer, par conditionnement, une récompense à une réponse spécifique.
Méthodes pédagogiques appropriées			
Apprentissage par projets, discussions, exercices, travaux.	Apprentissage par problèmes ouverts, étude de cas.	Exposé magistral, résolution de problèmes fermés.	Programme d'autoformation assistée par ordinateur.

Tableau 3 : Représentation schématique des principaux courants théoriques. **[11]**

La figure ci-dessous offre un aperçu global de l'évolution chronologique des courants théoriques et permet d'en identifier les auteurs respectifs les plus influents.

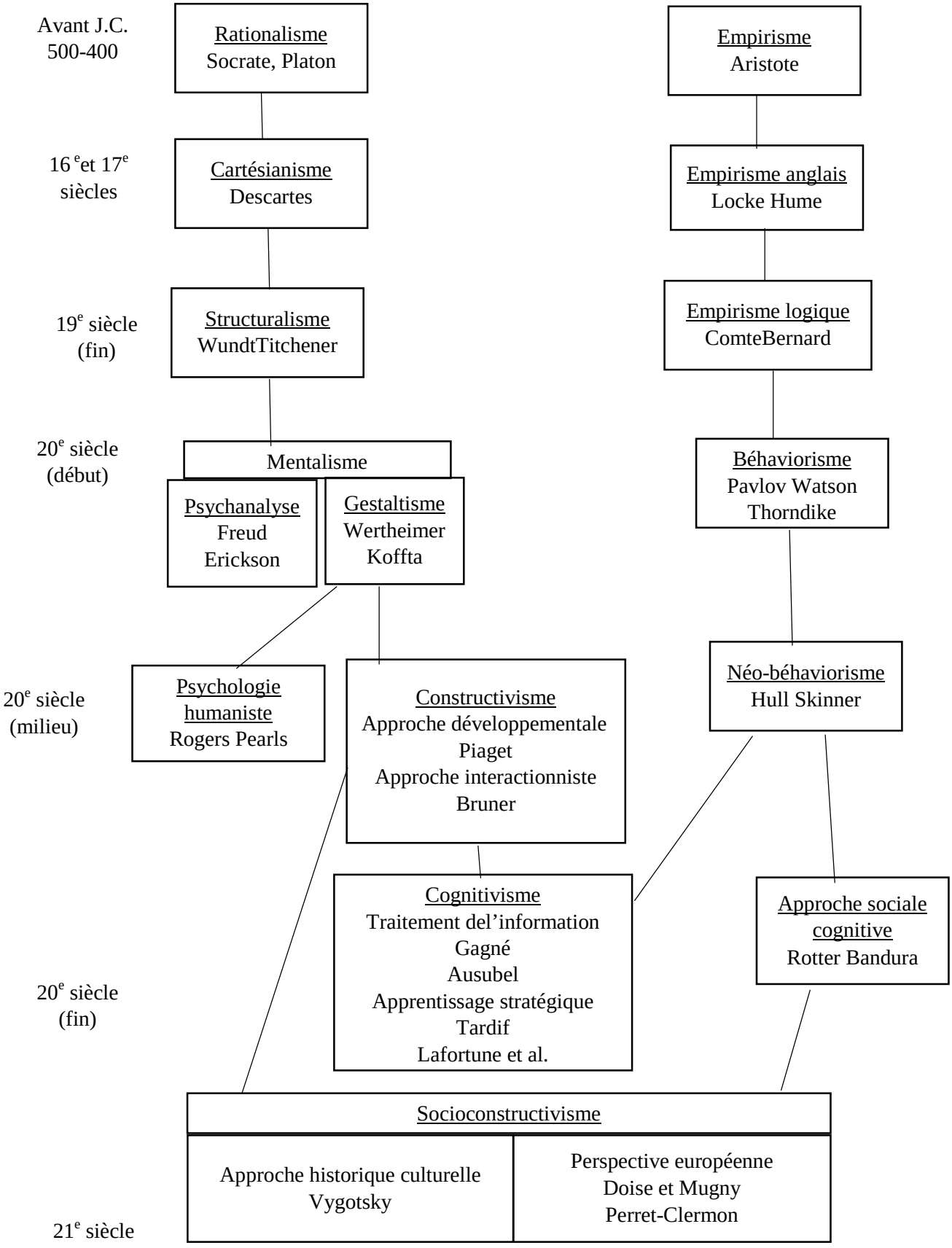

Figure 4 : Historique et évolution des courants théoriques de l'apprentissage. **[11]**

I. Les principaux courants théoriques de l'apprentissage :

I.1 Le behaviorisme :

Le behaviorisme est la première grande théorie de l'apprentissage à avoir fortement marqué les domaines de l'éducation, de l'enseignement et de la formation. Ce courant théorique qui a largement dominé les recherches en psychologie durant la première moitié du 20e siècle, exerce encore aujourd'hui une influence très forte, notamment dans les pays anglo-saxons. Avec le behaviorisme terme créé en 1913 par l'américain **Watson** à partir du mot behavior signifiant comportement, la psychologie est devenue la science du comportement. Le comportement dont il est ici question n'est pas une attitude ou une manière d'être de l'élève ; c'est le sens usuel du mot quand on dit qu'il doit améliorer son comportement. Il s'agit de la manifestation observable de la maîtrise d'une connaissance, celle qui permettra de s'assurer que l'objectif visé est atteint.

Le behaviorisme n'a pas très bonne presse car il est souvent réduit au conditionnement, avec le fameux schéma [Stimulus → Repense] issu des travaux de Pavlov. Mais le behaviorisme n'en est pas resté à ce mécanisme d'apprentissage primaire. De là sont issus, notamment, le conditionnement répondant, l'enseignement programmé, une bonne part de la pédagogie par objectifs (PPO) et de l'enseignement assisté par ordinateur (EAO) ainsi que le développement actuel des référentiels de compétences et de la pédagogie de maîtrise. **[12]**

La force du behaviorisme a été de proposer une théorie complète de l'apprentissage :

- En le définissant : apprendre c'est devenir capable de donner la réponse adéquate,
- En précisant les mécanismes psychologiques à l'œuvre : répétition de l'association stimulus- réponse,
- En proposant une méthode d'enseignement-apprentissage : opérationnaliser des objectifs d'apprentissage, conditionner, apprendre par essais-erreurs, provoquer des renforcements positifs en cas de bonnes réponses, et des renforcements négatifs pour rectifier les erreurs.

I.2 Le cognitivisme :

Le cognitivisme a pour objet d'étude la connaissance, la mémoire, la perception et le raisonnement, et regroupe différents modèles de l'enseignement et de l'apprentissage. Le terme vient du latin cognitio, qui signifie connaissance. Le point de départ du cognitivisme est la réintroduction de l'étude des phénomènes mentaux, frappée d'ostracisme par les béhavioristes. L'approche cognitive, caractérisée par son opposition au béhaviorisme radical de **Skinner**, revendique donc l'accès aux processus cognitifs internes. Cette rupture avec les

conceptions béhavioriste a permis l'élaboration du courant cognitiviste qui se prolonge dans deux versions de la psychologie cognitive. La première emprunte beaucoup à la représentation des opérations qui se déroulent dans un ordinateur et assimile l'esprit humain à un système de traitement de l'information. La deuxième est fondée sur l'importance de l'appropriation graduelle et effective de stratégies mentales (stratégies cognitives et métacognitives) jugées nécessaires à une démarche structurée d'apprentissage. Ces deux versions du cognitivisme ont vu le jour en s'opposant aux idées reçues des béhavioristes. **[11]**

I.3 Le constructivisme :

La perspective constructiviste est rattachée essentiellement à son représentant le plus célèbre **J. Piaget**. Cette perspective peut se replacer dans le cadre plus général du cognitivisme. Cette théorie de l'apprentissage développe l'idée que les connaissances se construisent par ceux qui apprennent. Pour le constructivisme, acquérir des connaissances suppose l'activité des apprenants, activité de manipulation d'idées, de connaissances, de conceptions. Activité qui vient parfois bousculer, contrarier les manières de faire et de comprendre qui sont celles de l'apprenant. L'individu est donc le protagoniste actif du processus de connaissance, et les constructions mentales qui en résultent sont le produit de son activité. Pour **Piaget**, celui qui apprend n'est pas simplement en relation avec les connaissances qu'il apprend ; il organise son monde au fur et à mesure qu'il apprend, en s'adaptant. Cette perspective constructiviste insiste sur la nature adaptative de l'intelligence, sur la fonction organisatrice, structurante qu'elle met en œuvre. Cette capacité d'adaptation s'appuie sur deux processus d'interaction de l'individu avec son milieu de vie : l'assimilation et l'accommodation. **[12]**

I.4 Le socioconstructivisme :

Par rapport au constructivisme, l'approche sociocognitive ou socioconstructive introduit une dimension supplémentaire : celle des interactions, des échanges, du travail de verbalisation, de co- construction, de co-élaboration. Cette idée de base transparaît dans bon nombre de tires d'ouvrages d'aujourd'hui : *interagir et connaître, on n'apprend pas tout seul, interagir pour apprendre, etc.*

L'apprentissage est alors davantage considéré comme le produit d'activités sociocognitives liées aux échanges didactiques enseignant - élèves et élèves - élèves. Dans cette perspective, l'idée d'une construction sociale de l'intelligence est prolongée par l'idée d'une auto-socio-construction des connaissances par ceux qui apprennent. Dans le cadre

socioconstructiviste, les conditions de mise en activité des apprenants sont essentielles, car ce qui se joue dans les apprentissages ce n'est pas seulement l'acquisition de connaissances nouvelles ou la restructuration de connaissances existantes ; c'est également le développement de la capacité à apprendre, à comprendre, à analyser ; c'est également la maîtrise d'outils. Ce n'est donc plus seulement par ce que l'enseignant transmet, et par les formes de mise en activité des élèves confrontés à des situations problèmes, que les élèves apprennent. C'est par des mises en interactivité (entre élèves et entre enseignant et élèves) que le savoir se construit.**[12]**

Dans ce chapitre nous allons s'intéresser à l'approche par compétences car notre objectif est de pouvoir manipuler une ontologie de domaine de compétences disciplinaires.

II. De l'approche par les objectifs à l'approche par les compétences :

II. 1 L'approche par les objectifs :

L'approche pédagogique adoptée pour le domaine de l'enseignement, fut pendant longtemps la **pédagogie par objectif** (PPO). Les recherches actuelles s'orientent vers l'**approche pédagogique par les compétences** (APC). La PPO a pour objectif de répondre à la question « que doit **savoir** et **savoir-faire** un apprenant à la fin d'un cycle scolaire. Elle permet de décomposer un savoir à transmettre aux apprenants en autant d'objectifs à atteindre. L'inconvénient majeur de cette approche, est que l'apprenant apprend par petits morceaux sans en comprendre le sens et sans savoir quel lien a son apprentissage avec la vie extrascolaire de tous les jours. C'est à ces questions essentielles que l'APC tente d'apporter des solutions, en ouvrant une brèche pour mettre en exergue et cristalliser le **capital-savoir** des apprenants durant les **situations d'apprentissage**. **[13]**

II.2 Principes de l'approche par compétences :

Dans les années 80 et avant son apparition dans le domaine scolaire, l'APC était adoptée dans les formations professionnelles visant à perfectionner les compétences de personnels et améliorer leur productivité. C'est une méthodologie ciblée dans la mesure où elle fixe un référentiel de compétences à atteindre vers la fin de la formation dans un poste de travail bien déterminé.

Partant de ce principe (un référentiel de compétences) l'APC fut adoptée dans le domaine de l'enseignement et elle est de plus en plus admise dans les systèmes éducatifs. A nos jours on parle de compétences noyaux en Suisse, Canada et en France et les socles de compétences pour l'enseignement fondamental et les compétences terminales et savoirs requis

pour l'Humanité générales et technologiques en Belgique et les compétences de base en Mauritanie, Djibouti et en Tunisie…

Dans le processus Enseignement/Apprentissage l'approche permet à l'élève d'acquérir des compétences durables susceptibles de l'aider dans son parcours éducatif et dans sa vie quotidienne. Elle met l'accent sur tous ce qui est fondamental afin de garantir une meilleure transmission des savoirs. L'APC devient donc la base pédagogique de tous les constituants de l'enseignement.

Les actions et les réflexes de l'apprenant deviennent la principale source de son apprentissage, elle vise à mettre l'apprenant dans le centre du processus éducatif pour lutter contre son échec.

L'approche par compétences était l'objet de plusieurs travaux élaborés par les didacticiens tel, **Philippe Perrenoud** qui suppose que pour garantir la bonne pratique de cette approche dans les systèmes éducatifs il faut rénover et réécrire les programmes pour qu'il ait une cohérence entre les intentions (les objectifs) et leurs mises en œuvre (la pratique).

II.3 Définition de l'approche par les compétences :

Les changements sociopolitiques et économiques effectués au niveau mondial, sont les premiers facteurs dus à la réforme actuelle qui s'oriente vers une approche par compétences (APC), après avoir longtemps utilisée une pédagogie par objectifs (PPO). L'APC s'est alors imposée progressivement dans les programmes scolaires de nombreux pays. Elle permet de passer d'une **pédagogie centrée sur les savoirs** (savoirs scolaires), à une **pédagogie centrée sur des activités** dans lesquelles ces savoirs s'incarnent. Ces savoirs sont en construction dynamique et en permanence recombinés par l'élève en vue d'un apprentissage de savoir-faire voire de savoir-réfléchir, de démarches plutôt qu'un apprentissage de connaissances (ou contenu).

Cependant, chaque texte officiel portant sur l'APC, adopte une définition de la **notion de compétence** et **une formulation en termes de compétences** de la manière de concevoir des enseignements qui lui sont propres. Néanmoins malgré la pluralité des définitions de la notion de compétence, deux courants de pensées émergent :

Dans le premier courant, les auteurs réduisent la formulation des compétences à des **habilités** sommaires (un verbe d'action et un objet), portant strictement sur des contenus notionnels reliés à des savoirs disciplinaires traditionnels (Joannert, 2011). Ils considèrent les compétences comme un regroupement d'objectifs spécifiques (opérationnels) (Rogiers, 2006). Dans le second courant, la vision des auteurs est de rendre les apprentissages actifs en mettant

l'accent sur le développement de **situations d'apprentissage** en remplacement de leçons magistrales axées sur le discours de l'enseignant (Rogiers, 2006). Il s'agit dans cette vision, de rendre l'apprenant acteur de ses apprentissages, et l'enseignant comme un guide, un orienteur et un facilitateur de l'apprentissage.

La notion de compétences ici réfère aux résultats des **actions de personnes** traitant des **situations** (Joannert, 2011 ; Rogiers, 2006) appartenant à des **familles de situations**. En effet, dans l'approche par situation (APS), le développement des compétences et connaissances se déploie en contexte, dans et par l'action. Une situation dans ce cas, se définit comme un ensemble contextualisé d'informations fournies à des apprenants, pour être articulé en vue de réaliser une tâche précise. Cette notion de situation adoptée dans la plupart des travaux, est désignée selon le cas par **situation problème** ou encore **situation tâche** (Scallon, 2004).

Une situation problème (SP) est une tâche concrète adaptée aux élèves, imaginée par un enseignant comme un espace de réflexion autour d'une question ou d'un problème à résoudre. Un problème se définit par la présence d'un obstacle à franchir, des informations à articulées et de la tâche à réaliser. Afin d'accomplir cette tâche, l'enseignant est supposé choisir un ensemble de ressources (savoirs, savoir- faire et savoir-être) et de contraintes (matériels et humaines) en fonction des compétences souhaitées ; à développer chez les élèves. Ces conditions d'exécution sont sensés aider les élèves à franchir des obstacles, incontournables au premier abord. Le franchissement de ces obstacles doit se faire au prix d'un apprentissage, qu'il s'agisse d'un simple transfert**,** d'une généralisation ou de la construction d'un savoir entièrement nouveau. En effet, la SP doit être conçue de telle façon à permettre l'auto-construction de savoirs par les élèves et d'enrichir à terme leurs connaissances de nouvelles représentations(ou conceptions), donc d'apprendre.

Cependant pour mieux piloter l'apprentissage des élèves, ces derniers doivent être observés et leurs compétences évaluées. L'observation porte sur les actions pédagogiques qu'ils effectuent durant leur apprentissage. L'évaluation est généralement menée durant l'apprentissage, en tenant compte des obstacles et des critères (et/ ou normes) d'évaluation arrêtés. Elle a pour but d'établir des bilans périodiques de compétences et prendre des décisions de progression. Elle permet en quelque sorte d'apprécier les actions pédagogiques des élèves mais aussi lever ou contourner les obstacles éventuels à l'apprentissage. Affronter les obstacles, revient à affronter, l'absence de toute solution, voire de toute piste ou de toute méthode de la part des élèves. Mais, une fois le problème dévoilé, les élèves se l'approprient, leur esprit échafaude des hypothèses, procède à des explorations, propose des essais, etc. Dans

un cadre de travail collectif, une discussion s'engage et, à chacun de préciser sa pensée, voire ses représentations tout en tenant compte de celles des autres.

Nous tenons à signaler que la notion d'obstacle serait décrite chez certains comme une maladresse, une ignorance, un problème de niveau, une question d'aptitude, une qualité de l'enseignement des cycles antérieurs (Madrane, 2007). Pour d'autres, un obstacle serait un signe de difficultés ou d'erreurs (Deval, 2000). Une prise en compte sérieuse et/ou sens large de cette notion dans toute entreprise de projet d'enseignement/ apprentissage est souhaitable car, elle conditionne grandement la qualité de l'enseignement et de l'apprentissage. **[13]**

II.4 L'évaluation selon l'approche par les compétences :

Dans l'optique des compétences, évaluer consiste à proposer une ou des situations complexes, appartenant à la famille de situations définie par la compétence, qui nécessiteront, de la part de l'élève, une production elle-même complexe pour résoudre la situation. Les épreuves élaborées à des fins d'évaluation seront inévitablement différentes. Néanmoins, les questions qui se posent par rapport à ces épreuves sont les questions classiques relatives à tout recueil d'information : sont-elles pertinentes, valides et fiables? **[14]**

D'une part, une évaluation valide est une évaluation qui mesure effectivement ce qu'elle prétend mesurer. Une évaluation peu valide serait par exemple de corriger uniquement l'orthographe des étudiant pour une dissertation par laquelle on voudrait vérifier leurs capacités à articuler des idées par écrit et à argumenter leur opinion. D'autre part, une stratégie d'évaluation fiable est une stratégie qui permet d'évaluer toujours la même chose de la même façon : que l'on corrige des travaux à 7h ou à 19h ou que l'étudiant soit évalué par l'enseignant A ou l'enseignant B. Une façon de diminuer la subjectivité de l'évaluation est d'utiliser des grilles d'évaluation critériées. Le principe d'une grille d'évaluation est d'établir une correspondance entre d'une part le résultat d'une tâche que l'on demande à un étudiant (un texte, une explication orale, une action précise dans un contexte professionnel, etc.) et d'autre part des critères de qualité de ce résultat fourni par l'étudiant accompagnés ou non d'une échelle à plusieurs degrés de performance.

L'usage des grilles d'évaluation se fait généralement pour une évaluation sommative (en fin d'apprentissage) mais d'autres usages sont tout à fait possibles bien sûr, comme pour une auto-évaluation, une évaluation par les pairs ou comme fiche de feedback à remettre aux étudiants après une présentation orale par exemple. **[15]**

L'évaluation des élèves par critères signifie que, partout dans le monde, les élèves sont jugés en fonction de critères d'évaluation prédéterminés pour chaque groupe de matières. Les enseignants déterminent les tâches d'évaluation qui seront utilisées en interne, à savoir au sein de l'établissement. Pour s'assurer que les normes sont appliquées de manière cohérente à travers le monde, ces évaluations internes font l'objet de vérifications externes (révision de notation ou service-conseil sur l'évaluation) réalisées par des examinateurs. **[16]**

Un **critère** est une qualité attendue de la production ou de la prestation de l'étudiant. Sa formulation doit donc exprimer cette qualité. Les critères doivent être pertinents, indépendants et peu nombreux.

- Ils sont pertinents s'ils mettent en évidence les ressources (savoirs, savoir-faire et attitudes) acquises par l'étudiant et s'ils permettent de vérifier son aptitude à intégrer et à organiser ses ressources afin d'accomplir la tâche qui lui est imposée.
- Ils sont indépendants quand l'échec ou la réussite pour un critère n'entraine pas automatiquement l'échec ou la réussite pour un ou plusieurs autres critères.
- Ils sont peu nombreux pour éviter de « parcelliser » exagérément l'évaluation. **[17]**

II.4.1 La construction de grille critériée :

Une grille critériée est un tableau qui détaille à la fois les critères utilisés pour interpréter la preuve d'apprentissage fournie par l'étudiant dans un travail écrit ou encore lors d'une présentation orale, et les indicateurs ou niveaux de performance possibles pour chaque critère. Il faut faire attention à ne pas confondre grille critériée avec grille de correction. La première est utile pour des méthodes d'évaluation qui demandent à l'étudiant de fournir passablement d'informations, desquelles est extraite la preuve d'apprentissage (par exemple, dissertation, exposé oral, réponses à développement). La seconde, qui est plutôt une liste de réponses acceptées, est davantage utile pour des méthodes d'évaluation qui ne demandent à l'étudiant que de choisir la bonne réponse ou encore de fournir quelques informations. La construction d'une grille critériée comporte un certain nombre d'étapes. Voici un sommaire de ces étapes :

- Identifier et décrire concrètement les critères d'évaluation, ceux-ci étant généralement étroitement liés aux objectifs d'apprentissage ;
- Établir les niveaux de réussite, ceux-ci pouvant varier de deux (par exemple, pour les évaluations de type succès/échec) à plusieurs niveaux, sachant que plus le nombre de niveaux est grand, plus les différences entre ceux-ci s'amenuisent ;

- Choisir le type de grille pour le rendu, soit une grille détaillant tous les niveaux, soit une grille qui ne détaillerait qu'un niveau mais qui permettrait à l'enseignant de fournir plusieurs commentaires justifiant la note allouée ;
- Tester la grille en l'utilisant pour quelques copies ou encore en l'utilisant dans le cadre de présentations orales à visées formatives (feedback) plutôt que sommatives (jugement final) ; revoir les critères qui ne semblent pas pertinents, ajouter ceux qui prennent forme lors du test ; modifier le nombre de niveaux de performance ou ajuster les descripteurs si des problèmes surviennent lors du test.

Ce qui est particulièrement intéressant avec les grilles critériées, au plan de l'apprentissage, c'est que celles-ci peuvent être utilisées à divers moments d'un enseignement, pas seulement lors de l'évaluation des apprentissages. Ainsi, puisque les critères décrits dans la grille sont en lien direct avec les objectifs d'apprentissage du cours en question, il est possible de faire appel à la grille lors d'exercices complétés en classe ou hors de la classe. Les étudiants peuvent même être appelés à s'autoévaluer à l'aide de la grille, ou encore à évaluer leurs pairs, en cours de semestre, lors d'évaluations formative. **[18]**

II.4.2 La stratégie d'évaluation de la grille critériée :

Pour faciliter l'évaluation, tout travail d'un atelier devrait comprendre un nombre raisonnable d'éléments d'évaluation. Suivant sa complexité et sa longueur, un travail devrait avoir entre 5 et 15 éléments d'évaluation, chacun couvrant un aspect particulier du devoir. Le type des éléments d'évaluation dépend de la stratégie d'évaluation désirée. Cette stratégie est semblable à celle du niveau d'atteinte global des objectifs à la différence qu'elle comporte plusieurs critères.

- Le nombre de critères est déterminé dans les paramètres de l'atelier.
- Pour chaque critère, il peut y avoir jusqu'à cinq énoncés décrivant les niveaux atteints.
- Ce nombre de niveaux peut varier d'un critère à l'autre.
- Un niveau laissé vide par l'enseignant signale que le niveau précédent est le dernier.
- Les critères peuvent être pondérés.
- Les niveaux sont marqués par les chiffres : 0, 1, 2, 3 et 4. Ces chiffres correspondent à la note accordée à chaque niveau.
- La note finale pour ce travail est la somme pondérée des notes pour chaque critère**. [17]**

Conclusion :

Dans ce chapitre nous avons présenté les différentes théories d'apprentissage où nous avons concentré sur l'approche par compétences et l'évaluation selon cette approche. Dans le cadre de notre travail nous avons besoin d'une ontologie de domaine de compétences disciplinaires qui a comme termes : Compétence, Situation Problème, Evaluation critèriée, Obstacle et Consigne, cette ontologie sera exploitée et manipulée dans les chapitres suivants.

Introduction :

Dans le cadre de notre travail nous utilisons une ontologie de domaine de compétences disciplinaires qui sera éditée, manipulée et exploitée à l'aide des technologies du web sémantique, pour cela nous supposons une liste de compétences disciplinaires et un ensemble de situations problème (SPs) dans lesquelles les compétences vont se développer moyennant un seuil donné. Les SPs sont des tâches à accomplir dans une discipline donnée, adaptées aux profils d'apprenants (chacun sa progression, ses acquis antérieurs, ses expériences, etc.). Chaque SP d'un degré de complexité et d'une durée donnée, est accompagnée de consignes et de critères d'évaluation des actions des apprenants. Les consignes sont des guides sous forme d'énoncés qui indiquent le but à atteindre ainsi que la démarche pouvant l'atteindre. Nous supposons qu'en réalisant une SP, un apprenant peut s'affronter à des obstacles spécifiques (propres à la discipline) et/ou généraux (liés à plusieurs disciplines). Les obstacles sont considérés dans notre cas, comme des difficultés ou encore des manques de la part des apprenants (ex. non maitrise d'une notion ou d'une technique) pouvant entraver leur apprentissage. A chaque SP, une cartographie mettant en évidence les interdépendances entre les compétences (compétences composées et/ou élémentaires, pré-requises) à développer durant la résolution de la SP, est associée. Cette cartographie est indispensable pour deux raisons : elle permet d'organiser et d'identifier les compétences à acquérir selon la dynamique de progression de l'apprenant et facilite son évaluation formative. **[13]**

I. La modélisation de domaine des compétences à l'aide d'un diagramme UML :

Afin d'identifier et de définir le vocabulaire utilisé (concepts, relations entre concepts, propriétés, les restrictions sur les attributs, etc.) pour le domaine de compétences nous avons utilisé le diagramme de classes UML (Unified Modeling Language) qu'est un formalisme graphique utilisé pour formaliser des ontologies conceptuelles.

Il n'était pas possible de savoir dès le départ, que les termes collectés du modèle UML de l'ontologie, sont suffisants pour répondre à l'objectif pour lequel l'ontologie a été construite, on a ajouté des nouveaux termes lorsque c'était nécessaire, tout de même on a éliminé des termes qu'on a jugés inutiles. Dans ce chapitre nous allons transformer le modèle UML en une ontologie OWL en utilisant l'éditeur d'ontologie Protégé ensuite nous allons générer une base de connaissances des compétences pour le module de base de données par l'instanciation des différents concepts constituant l'ontologie.

II. La génération de la base de connaissances à partir de l'ontologie de domaine de compétences :

L'objectif d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné, d'associer et de relier les différents concepts de ce domaine, et tout cela de manière compréhensible par les machines. Les ontologies décrivent généralement :

- Individus : les objets de base
- Classes : ensembles, collections, ou types d'objets
- Attributs : propriétés, fonctionnalités, caractéristiques ou paramètres que les objets peuvent posséder et partager
- Relations : les liens que les objets peuvent avoir entre eux
- Evénements : changements subis par des attributs ou des relations.

Dans notre cas nous allons générer deux bases de connaissances ; la première basée sur l'ontologie modélisant le domaine de compétences et la deuxième basée sur une autre ontologie qui modélise le domaine des apprenants. Les différents concepts constituant l'ontologie de domaine de compétences sont présentés dans les tableaux ci-dessous.

II.1 Définition des classes et sous classes de l'ontologie de domaine de compétences :

Les classes du diagramme UML	**Description**	**Correspondance en OWL**
Compétence disciplinaire	La Compétence disciplinaire est une composante appartenant à un haut niveau d'abstraction. Elle est composée d'un ensemble de compétences composées	<owl:Class rdf:ID=" Competence_disciplinaire ">
Compétence composée	La compétence composée est une compétence intermédiaire ; compose la compétence disciplinaire et composée d'un ensemble de compétences élémentaires.	<owl:Class rdf:ID=" Competence_composee ">
Compétence élémentaire	La compétence élémentaire est la composante la plus élémentaire du domaine à enseigner, c'est une «granule» de la matière à	<owl:Class rdf:ID=" Competence_elementaire ">

	enseigner.	
Situation problème	Englobe les éléments d'évaluation comme les exercices, questions et étude de cas	<owl:Classrdf:ID="Situation_probleme">
Obstacle	L'obstacle est lié aux situations problème	<owl:Classrdf:ID=" Obstacle ">
Obstacle disciplinaire	L'obstacle disciplinaire est un obstacle propre à la discipline	<owl:Classrdf:ID="Obstacle_disciplinaire">
Obstacle général	L'obstacle général est lié à plusieurs disciplines	<owl:Classrdf:ID=" Obstacle_general">
Consigne	C'est une illustration et aide pour la résolution d'une SP	<owl:Classrdf:ID="Consigne">
Evaluation critèriée	Englobe l'ensemble des critères utilisés pour évaluer une compétence élémentaire donnée	<owl:Classrdf:ID="Evaluation_criteriee">

Tableau 4 : Classes et sous classes de l'ontologie de domaine de compétences.

II.2 Définition des attributs (Datatypeproperties) des classes de l'ontologie de domaine de compétences :

Attributs	**Classe**	**Type**	**Correspondance en OWL**
Id_CpD	Compétence disciplinaire	Entier	<owl:DatatypeProperty rdf:ID="Id_CpD">
Nom_CpD	Compétence disciplinaire	Chaîne de caractères	<owl:DatatypeProperty rdf:ID=" Nom_CpD ">
Id_CpC	Compétence composée	Entier	<owl:DatatypeProperty rdf:ID="Id_CpC">
Nom_CpC	Compétence composée	Chaîne de caractères	<owl:DatatypeProperty rdf:ID="Nom_CpC">
Desc_CpC	Compétence composée	Chaîne de caractères	<owl:DatatypeProperty rdf:ID=" Desc_CpC ">
Id_CpE	Competence élémentaire	Entier	<owl:DatatypeProperty rdf:ID=" Id_CpE ">
Nom_CpE	Competence élémentaire	Chaîne de caractères	<owl:DatatypeProperty rdf:ID=" Nom_CpE ">
Desc_CpE	Competence élémentaire	Chaîne de caractères	<owl:DatatypeProperty rdf:ID=" Desc_CpE ">
Id_SP	Situation problème	Entier	<owl:DatatypePropertyrdf:ID="Id_SP ">

Enoncé_SP	Situation problème	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="Enoncé_SP ">
URL_SP	Situation problème	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="URL_SP ">
Id_Obs	Obstacle	Entier	<owl:DatatypePropertyrdf:ID="Id_Obs ">
Nom_Obs	Obstacle	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="Nom_Obs ">
URL_Obs	Obstacle	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="URL_Obs ">
Id_ObD	Obstacle disciplinaire	Entier	<owl:DatatypePropertyrdf:ID="Id_ObD ">
Nom_ObD	Obstacle disciplinaire	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="Nom_ObD ">
Desc_ObD	Obstacle disciplinaire	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="Desc_ObD ">
Id_ObG	Obstacle général	Entier	<owl:DatatypePropertyrdf:ID="Id_ObG ">
Nom_ObG	Obstacle général	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="Nom_ObG ">
Desc_ObG	Obstacle général	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="Desc_ObG ">
Id_Cn	Consigne	Entier	<owl:DatatypePropertyrdf:ID="Id_Cn ">
Enoncé_Cn	Consigne	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="Enoncé_Cn ">
URL_Cn	Consigne	Chaîne de caractères	<owl:DatatypePropertyrdf:ID="URL_Cn ">
Degré_Per_max	Evaluation critèriée	Entier	<owl:DatatypePropertyrdf:ID="Degré_Per_max ">
Degré_Imp	Evaluation critèriée	Entier	<owl:DatatypePropertyrdf:ID="Degré_Imp ">

Tableau 5 : Les attributs des classes de l'ontologie de domaine de compétences.

II.3 Définition des relations (Object Properties) reliant les classes de l'ontologie de domaine de compétences :

Relation	**Classe source**	**Classe cible**	**cardinalité**	**Correspondance en OWL**
Prérequis	Compétence disciplinaire	Compétence disciplinaire	0..n	<owl:ObjectProperty rdf:ID="Prerequis">
Est_composé_de1	Compétence composée	Compétence composée	0..n	<owl:ObjectProperty rdf: ID="Est_compose_de1">

Est_composé_de2	Compétence composée	Compétence élémentaire	1..n	<owl:ObjectProperty rdf:ID="Est_compose_de 2">
Evalué_par	Compétence élémentaire	Situation problème Evaluation critèriée	1..n	<owl:ObjectProperty rdf:ID="Evalue_par">
Est_défini1	Situation problème	Compétence élémentaire	1..n	<owl:ObjectProperty rdf:ID="Est_defini1">
Possède2	Situation problème	Consigne	1..n	<owl:ObjectProperty rdf:ID="Possede2">
Est_défini2	Consigne	Situation problème	1..n	<owl:ObjectProperty rdf:ID="Est_defini2">
Evite	Consigne	Obstacle	0..n	<owl:ObjectProperty rdf:ID="Evite">
Est_evité	Obstacle	Consigne	0..n	<owl:ObjectProperty rdf:ID="Est_evite">
A_obstacle_ disciplinaire	Compétence disciplinaire	Obstacle disciplinaire	0..n	<owl:ObjectProperty rdf:ID="A_obstacle_ disciplinaire">
Est_relié2	Obstacle disciplinaire	Compétence disciplinaire	1..n	<owl:ObjectProperty rdf:ID="Est_relie2">
Possède1	Situation problème	Obstacle	0..n	<owl:ObjectProperty rdf:ID="Possede1">
Est_relié1	Obstacle	Situation problème	1..n	<owl:ObjectProperty rdf:ID="Est_relie1">
Prérequis	Compétence disciplinaire	Compétence disciplinaire	0..n	<owl:ObjectProperty rdf:ID="Prerequis">

Tableau 6 : Les relations reliant les classes de l'ontologie de domaine de compétences.

II.4 Les règles d'inférences associées à l'ontologie de domaine de compétences :

Semantic Web Rule Language (SWRL) est une expression OWL à base de langage de règles. SWRL permet aux utilisateurs d'écrire des règles qui peuvent être exprimées en termes de concepts OWL pour fournir plus de capacités de raisonnement déductif qu'OWL seul. Sémantiquement, SWRL est construit sur le même fondement logique de description qu'OWL et offre de solides garanties formelles similaires lors de l'exécution des inférences.

Une règle SWRL contient une partie antécédente qui est désignée comme le corps et une partie conséquente qui est désignée comme le responsable :

atome ^ atome .... → atome ^ atome

Officieusement, une règle SWRL peut être lue comme signifiant que si tous les atomes dans l'antécédent sont vrais, alors le conséquent doit également être vrai. Un atome est une

expression de la forme : p (arg1, arg2, ... argn) où p est un symbole de prédicat et arg1, arg2, ..., argn sont les termes ou les arguments de l'expression. Dans SWRL, les symboles de prédicat peuvent inclure des classes OWL, les propriétés ou les types de données. Les arguments peuvent être des individus OWL ou des valeurs de données, ou les variables qui s'y réfèrent. Les arguments sont indiqués en utilisant la convention standard ; les préfixant avec un point d'interrogation. **[21]**

Dans notre cas, nous avons définis les règles suivantes pour l'ontologie modélisant le domaine de compétences :

1. si X est une situation problème et Y est une compétence élémentaire et Z est une compétence composée, si X est défini pour Y et Z est composé de Y alors X est une situation problème de Z.
2. si X est une compétence composée et Y est une autre compétence composée et Z est une compétence élémentaire, si X est composé de Y et Y est composé de Z alors X est composé de Z.
3. si X est une compétence composée et Y est une autre compétence composée et Z est une compétence élémentaire, si X est prérequis de Y et Y est composé de Z alors X est prérequis de Z.
4. si X est un obstacle disciplinaire et Y est une compétence disciplinaire et Z est une compétence composée, si X est relié à Y et Y est composé de Z alors X est relié à Z.

La correspondance de ces règles en SWRL sous Protégé est décrite comme suit :

1. Situation problème(?X) ^ compétence élémentaire(?Y) ^ compétence composée(?Z) ^ est défini pour(?X, ?Y) ^ est composée de (?Z, ?Y) ⟶ est défini pour(?X, ?Z).

2. Compétence composée(?X) ^ Compétence composée(?Y) ^ compétence élémentaire(?Z) ^ est composée de(?X, ?Y) ^ est composée de(?Y, ?Z) ⟶ est composé de(?X, ?Z).

3. Compétence composée(?X) ^ Compétence composée(?Y) ^ compétence élémentaire(?Z) ^ prérequis(?X, ?Y) ^ composé de(?Y, ?Z) ⟶ prérequis(?X, ?Z).

4. Obstacle disciplinaire(?X) ^ Compétence disciplinaire(?Y) ^ compétence composée(?Z) ^ est relié(?X, ?Y) ^ est composé de(?Y, ?Z) ⟶ est relié(?X, ?Z).

La deuxième ontologie que nous avons généré est l'ontologie modélisant le domaine des apprenants contient deux classes ; la classe apprenant caractérisée par un numéro, nom, prénom, préférence et style. La classe performance caractérisée par le code-compétence et le degré d'acquisition, cette ontologie est éditée pour la génération du plan guide adaptatif qui permet de déduire les compétences acquises par un apprenant donné.

III. Génération de la base de connaissance de domaine de compétences pour la discipline de base de données :

Après avoir créé des différents concepts de l'ontologie (classes, liens, propriétés,… etc.) nous passons à la dernière étape qui est l'instanciation.

III.1 L'instanciation de classe compétences disciplinaires, compétences composées et compétences élémentaire :

La discipline choisie comme exemple à implémentée dans notre application est le module de base de données pour lequel nous avons instancié trois compétences disciplinaires ; seize compétences composées et plus de cent compétences élémentaires.

Compétences générales	**Compétences composées**	**Compétences élémentaires**
1. Conception des bases de données	1. Introduction aux BDD	1. Définition d'une BDD 2. Types de BDD
	2. Démarche de conception des BDD	3. Elaboration du modèle E/A (niveau conceptuel) 4. Passage au modèle relationnel (niveau logique ou organisationnel) 5. Implémentation sur un SGBD-R cible (niveau physique)
	3. Modélisation conceptuelle (entités- associations)	6. Les entités 7. Les propriétés 8. Types des associations 9. Les cardinalités 10. L'identifiant 11. Le schéma conceptuel
	4. Modélisation logique relationnelle	12. Définition d'une relation 13. Attributs 14. Degré d'une relation 15. Clé d'une relation 16. Le schéma d'une relation 17. Le domaine 18. Occurrence d'une relation 19. Cardinalité d'une relation 20. Les n-uplets 21. Contrainte de domaine 22. Contrainte de clé 23. Contrainte d'intégrité référentielle ou d'inclusion
	5. Les dépendances fonctionnelles	24. Définition d'une DF 25. Clé d'une relation 26. Les propriétés des DFs 27. Une DF élémentaire 28. Une DF canonique 29. Une DF directe 30. Graphe de DFs 31. La fermeture transitive 32. La couverture minimale

	6. La normalisation de relations	33. La première forme normale 34. La deuxième forme normale 35. La troisième forme normale 36. La forme normale de Boyce et Codd
	7. Traduction d'un schéma conceptuel en schéma relationnel	37. Nom de la table 38. Clé de la table 39. Autres attributs de la table 40.Traduction de relation binaire plusieurs à plusieurs 41. Traduction de relation binaire un à plusieurs 42. Traduction de relation binaire un à un
2. Manipulation des bases de données	8. l'algèbre relationnel	43. Principe des opérateurs unaires 44. Opérateur de la sélection 45. Opérateur de complément 46. Opérateur de la projection 47. Opérateur de l'affectation 48. Opérateur de l'auto Jointure 49. Principe des opérateurs binaires 50. Opérateur de l'union 51. Opérateur de l'intersection 52. Opérateur de différence 53. Opérateur de produit cartésien 54. Opérateur de Théta-produit 55. Opérateur de la jointure naturelle 56. Opérateur de la jointure extérieure 57. Opérateur de la semi jointure
	9. Expression des requêtes avec le langage des expressions algébriques	58. Expression des opérateurs unaires 59. Expression de la sélection 60. Expression du complément 61. Expression de la projection 62. Expression des symboles binaires 63. Expression d'union 64. Expression d'intersection 65. Expression de la différence 66. Expression du produit cartésien 67. Expression du théta-produit 68. Expression de la jointure naturelle 69. Expression de la jointure extérieure 70. Expression de la semi jointure gauche 71. Expression de la semi jointure droite.
	10. Représentation de graphique avec le formalisme des arbres	72. Principe des symboles unaires 73. Formalisme de la sélection 74. Formalisme du complément 75. Formalisme de la projection 76. Principe des symboles binaires 77. Formalisme d'union 78. Formalisme d'intersection 79. Formalisme de la différence 80. Formalisme du produit cartésien 81. Formalisme du théta-produit 82. Formalisme de la jointure naturelle 83. Formalisme de la jointure

		extérieure 84. Formalisme de la semi jointure gauche 85. Formalisme de la semi jointure droite.
	11. représentation de requêtes avec SQL	86. Définition du langage de requêtes algébriques SQL 87. Création d'une base de données 88. Création d'une table 89. Suppression d'une table 90. Modification d'une table 91. Renommer une table 92. La commande create 93. La commande alter 94. La commande rename 95. La commande drop 96. La commande select 97. La commande insert 98. La commande update 99. La commande delete 100. La clause unique 101. La clause not null 102. La clause primary key 103. La clause references 104. L'expression foreign key 105. La clause CHECK
3. Administration des bases de données (maitrise des fonctions des SGBD)	12. Evaluation et optimisation des requêtes	106. Optimisation algébrique 107. Optimisation par une fonction de coût. ..
	13. La transaction	108. Définition de la transaction 109. Propriétés d'une transaction ..
	14. Gestion des droits d'accès	110. La commande grant 111. La commande revoke 112. Contraintes d'intégrité
	15. gestion des accès concurrents	113. Journalisation des transactions. 114. Le Verrouillage 115. Les types de verrous
	16. Reprise après panne	116. Reprise à chaud 117. Reprise à froid ..

Tableau 7 : Les instances de la classe compétences disciplinaires, compétences composées et compétences élémentaires. [**22][23]**

III.2 L'instanciation de la classe situation problème :

Une situation problème est utilisée pour contrôler l'assimilation d'une ou plusieurs compétences élémentaires constituant une compétence disciplinaire, pour chaque compétence élémentaire on peut trouver plusieurs situations problème comme on peut y avoir une

situation problème qui définit plusieurs compétences élémentaires, le tableau suivant renferme certaines situations problème.

Situations problème	**Compétence générale évaluée**	**Compétence composée évaluée**	**Compétence élémentaire évaluée**
Un éditeur souhaite installer une BDD pour mémoriser les informations suivantes : Les livres sont identifiés par leur numéro ISBN. Un livre possède un titre et un prix de vente. Il est écrit par un ou plusieurs auteurs, chaque livre est tiré en une ou plusieurs éditions, datées et identifiées par leur ordre. Chaque édition comporte un certain nombre d'exemplaires. Les auteurs sont identifiés par leur nom, prénom et peuvent avoir un pseudonyme. Pour chaque livre, un auteur perçoit des droits d'auteur, calculés comme pourcentage du prix de vente. Les librairies identifiées par leur nom et adresse ; peuvent envoyer des commandes d'un ou plusieurs livres en quantité quelconque. Elaborer le schéma entité-association pour la BDD de la maison d'édition ?	Conception des bases de données	Modèle entité-association	Les entités, les propriétés, types des associations, les cardinalités, l'identifiant et le schéma conceptuel.
Soit la base de données représentée par les relations suivantes : Boutique (NomB, VilleB, QuartierB) Client (NomC, rueC, villeC) Employé (NomE, NomB, salaireE) Achat (NomC, NomB, NomE, montant, date, heure) 1. Désigner la clé primaire, la(es) clé(s) secondaire (s) éventuelle (s) et la ou les clés étrangères pour chaque relation ? 2. Formuler les expressions algébriques pour répondre aux requêtes suivantes : – La liste des boutiques du quartier « Champlain » – La liste des noms des boutiques de la ville Sainte-Foy – Nom des clients de la rue René-levesque et qui sont employé d'une boutique – Afficher les noms des boutiques qui ont vendu quelque chose – La liste des noms des clients qui n'ont jamais fait d'achats à la ville Sainte-Foy. 3. Représenter chacune des requêtes ci haut avec un arbre algébrique.	Conception des bases de données Manipulation des bases de données	Modélisation logique relationnelle Algèbre relationnelle . Représentation graphique des opérateurs	Définition d'une relation, attributs, clé d'une relation, le schéma d'une relation, La projection, la sélection, l'intersection, le produit cartésien. Principe des symboles unaires, Formalisme de la sélection, Formalisme de la projection, Principe des symboles

			binaires Formalisme du produit cartésien.
Soit le schéma relationnel suivant relatif à une base de données d'un centre de recherche : Personne (CodeP, nom ; prénom ; statut, grade, NumE). Equipe (NumE, nomE, datecreat, CodeP). DomaineRech (NumD, designation). Projet (NumPro, NomPro, debut, duree, coderesp, NumD, nomE, montant, date). Contrat (CodeC, debutC, montant, NomPro). Financement (CodeC, organisme, apport). Développe (NumE, NomPro). Participe (CodeP, NomPro, tauxPart). Publication (NumPub, titre, date, NumD, NomPro). Auteur (CodeP, NumPub). Exprimer les requêtes suivantes en SQL : 1. Quels sont les noms des domaines de recherche du laboratoire 2. Quels sont les noms et prénoms des membres de l'équipe numéro 1 3. Quels sont les noms et prénoms des responsables de l'équipe numéro 1 4. Donner les noms des personnes qui ont le même grade que leur responsable de l'équipe	Manipulation des bases de données	langage relationnel SQL	Définition du langage de requêtes algébriques SQL, La commande Select.
Soit R(A, B, C, D) avec F={A→B, B→C, C→D}. Est-ce que A→D ? Trouver F+ la fermeture transitive ?	Conception des bases de données	Modélisation logique relationnelle Les dépendances fonctionnelles	Les attributs, Relations ou tables, Le schéma d'une relation. Définition d'une DF, Clé d'une relation, Propriétés des DFs, Une DF élémentaire, Une DF canonique, Une DF directe, Graphe de DFs, La fermeture transitive.

Soit R(A, B, C, D) avec F={A→B, A→C, A→D}. Démontrer que l'attribut A est une clé candidate ? Est-ce que R est en 3FN ?		Modélisation logique relationnelle	Les attributs, Relations ou tables, Le schéma d'une relation.
	Conception des bases de données	Les dépendances fonctionnelles	Définition d'une DF, Clé d'une relation, Les propriétés des DFs, Une DF élémentaire, Une DF canonique, Une DF directe.
		La normalisation de relations	La première forme normale, La deuxième forme normale, La troisième forme normale.

Tableau 8 : Quelques instances de la classe situation problème.

III.3 L'instanciation de classe obstacle et consigne :

Lors de traitement d'une SP, un apprenant peut s'affronter à des obstacles spécifiques (propres à la discipline) et/ou généraux (liés à plusieurs disciplines). Les obstacles sont considérés comme des difficultés ou encore des manques de la part des apprenants pouvant entraver leur apprentissage. Les consignes sont des guides sous forme d'énoncés qui indiquent le but à atteindre ainsi que la démarche pouvant l'atteindre pour résoudre une SP. Le tableau suivant contient un ensemble d'obstacles disciplinaires relatif à la conception des bases de données (modèle entité-association).

Situation problèmes	Obstacles	Consignes
Un éditeur souhaite installer une BDD pour mémoriser les informations suivantes : Les livres sont identifiés par leur numéro ISBN. Un livre possède un titre et un prix de vente. Il est écrit par un ou plusieurs auteurs, chaque livre est tiré en une ou plusieurs éditions, datées et identifiées par leur ordre.	1. Comment extraire les entités et quelles sont les entités qu'on va prendre en considération. 2. Comment déterminer l'identifiant.	1. Extraire tous les objets concrets et abstraits qui ont une existence propre et présentant un intérêt pour le réel perçu. 2. L'identifiant est une propriété ou groupe de propriétés d'une entité ou d'une association qui permet de distinguer (voir d'une façon

Chaque édition comporte un certain nombre d'exemplaires. Les auteurs sont identifiés par leur nom, prénom et peuvent avoir un pseudonyme. Pour chaque livre, un auteur perçoit des droits d'auteur, calculés comme pourcentage du prix de vente. Les librairies identifiées par leur nom et adresse ; peuvent envoyer des commandes d'un ou plusieurs livres en quantité quelconque. Elaborer le schéma entité-association pour la BDD de la maison d'édition ?	3. Comment savoir quelles sont les associations reliant les entités. 4. Comment définir les cardinalités. 5. Comment s'assurer que le schéma obtenu répond aux besoins des utilisateurs.	unique) ces dernières. 3. Extraire les liens qui attachent entre deux ou plusieurs entités présentant un intérêt pour le réel perçu. 4. La cardinalité est le nombre de fois minimum et le nombre de fois maximum qu'une même occurrence d'une entité peut intervenir dans les occurrences d'une association. 5. Supprimer les transitivités, assurer que le schéma est connexe, assurer qu'il répond aux demandes en le validant avec les utilisateurs.

Tableau 9 : Quelques instances de classe obstacle et consigne.

III.4 Edition et visualisation de l'ontologie de domaine de compétences avec l'outil protégé:

Après avoir terminé la conception de l'ontologie nous allons à présent passer à l'édition de celle-ci tout en utilisant l'éditeur d'ontologie « **Protégé version 3.4.4** », c'est un éditeur open source disponible à l'adresse http://protege.standford.edu, développé au département d'Informatique Médicale de l'Université de Standford. Cela nous permet en effet d'utiliser les technologies du web sémantique.

On a choisi protégé comme outil d'édition de l'ontologie de domaine de compétences car il permet de représenter tous les concepts de l'ontologie sous forme OWL DL, il permet aussi la création des règles d'inférences avec la syntaxe SWRL**.** Il a été utilisé dans l'objectif de générer automatiquement le fichier OWL correspond à l'ontologie et pour également pouvoir le tester au fur et à mesure avec des requêtes **SPARQL** (pour plus de détails consulter la partie réalisation). Il est à noter par ailleurs que <<**Protégé** >> offre bien sûr beaucoup d'autres fonctionnalités, mais qu'on n'a certainement pas toutes utilisé.

III.4.1 Représentation des concepts de l'ontologie de domaine de compétences sous protégé:

Dans le modèle des connaissances de Protégé, les ontologies consistent en une hiérarchie de classes qui ont des attributs, qui peuvent eux-mêmes avoir certaines propriétés. C'est à dire les classes constituent une hiérarchie taxonomique. On peut voir la relation de

sous classes dans un arbre de composites, comme on peut avoir l'héritage multiple. La racine de la hiérarchie est la classe: THING.

Grâce aux trois types d'objets classes, attributs et propriétés et l'interface graphique intermédiaire, on n'a pas besoin d'exprimer ce que l'on a à spécifier dans un langage formel, il suffit juste de remplir les différents formulaires correspondant à ce que l'on veut spécifier.

Les figures ci-dessous présentent les étapes suivies pour la génération de la base de connaissances à partir de l'ontologie de domaine de compétences. On y retrouve des aperçus de la structure taxonomique représentée dans le modèle UML. On y distingue toutes les classes principales et les sous classes (compétences disciplinaires, compétences composées, compétences élémentaires, situation problème …etc.), les attributs, les relations ainsi que les individus identifiées pour notre cas.

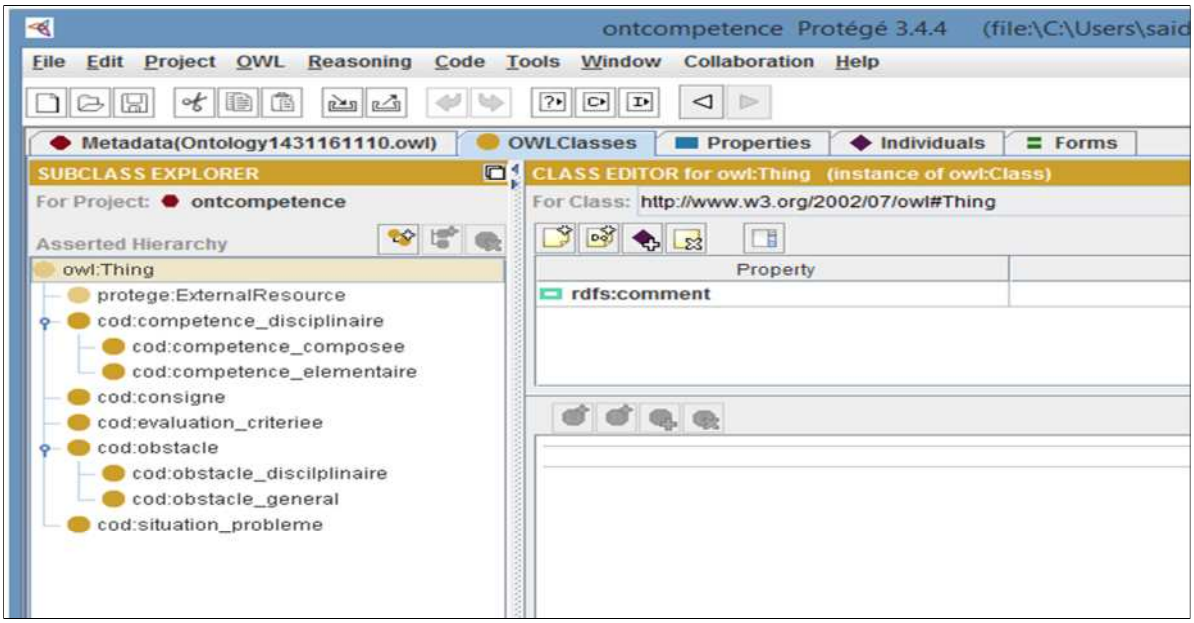

Figure 5 : Représentation des classes et sous classes de l'ontologie de domaine de compétences sous Protégé.

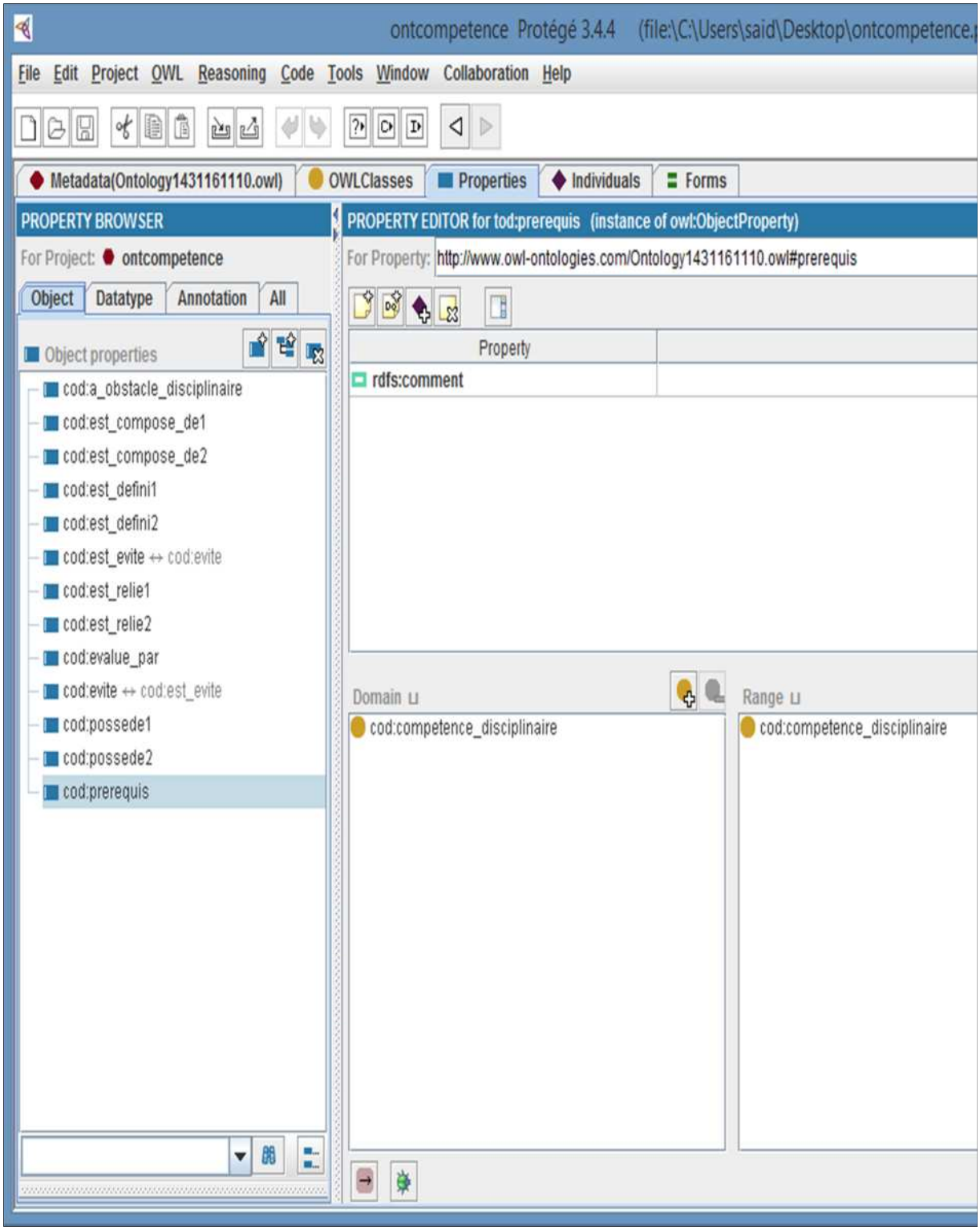

Figure 6 : Représentation des relations de l'ontologie de domaine de compétences sous Protégé.

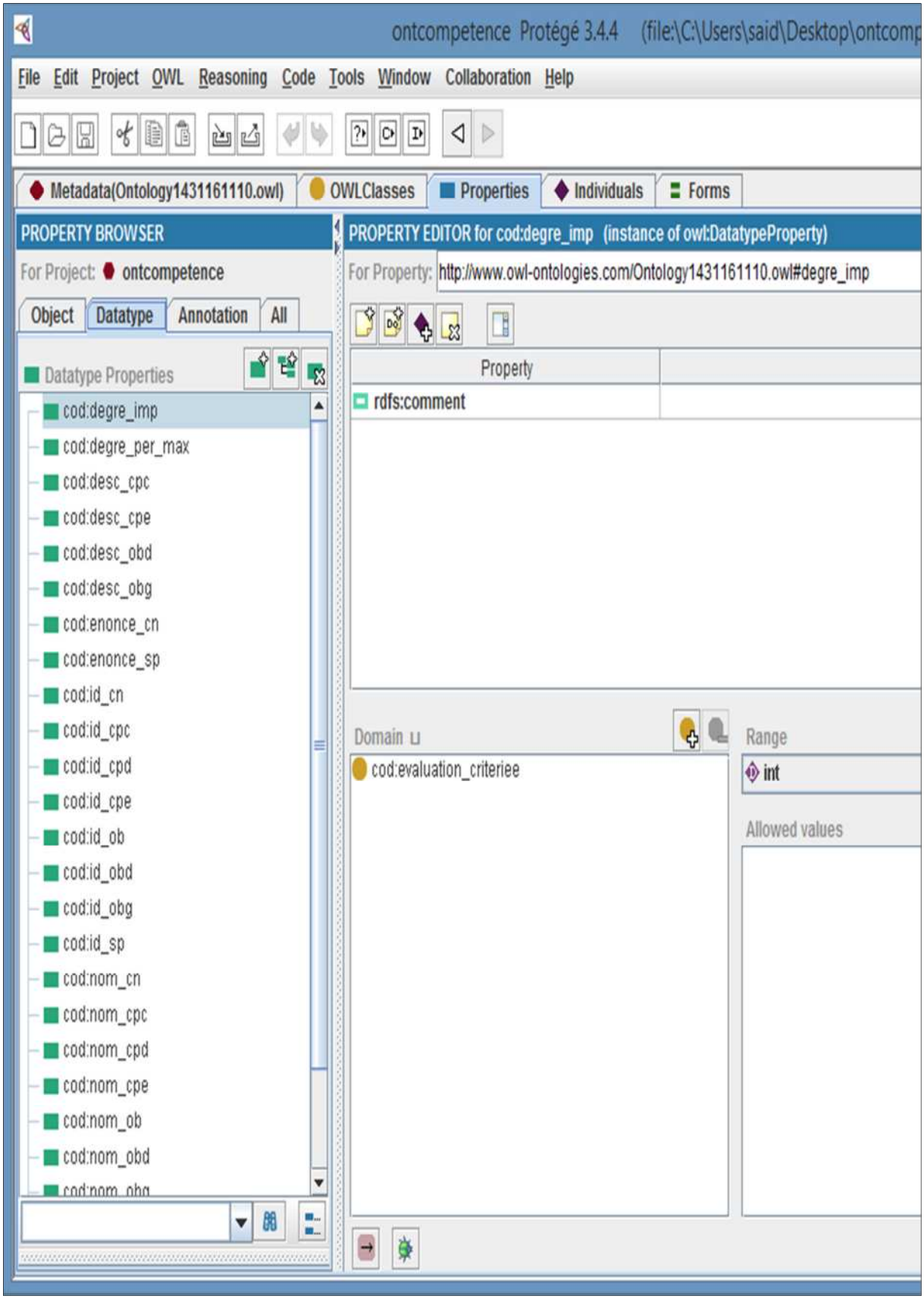

Figure 7 : Représentation des attributs de l'ontologie de domaine de compétences sous Protégé.

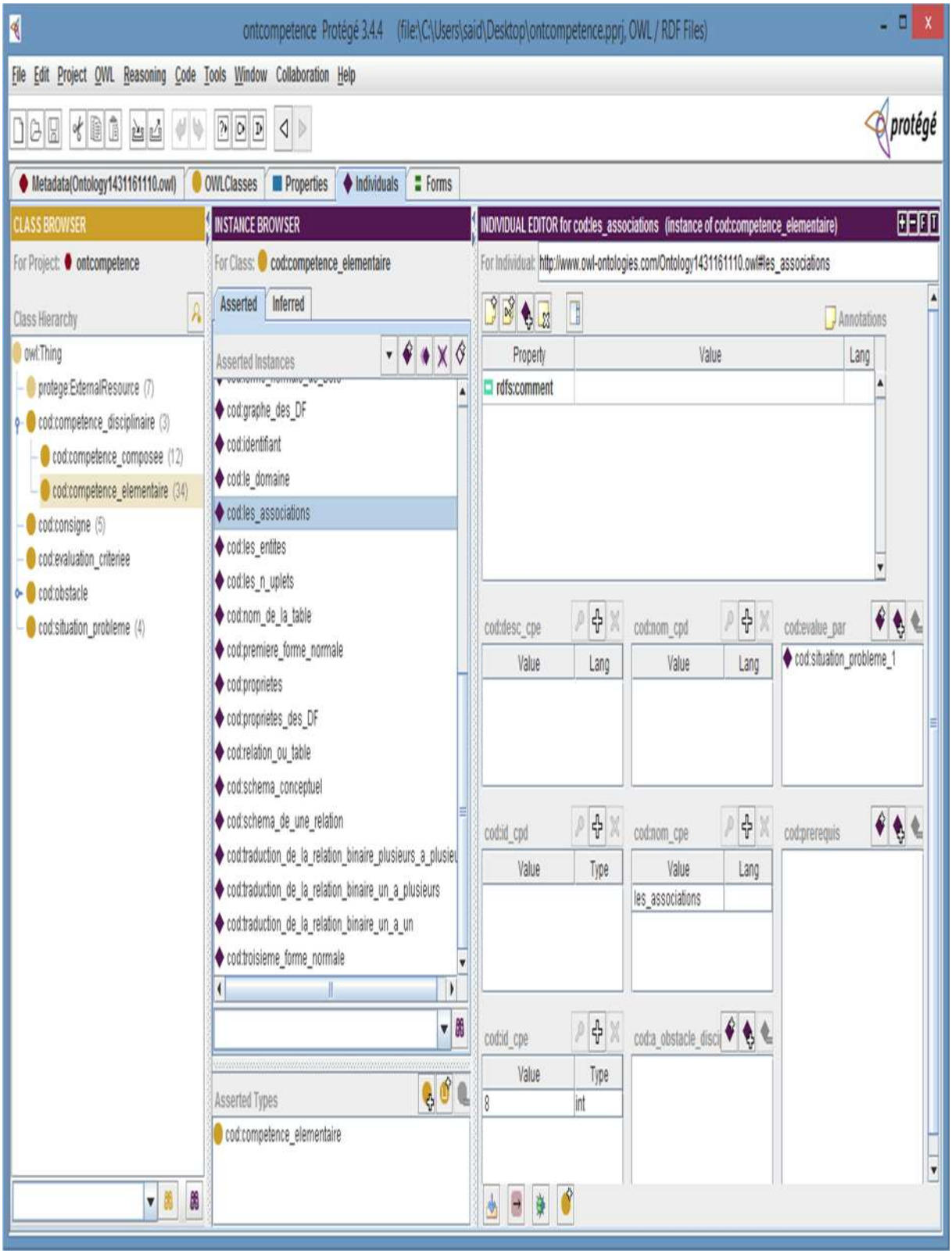

Figure 8 : Représentation des individus de l'ontologie de domaine de compétences sous Protégé.

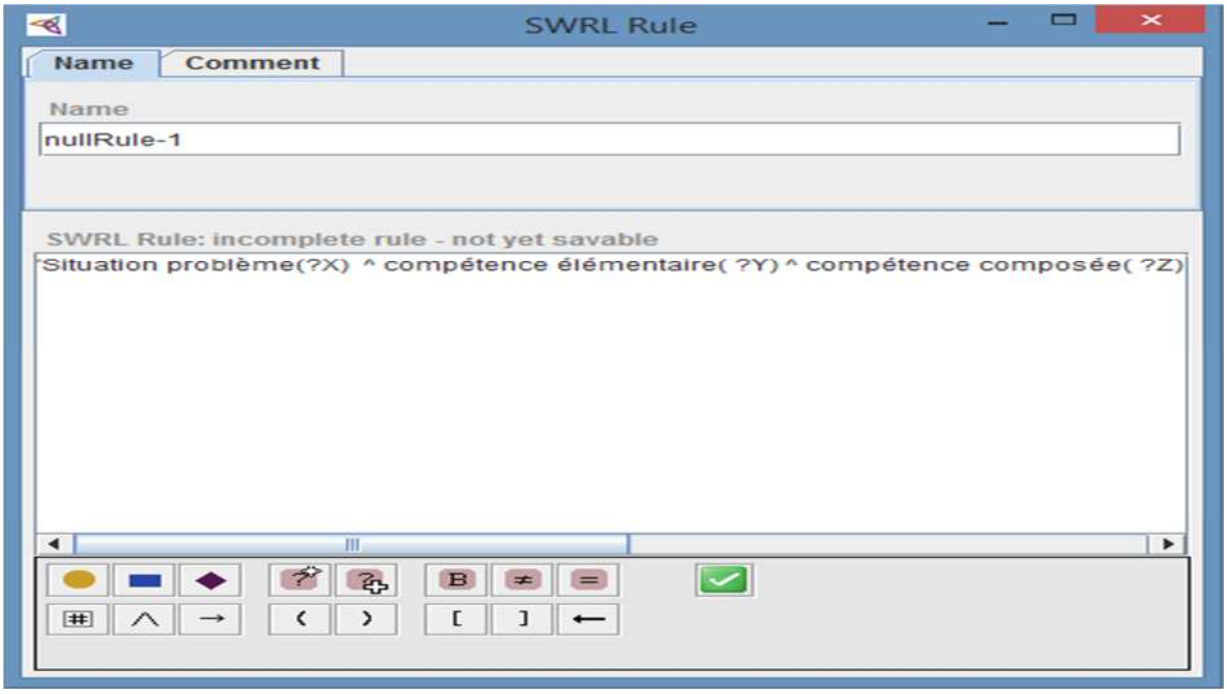

Figure 9 : Représentation d'une règle d'inférence SWRL sous Protégé.

III.4.2 Visualisation graphique de l'ontologie de domaine de compétences sous protégé:

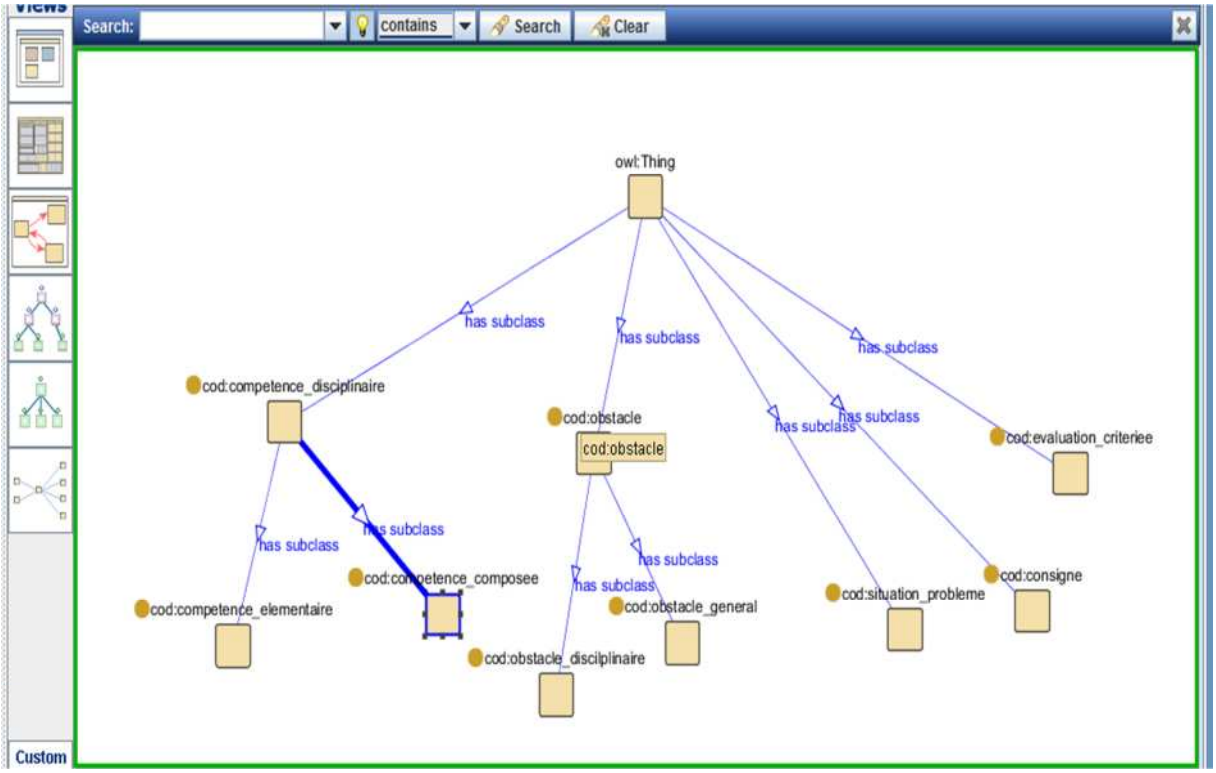

Figure 10 : Représentation des classes et sous classes avec Jambalaya.

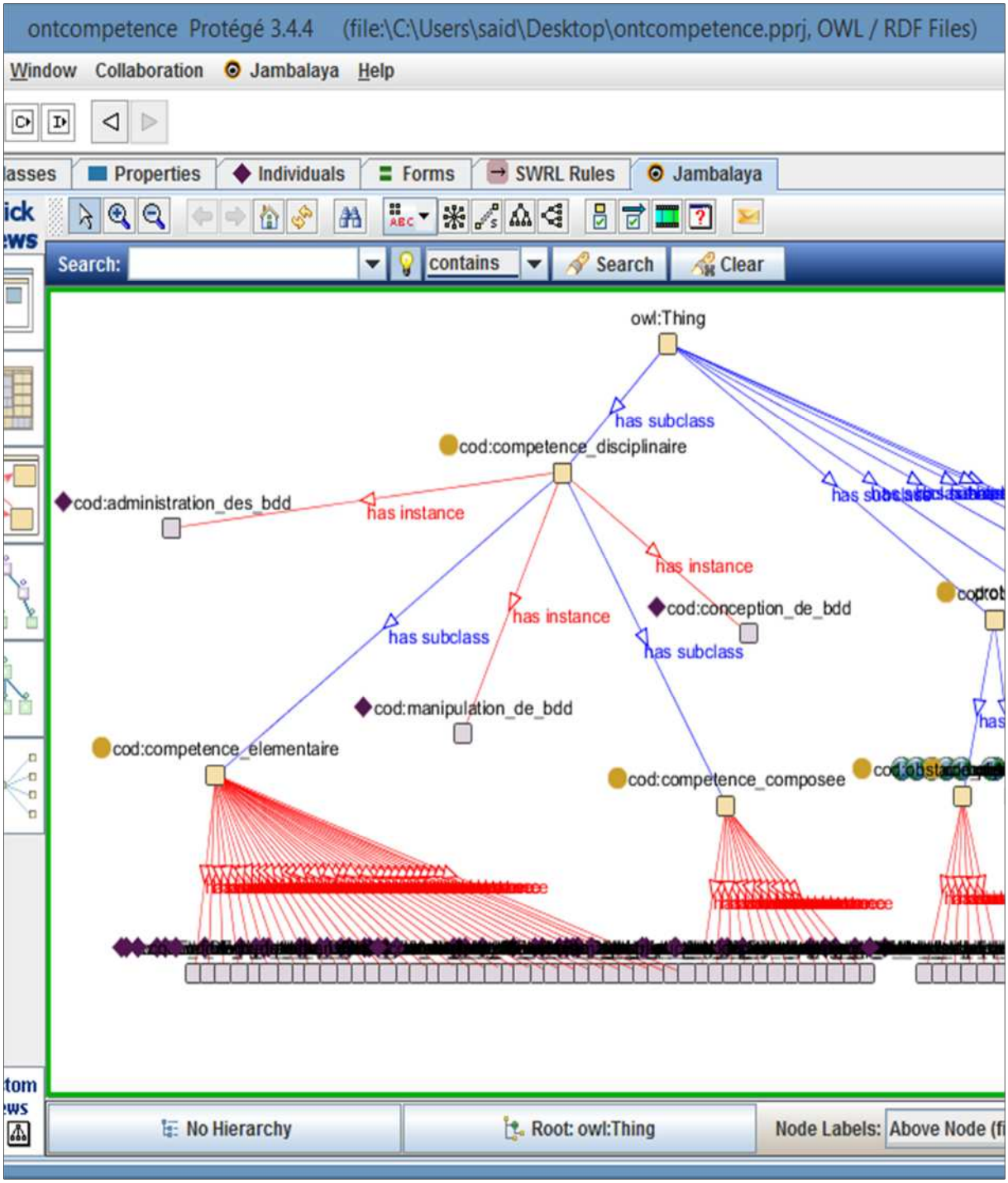

Figure 11 : Représentation des individus sous Jambalaya.

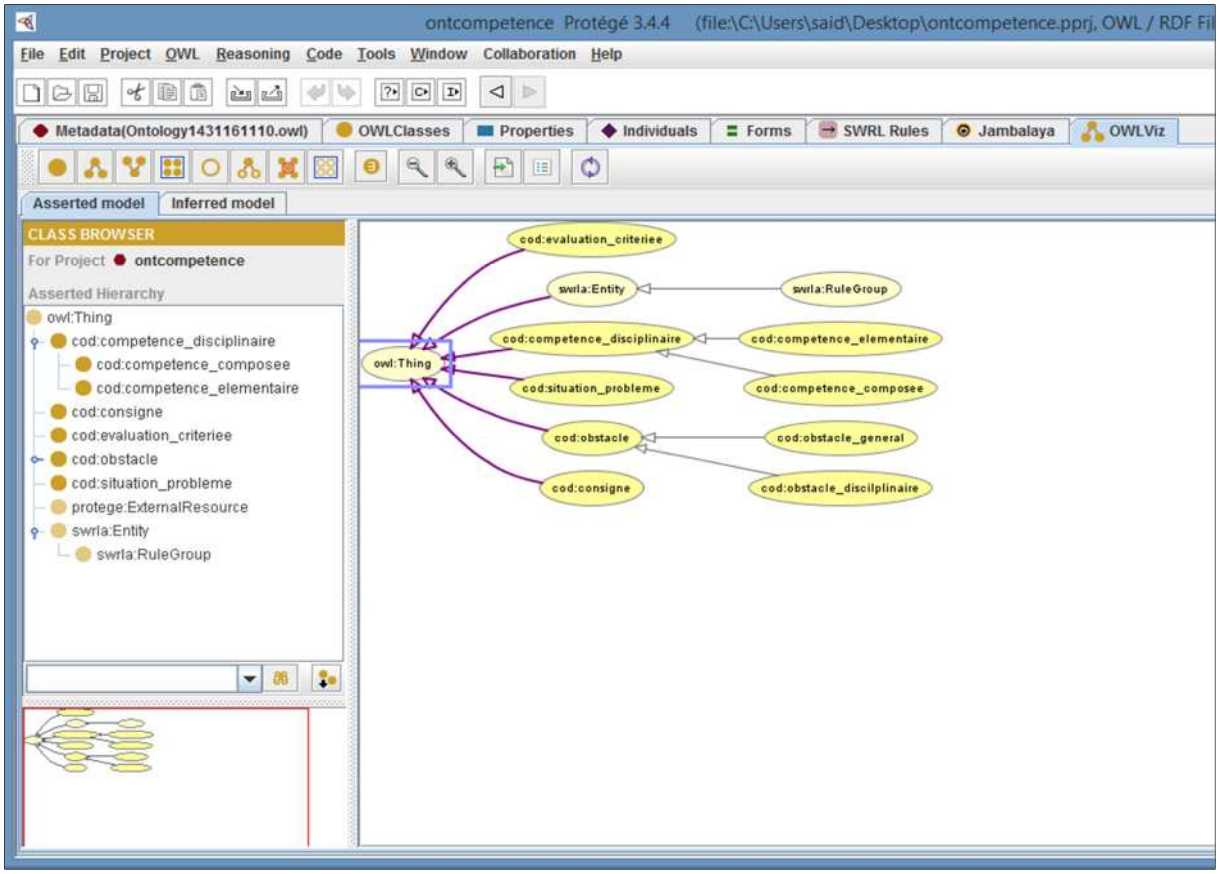

Figure 12 : Représentation de la hiérarchie de l'ontologie de domaine de compétences sous Protégé à l'aide de GraphViz.

III.4.3 Interrogation de l'ontologie de domaine de compétences avec des requêtes SPARQL sous Protégé:

SPARQL (Protocol And RdfQuery Language) est un protocole et un langage de requêtes du W3C (15 janvier 2008) qui permet d'exploiter l'approche sémantique des données RDF. Il est doté : D'un langage de requêtes avec syntaxe basée sur des triplets, d'un protocole d'accès comme un service Web (SOAP), le protocole permet à un client Web de consulter au travers d'une requête SPARQL, un service ou point d'accès SPARQL (end point aussi nommé processor en anglais) et d'un langage de présentation des résultats (XML, HTML, RDF/XML…). Grâce à cette technologie d'interrogation, les utilisateurs peuvent se concentrer sur leur recherche plutôt que sur la technologie de base de données ou le format sur lesquels repose le stockage des données.

SPARQL cible donc l'interrogation de métadonnées RDF, structure de base du Web sémantique. Il fonctionne en parfaite synergie avec les autres technologies Web sémantique du W3C.Avec SPARQL, nous n'avons pas besoin de connaître a priori la structure et le contenu des données pour pouvoir les interroger. En effet, SPARQL permet d'interroger

n'importe quel composant d'un triplet qui a la forme Sujet-Prédicat-Objet**. [24]** La figure suivante montre l'avantage du SPARQL lors de l'interrogation d'un fichier RDF.

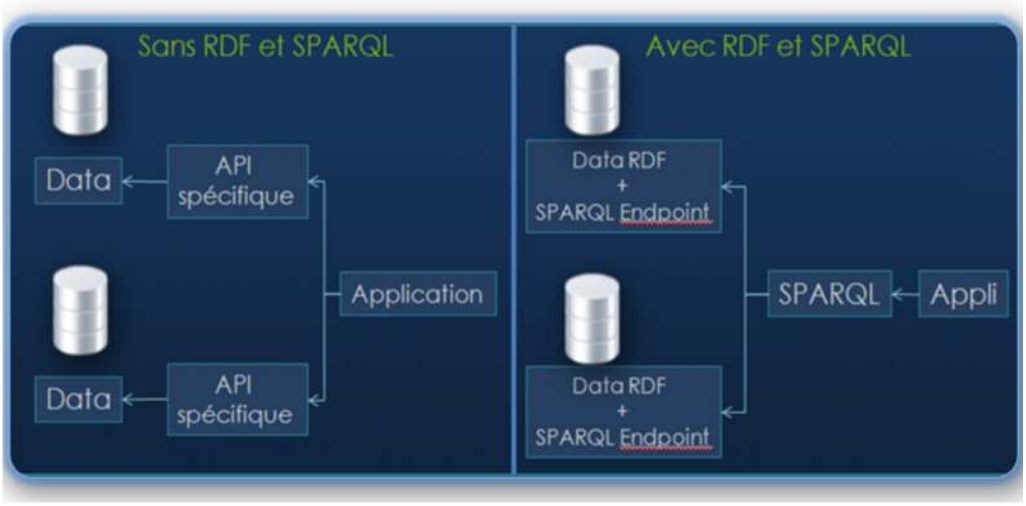

Figure 13 : SPARQL ; une API universelle d'accès aux données.

Une requête SPARQL permet d'interroger l'ontologie afin d'afficher l'ensemble des composants (classes, sous classes, instances, attributs) vérifiant un critère donné. La figure suivante montre un exemple d'une requête sous Protégé avec le résultat de son exécution.

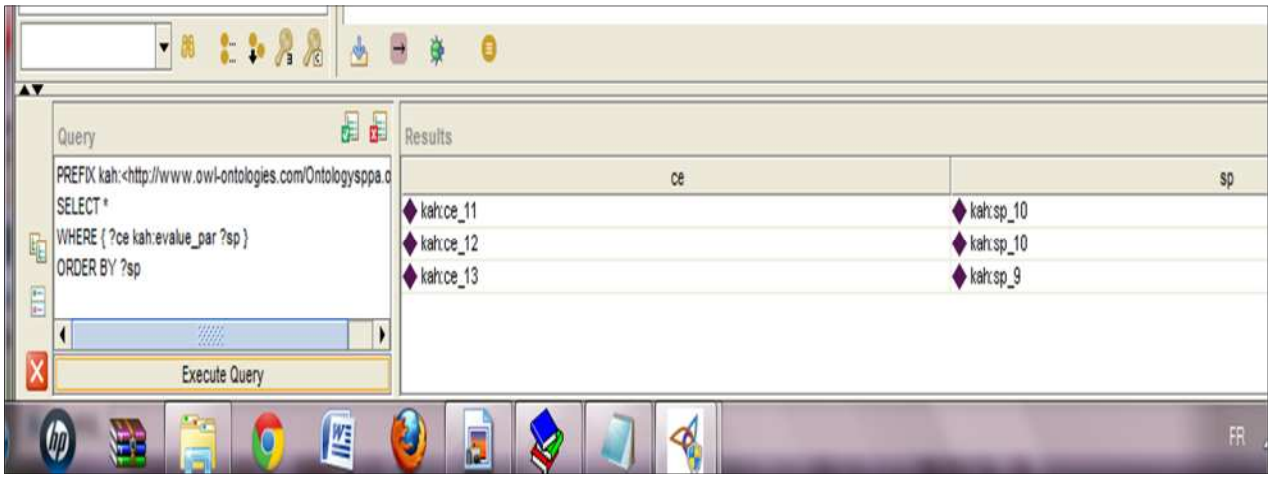

Figure 14 : Exécution d'une requête SPARQL sous Protégé.

III.4.4 Génération du code OWL :

Le code OWL est généré automatiquement par Protégé, pour le visualiser, il suffit de cliquer sur « code », et choisir l'option « show RDF/XML source code » comme illustré dans la figure suivante :

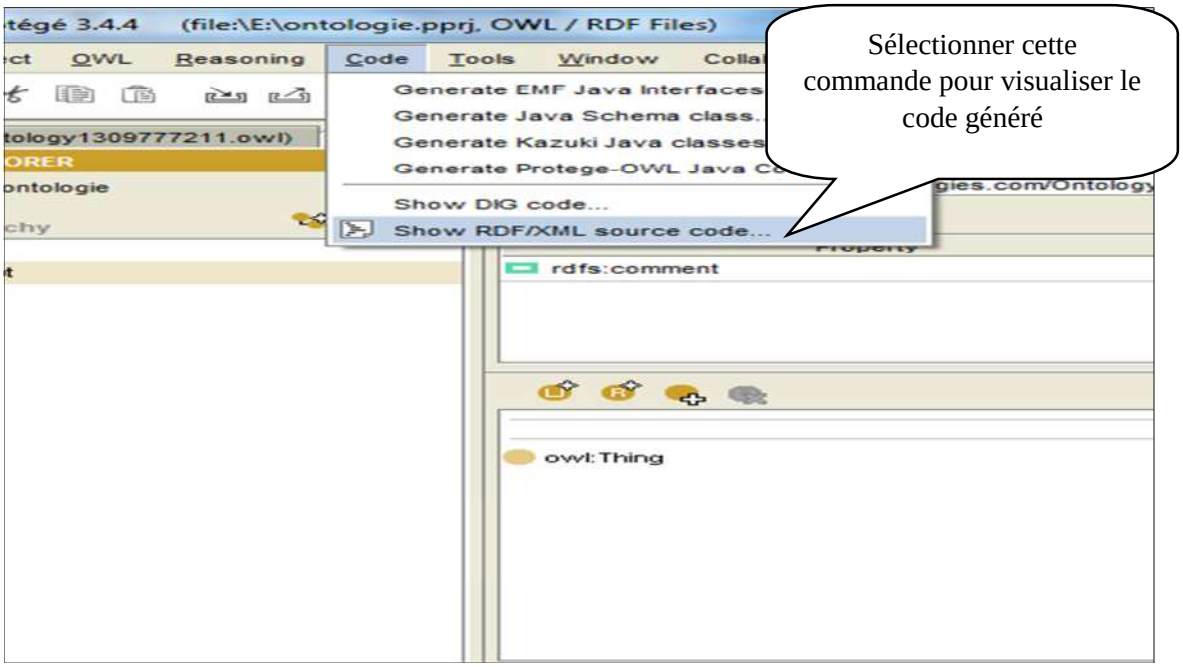

Figure 15 : Visualisation du code OWL sous Protégé.

Le code OWL correspondant à l'ontologie de domaine de compétences étant relativement long, voici un extrait : (Le code complet est présenté en Annexe).

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:protege="http://protege.stanford.edu/plugins/
   xmlns:tod="http://www.owl-ontologies.com/Ontolo
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schem
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#
 xml:base="http://www.owl-ontologies.com/Ontology14
<owl:Ontologyrdf:about=""/>
<owl:ObjectPropertyrdf:ID="a_obstacle_disciplinaire">
<rdfs:domainrdf:resource="#competence_disciplinaire"/>
<rdfs:rangerdf:resource="#obstacle_disciplinaire"/>
</owl:ObjectProperty>
<owl:Classrdf:ID="consigne"/>
<owl:DatatypePropertyrdf:ID="enonce_cn">
<rdfs:domainrdf:resource="#consigne"/>
<rdfs:rangerdf:resource="http://www.w3.org/2001/XMLS
</owl:DatatypeProperty>
<owl:DatatypePropertyrdf:ID="enonce_sp">
<rdfs:domainrdf:resource="#situation_probleme"/>
<rdfs:rangerdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

La relation a_obstacle_disciplinaire entre les deux classes CpDG et ObD

L'attribut énoncé de la classe SP

```
<owl:ObjectPropertyrdf:ID="est_compose_de1">
<rdfs:domainrdf:resource="#competence_disciplinaire"/>
<rdfs:rangerdf:resource="#competence_disciplinaire"/>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="est_compose_de2">
<rdfs:domainrdf:resource="#competence_disciplinaire"/>
<rdfs:rangerdf:resource="#competence_elementaire"/>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="est_defini2">
<rdfs:domainrdf:resource="#consigne"/>
<rdfs:rangerdf:resource="#situation_probleme"/>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="est_defini1">
<rdfs:domainrdf:resource="#situation_probleme"/>
<rdfs:range>
<owl:Class>
<owl:unionOfrdf:parseType="Collection">
<owl:Classrdf:about="#competence_elementaire"/>
<owl:Classrdf:about="#evaluation_criterriee"/>
</owl:unionOf>
</owl:Class>
</rdfs:range>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="est_evite">
<rdfs:domainrdf:resource="#ostacle"/>
<owl:inverseOfrdf:resource="#evite"/>
<rdfs:rangerdf:resource="#consigne"/>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="est_relie1">
<rdfs:domainrdf:resource="#ostacle"/>
<rdfs:rangerdf:resource="#situation_probleme"/>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="est_relie2">
<rdfs:domainrdf:resource="#obstacle_disciplinaire"/>
<rdfs:rangerdf:resource="#competence_disciplinaire"/>
</owl:ObjectProperty>
<owl:Classrdf:ID="evaluation_criterriee"/>
<owl:ObjectPropertyrdf:ID="evalue_par">
<rdfs:domainrdf:resource="#competence_elementaire"/>
<rdfs:rangerdf:resource="#situation_probleme"/>
</owl:ObjectProper
<competence_disciplinaire rdf:ID="administration_des_bdd">
 <id_cpdrdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</id_cpd>
<nom_cpd rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
<administration_des_bdd</nom_cpd>
<prerequis rdf:resource="#conception_de_bdd"/>
<prerequis rdf:resource="#manipulation_de_bdd"/>
 </competence_disciplinaire>
```

Instance de la classe CpDG avec ses prérequis

Figure 16: Représentation d'un extrait de code OWL généré automatiquement par protégé.

III.5 Structure du code OWL :

La conception d'OWL a pris en compte la nature distribuée du web sémantique et, de ce fait, a intégré la possibilité d'étendre des ontologies existantes, ou d'employer diverses ontologies existantes pour compléter la définition d'une nouvelle ontologie.

Une ontologie formalisée en OWL comprend une URI, un espace de nom et un espace de nom préfixe.

III.5.1 Espace de nommage :

Afin de pouvoir employer des termes dans une ontologie, il est nécessaire d'indiquer avec précision de quels vocabulaires ces termes proviennent. C'est la raison pour laquelle, comme tout autre document XML, une ontologie commence par une déclaration d'espace de nom (parfois appelée « de nommage ») contenue dans une balise rdf:RDF.

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:tod="http://www.owl-ontologies.com/Ontology1431161110.owl#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
 xml:base="http://www.owl-ontologies.com/Ontology1431161110.owl">
  <owl:Ontology rdf:about=""/>
```

La déclaration « xmlns:tod=http://www.owl-ontologies.com/Ontology1431161110.owl# » indique à quelle ontologie se rapporter en cas d'utilisation de noms avec un préfixe dans la suite de l'ontologie. La déclaration « xml:base="http://www.owl-ontologies.com/Ontology1431161110.owl » identifie l'URI de base de l'ontologie courante.
Les autres déclarations introduisent le vocabulaire d'OWL et les objets définis dans l'espace de nommage de RDF, du schéma RDF et des types de données du Schéma XML.

III.5.2 L'en-tête d'une ontologie :

Tout comme il existe une section d'en-tête <head>…</head> en haut de tout document XHTML bien formé, on peut écrire, à la suite de la déclaration d'espaces de nom, un en-tête décrivant le contenu de l'ontologie courante. C'est la balise owl:Ontology qui permet d'indiquer ces informations :

```
<owl:Ontologyrdf:about="">
<owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
<owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"/>
</owl:Ontology>
```

IV. Description de l'application :

Notre application peut être vue comme un portail d'acquisition de connaissances, utilisée par deux types d'intervenants : les apprenants et les experts de domaine. C'est une application qui vise à exploiter une ontologie de domaine de compétences à l'aide des technologies de web sémantique.

Ainsi, nos objectifs sont :

- Offrir la possibilité de mettre à jour les éléments de l'ontologie générique relative au domaine de compétences.
- Fournir la possibilité de créer des éléments pour une nouvelle ontologie présentant un domaine de compétences quelconque.
- Donner la possibilité de choisir une compétence disciplinaire et d'afficher les compétences intermédiaires qui la composent.
- Donner la possibilité de choisir une situation problème et d'afficher son énoncé ainsi que la consigne qui lui correspond.
- Détecter tous les obstacles disciplinaires qui peuvent être liés à une compétence disciplinaire donnée.
- Afficher les compétences prérequis pour une compétence donnée.
- Afficher les compétences acquises par un apprenant donné.

Pour réaliser nos objectifs, nous avons développé une application web permettant aux apprenants de s'inscrire et de bénéficier des connaissances disponibles.

L'application permet aussi aux experts de domaine de produire des compétences disciplinaires, composées et élémentaires, des situations problème et des consignes, ainsi la possibilité d'héberger ces ressources pédagogiques dans le serveur.

Nous avons intéressé au module de bases de données, où nous avons proposé trois compétences disciplinaires à acquérir à savoir ; conception de BDD, manipulation de BDD et administration des BDD.

IV.1 Le diagramme de contexte :

Notre application est un système multi- utilisateurs : à tout instant, on peut avoir plusieurs instances de l'apprenant, de l'expert et d'autres utilisateurs connectés au système.

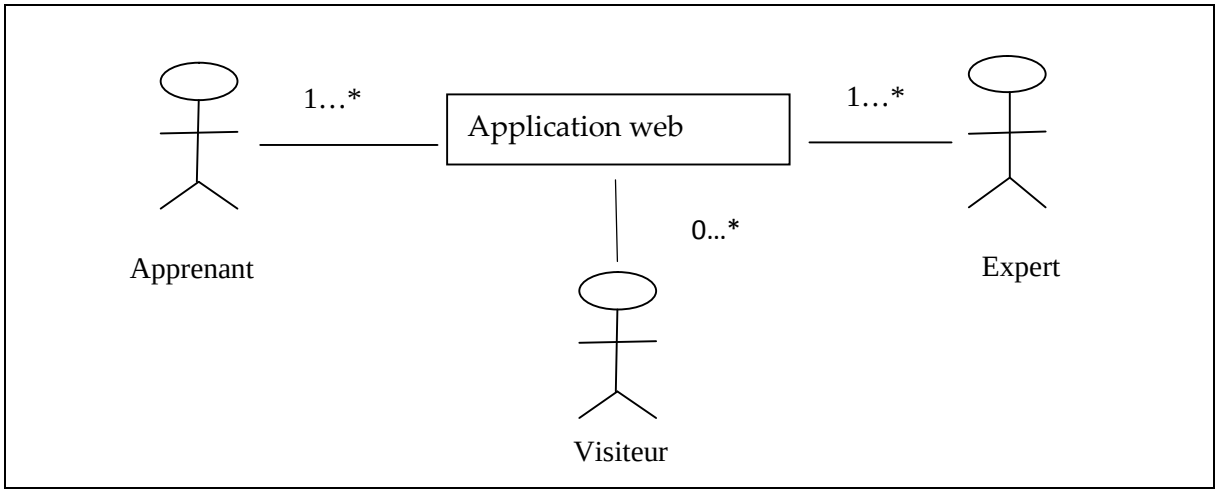

Figure 17 : Diagramme de contexte.

IV.2 Détermination des cas d'utilisation de l'application :

IV.2.1 Diagramme de cas d'utilisation de l'expert :

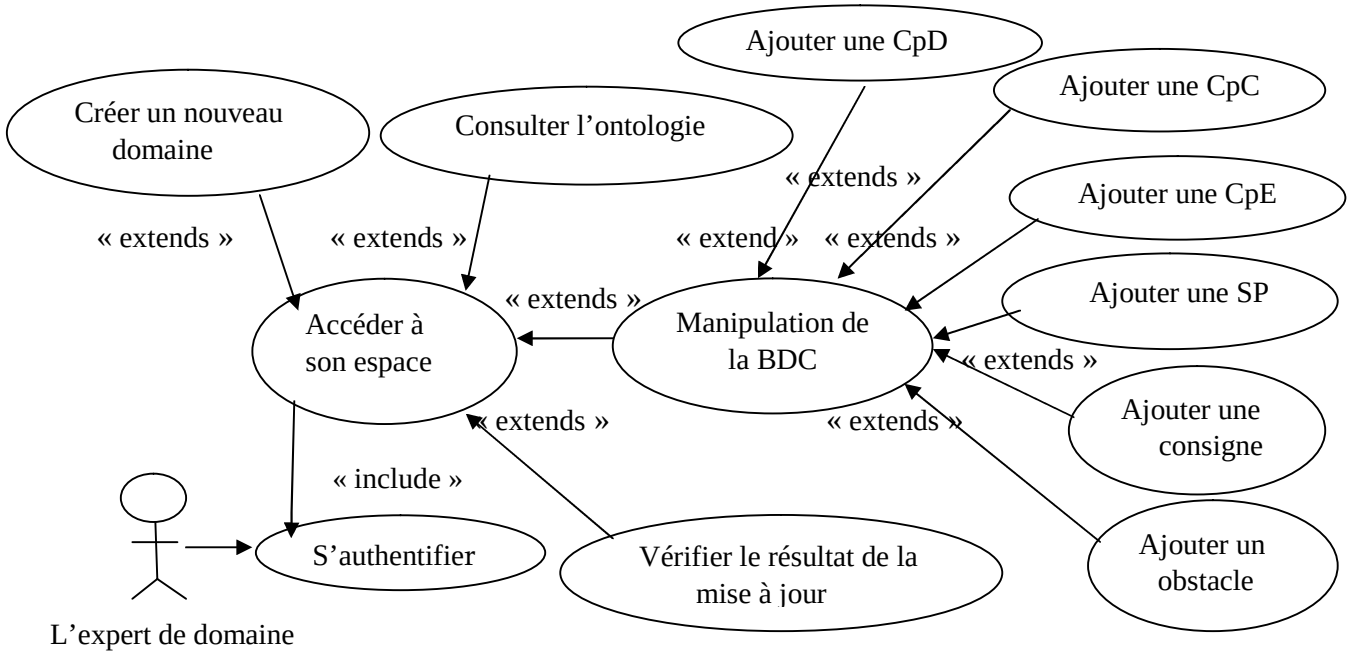

Figure 18 : Cas d'utilisation pour l'expert de domaine.

IV.2.2 Diagramme de cas d'utilisation de l'apprenant :

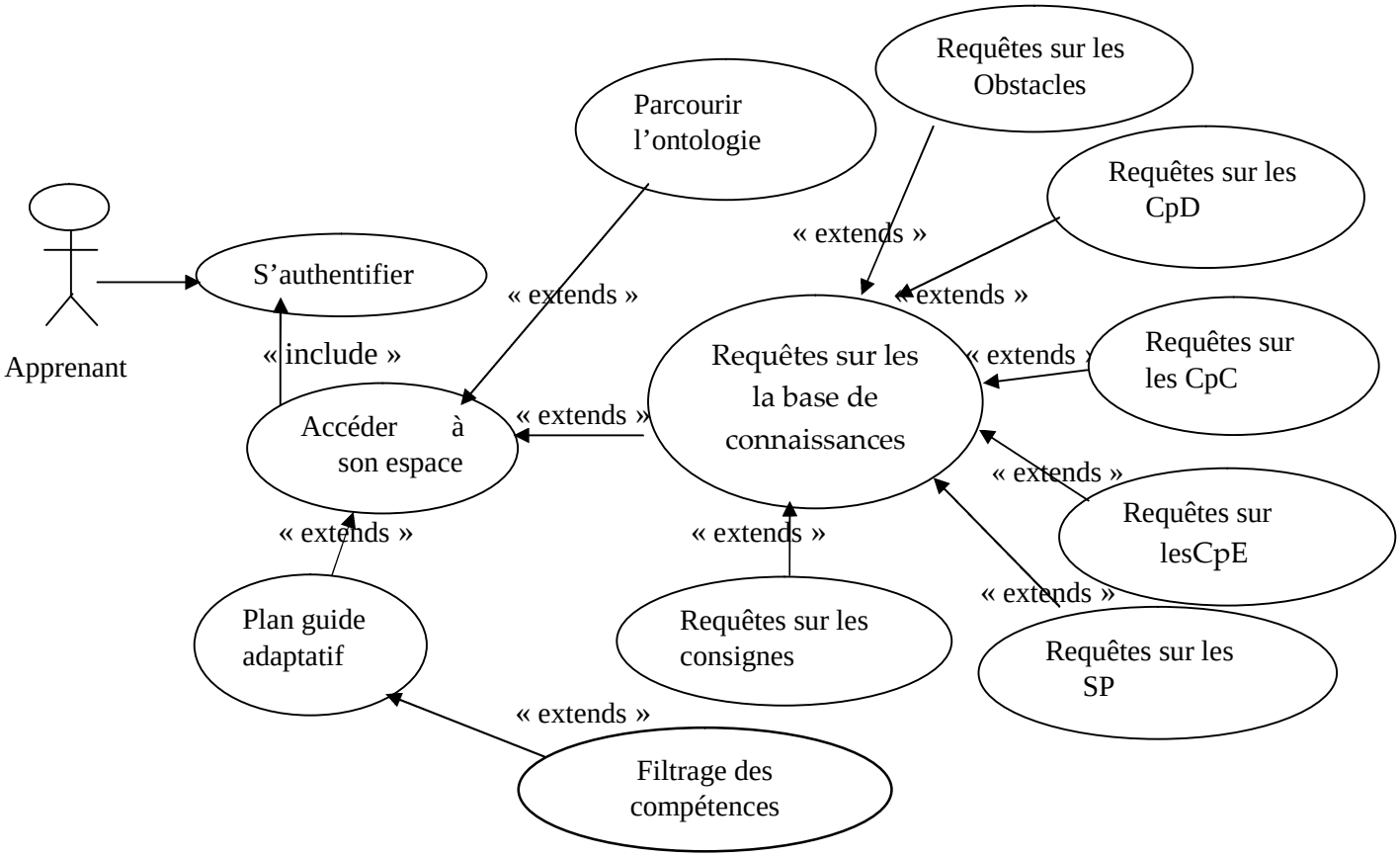

Figure 19 : Cas d'utilisation pour l'apprenant.

IV.3 Exemple de description textuelle de quelques tâches communes des cas d'utilisation :

Pour détailler le déroulement d'un cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre les acteurs et le système. Dans ce qui suit nous décrivons donc quelques cas d'utilisation de notre système.

- **Cas d'utilisation : Inscription**

Titre : Inscription. **Résumé** : Un visiteur accède à l'application et s'inscrit en tant qu'apprenant ou expert de domaine **Acteurs** : visiteur

Enchainement :

Ce cas d'utilisation commence lorsque l'utilisateur clique sur le lien « S'inscrire »

Scénario nominal :

1. Le système affiche le formulaire d'inscription ;

2. L'utilisateur remplit le formulaire et clique sur le bouton « envoyer » pour valider l'inscription ;

3. Le système vérifie la cohérence des informations saisies ;

4. Le système retourne un message de confirmation indiquant à l'utilisateur que son compte est validé.

Scénario alternatif : Les données saisies sont erronées.

Le scénario nominal démarre au point 3.

5. Le système signale l'erreur et demande à l'utilisateur de corriger.

Le scénario nominal démarre au point 2.

Scénario d'exception : L'utilisateur clique sur le bouton « Annuler ».

L'enchainement commence au point 1 du scénario nominal.

6. L'utilisateur clique sur le bouton « annuler ».

7. Le système affiche la page d'accueil.

- **Cas d'utilisation : Authentification**

Titre : Authentification. **Résumé** : Un utilisateur inscrit s'identifie par son login et mot de passe pour accéder à son espace dans l'application. **Acteurs** : Expert de domaine et apprenant.

Enchainement :

Ce cas d'utilisation commence lorsque l'utilisateur clique sur le lien de son espace approprié.

Scénario nominal :

1. Le système affiche la page d'accueil avec le formulaire d'identification ;

2. L'utilisateur remplit les champs en tapant son login et son mot de passe et clique sur « valider » pour envoyer ;

3. Le système vérifie la cohérence des informations saisies (l'existence du compte) ;

4. Le système affiche l'espace approprié à l'utilisateur.

Scénario alternatif : Login ou mot de passe incorrect.

Le scénario nominal démarre au point 3 et enchaine :

5. Le système affiche un message d'erreur et demande à l'utilisateur de retaper le login et le mot de passe.

- **Cas d'utilisation : Voir les situations problème liées à une compétence choisie**

Titre : Voir une situation problème liées à une compétence choisie **Résumé** : Un apprenant accède à son espace et choisi une compétence pour l'évaluer, les

situations problème liées à cette dernière seront affichées, une fois une SP choisie, il peut se servir de consignes.
Acteurs : Apprenant.

Enchainement :

Ce cas d'utilisation commence une fois l'apprenant est dans son espace approprié.

Scénario nominal :

1. Le système accède à l'espace apprenant et un bouton d'interrogation de l'ontologie sera affiché ;
2. L'apprenant clique sur ce bouton une autre page s'affiche.
3. l'apprenant choisi une compétence à évaluer, les situations problème liées à cette dernière seront présentées sur l'écran ;
4. Le système affiche l'espace approprié avec la SP choisie une fois cliquer sur cette dernière.
5. L'apprenant peut servir d'une consigne qui l'aide à résoudre la SP.

V. L'architecture de l'application dans un environnement java :

V.1 Description du système :

Le système que nous allons construire se compose essentiellement de trois modules ; recherche, mise à jour de la base de connaissances et génération du plan guide pour le filtrage des compétences. Le module recherche traite les requêtes des utilisateurs et affiche les résultats de la recherche. Cette dernière se fait sur deux étapes : d'abord le module de recherche interroge l'ontologie (fichier OWL) pour récupérer les termes demandés par l'utilisateur et les lui afficher. L'expert du domaine fait l'enrichissement de l'ontologie en s'aidant des nouveaux éléments. Dans ce qui suit, nous allons décrire les différents modules du système.

V.1.1 Module de recherche :

Le processus de recherche représente l'interface du système avec l'utilisateur. En effet, c'est à travers ce processus que l'utilisateur exprime son besoin à l'aide d'une requête, cette dernière suit une syntaxe précise définie par le processus de recherche afin qu'il puisse l'exploiter pour fournir les concepts correspondants aux besoins de l'utilisateur. Le processus de recherche se complète par un processus d'affichage de résultats. Le processus de recherche est schématisé dans la figure suivante :

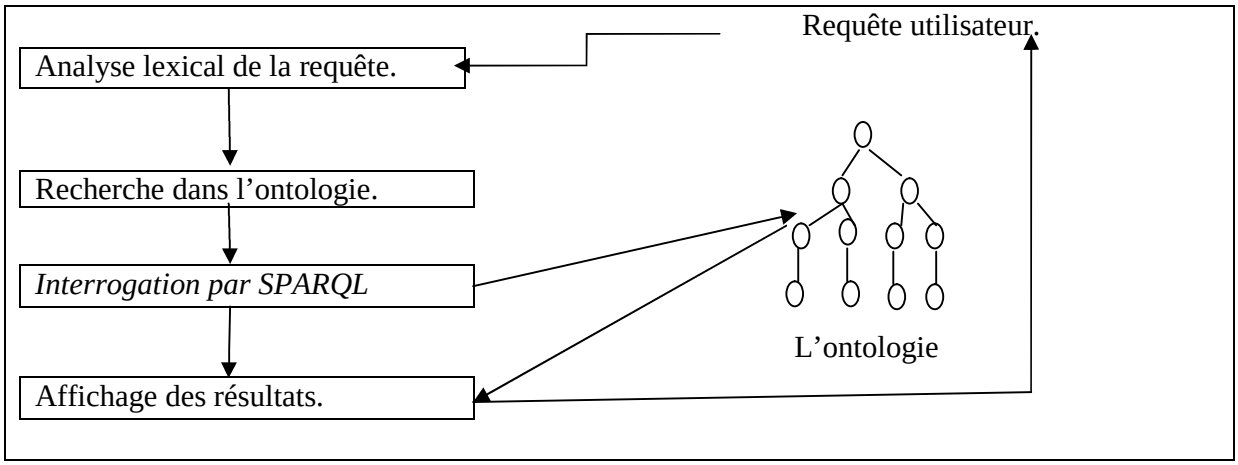

Figure 20 : Le processus de recherche.

Le processus de recherche est composé des étapes suivantes :

- Analyse lexical de la requête *:* Cette étape permet de découper la requête en un ensemble de tokens (reconnaissance des séparateurs entre mots).

- Recherche dans l'ontologie : Dans cette étape, on cherche dans l'ontologie les concepts de la requête.

- Interrogation par SPARQL : revient à faire une recherche sémantique à l'aide d'une requête SPARQL générée.

- Affichage des résultats : C'est la dernière étape du processus de recherche. Elle permet d'afficher le résultat de la recherche.

V.1.2 Module de la mise à jour de la base de connaissances :

Comme nous l'avons déjà vue, le cycle de vie d'une ontologie est un processus itératif et elle est en constante évolution. Notre système a besoin d'une ontologie plus riche possible pour avoir une meilleure recherche. Nous proposons alors un moyen simple pour aider l'expert du domaine à enrichir l'ontologie en plus de sa propre recherche dans le domaine.

V.1.3 génération du plan guide :

Ce module assure le filtrage des différentes compétences (disciplinaires, composées et élémentaires) pour chaque apprenant en tenant compte de son profile (état d'avancement) et la relation prérequis entre les compétences. Le résultat de cette tâche est la récupération de Lcd (liste des compétences disciplinaires non acquises), Lci (liste des compétences intermédiaires ou composées non acquises) et Lce (liste des compétences élémentaires non acquises) pour chaque apprenant à fin de suivre son avancement pendant son apprentissage.

Les différentes fonctionnalités du système à développer sont résumées dans la figure ci-dessous :

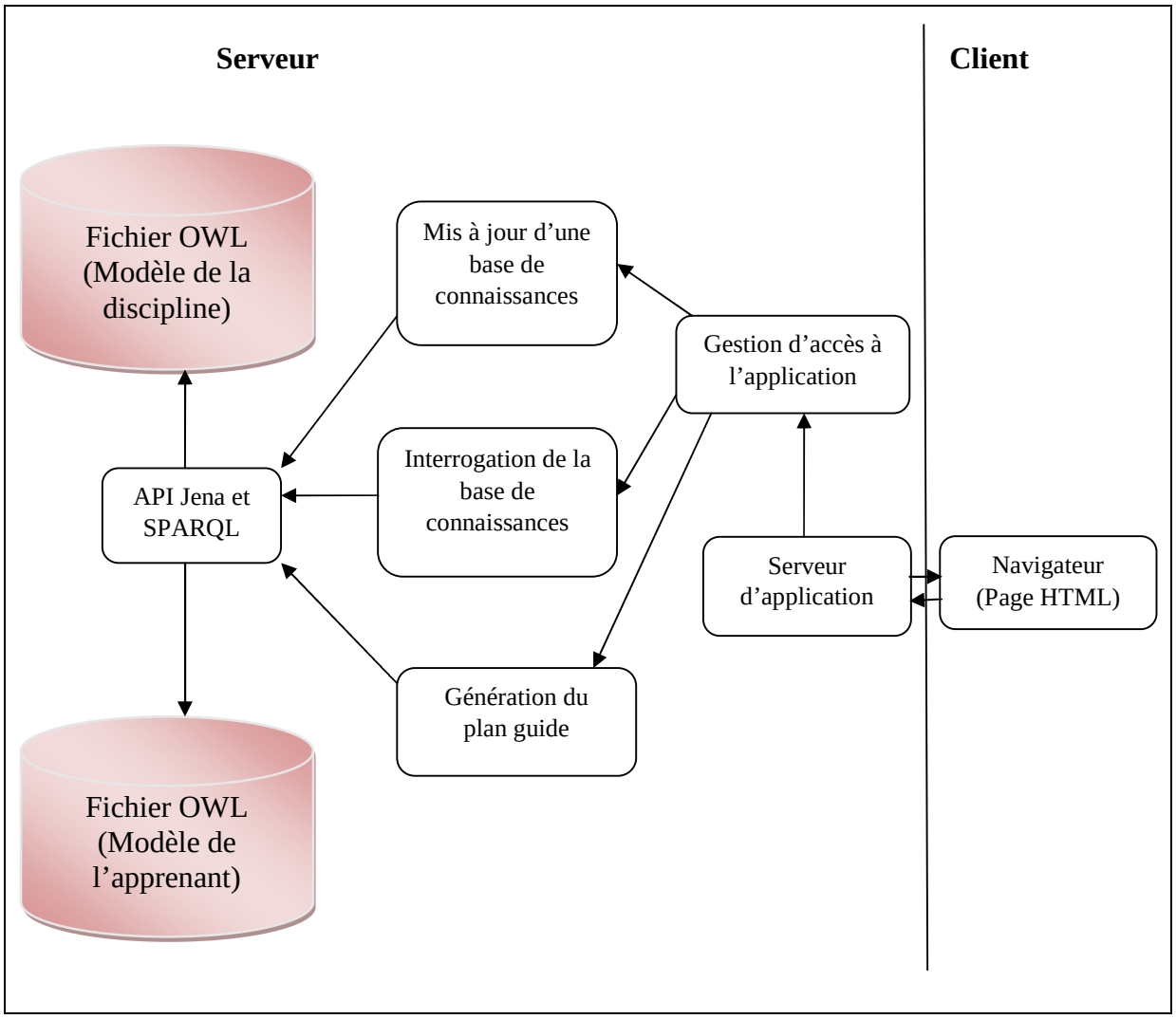

Figure 21 : L'architecture de l'application de domaine de compétences.

VI. Description de la génération du plan guide :

VI.1 L'organigramme pour le filtrage des compétences :

Nous considérons ici l'hypothèse suivante : Une compétence est acquise si le degré d'acquisition (DgA) est supérieur ou égale à 60%.

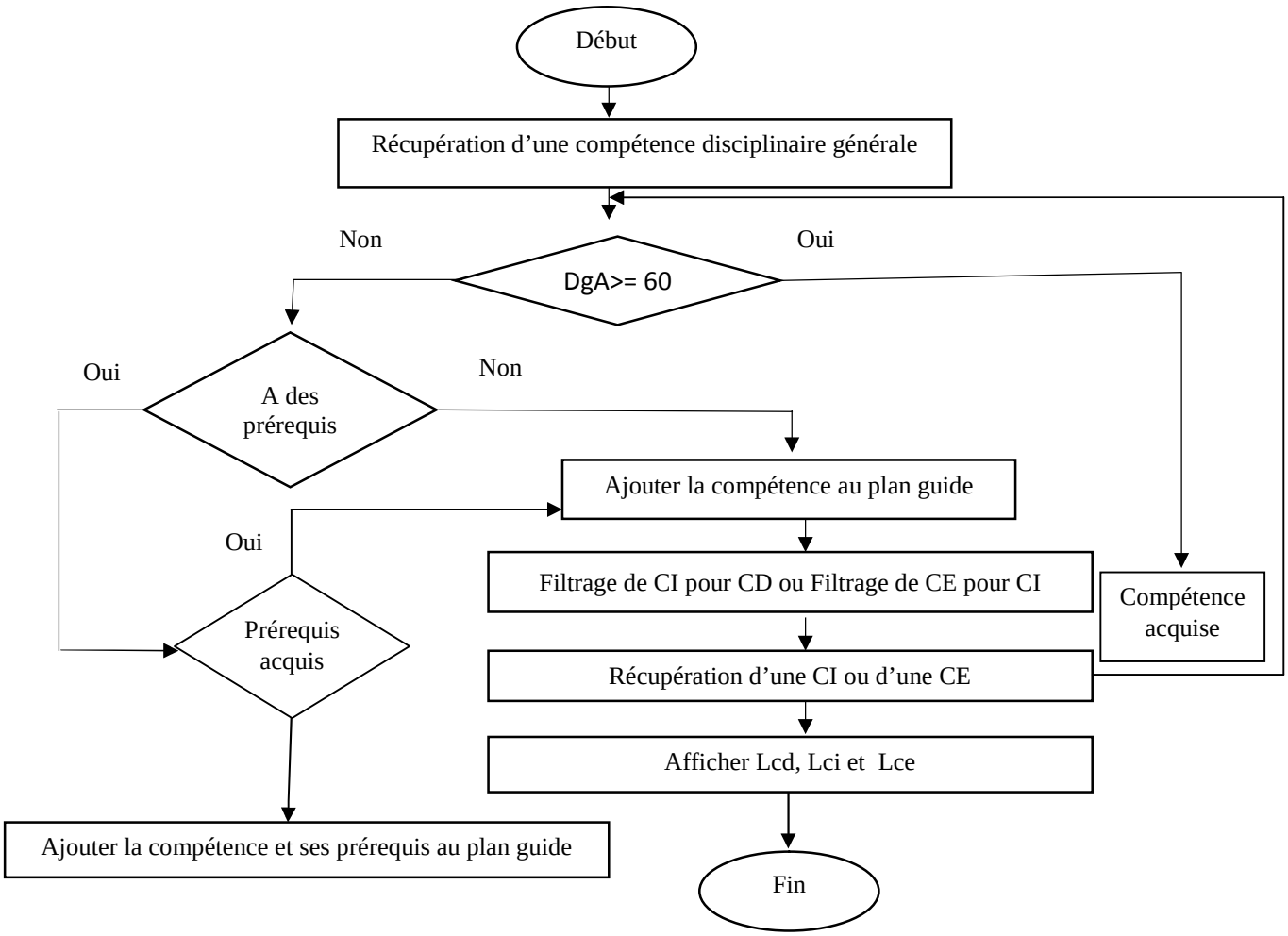

Figure 22 : L'organigramme de la génération du plan guide pour le filtrage des compétences.

VI.2 L'algorithme global pour la génération du plan guide :

Algorithme filtrage des compétences ;

Fonction filtrage-CDG (Lcd : liste-compétences-disciplinaires générale) : liste

Début

Charger toutes les compétences disciplinaires CDG à partir de la BDC ;
Choisir une compétence disciplinaire CDGi ;
Pour (chaque CDGi) faire
Si CDGi non acquise alors
Chercher les prérequis de CDGi ;
Si pas de prérequis alors
Ajouter CDGi à Lcd ;
Filtrage CI ;
Sinon
Début

Afficher tous ses prérequis à partir de la BDC ;

Si prérequis acquis alors

Filtrage CI ;

Sinon

Ajouter CDGi et ses prérequis à Lcd ;

Finsi

Fin

Finsi

Sinon

Les CIj ∈ Cdi aquises ;

Les CEk ∈ Cdi aquises ;

Les prérequis de CDGi sont acquis ;

Finsi

Fait

Retourner Lcd ; **renvoyé le résultat.

Fin

Fonction filtrage-CI (Lci : liste-compétences-intermédiaires) : liste

Début

Charger toutes les compétences intermédiaires CI pour chaque CDGi à partir de la BDC;

Choisir une compétence intermédiaires CIj;

Pour (chaque CIj ∈ CDGi) faire

Si CIj non acquise alors

Chercher les prérequis de CIj ;

Si pas de prérequis alors

Ajouter CIj à Lci ;

Filtrage CE ;

Sinon

Afficher tous ses prérequis à partir de la BDC ;

Si prérequis acquis alors

Filtrage CE ;

Sinon

```
                                    Début
                                    |  Ajouter CIjet ses prérequis à Lci ;
                                    |  Ajouter CDGi  à Lcd ;
                                    Fin
                                Finsi
                            Finsi
                        Sinon
                        |  CE ∈ CIj acquises ;
                        Finsi
                Fait
            Retourner Lci ;  ** renvoyé le résultat.
        Fin
```

Fonction filtrage-CE (Lce : liste-compétences-élémentaires) : liste

```
Début
|   Charger toutes les compétences élémentaires CE à partir de la BDC ;
|   Choisir une compétence élémentaires CEk;
|
|   Pour (chaque CEk ∈ CIj) faire
|   |       Si CEk non acquise alors
|   |       |   Chercher les prérequis de CEk ;
|   |       |   Si pas de prérequis alors
|   |       |   |       Ajouter CEk à Lce ;
|   |       |   Sinon
|   |       |   |   Début
|   |       |   |   |   Afficher tous ses prérequis à partir de la BDC ;
|   |       |   |   |   Si prérequis non acquis alors
|   |       |   |   |   |   Début
|   |       |   |   |   |   |   Ajouter CEk et ses prérequis à Lce ;
|   |       |   |   |   |   |   Ajouter CIj à Lci ;
|   |       |   |   |   |   Fin
|   |       |   |   |   Sinon
|   |       |   |   |   |   Ajouter CE à Lce ;
|   |       |   |   |   |   Ajouter CIj à Lci ;
|   |       |   |   |   Finsi
|   |       |   |   Fin
|   |       |   Finsi
|   |       Sinon
|   |       |   CI acquise ;
|   |       Finsi
```

```
        Fait
        Retourner Lce ;
Fin
```

Programma principal

```
i,j,k : entiers ;
Début
    Initialiser Lcd liste des compétences disciplinaires non acquises ;
    Initialiser Lci liste des compétences intermédiaires non acquises ;
    Initialiser Lce liste des compétences élémentaires non acquises ;
    Pour (un apprenant donné) faire
        Début
            Filtrage CDG ;
            Affichage du plan guide ; ** affichage de Lcd, Lci, Lce ;
        Fin
    Fait
Fin
```

VI.3 Exemple de filtrage des compétences :

Une compétence disciplinaire est considérée acquise quand ses compétences intermédiaires et élémentaires seront acquises ainsi que ses prérequis. Dans ce qui suit-on déroulera l'algorithme ci-dessus pour filtrer les compétences acquises et non acquises. Supposons la compétence disciplinaire « Conception des bases de données », non acquise qui a des compétences intermédiaires (composées) et élémentaires comme suit :

CI	Prérequis	CE	Prérequis
Introduction aux BDD **(acquise)**	Pas de prérequis	Définition d'une BDD Types de BDD **(acquises)**	Pas de prérequis
Démarche de conception des BDD **(acquise)**	Pas de prérequis	Elaboration du modèle E/A. Passage au modèle relationnel Implémentation sur un SGBD-R cible. **(acquises)**	Pas de prérequis
Modélisation conceptuelle **(acquise)**	Pas de prérequis	Les entités Les propriétés Types des associations Les cardinalités L'identifiant Le schéma conceptuel **(acquises)**	Pas de prérequis

Modélisation logique relationnelle **(non acquise)**	Modélisation conceptuelle **(acquise)**	Définition d'une relation Attributs Degré d'une relation Clé d'une relation Le schéma d'une relation Le domaine Occurrence d'une relation Cardinalité d'une relation Les n-uplets Contrainte de domaine Contrainte de clé Contrainte d'intégrité référentielle ou d'inclusion **(non acquises)**	Les entités Les propriétés Types des associations Les cardinalités L'identifiant Le schéma conceptuel **(acquises)**

Tableau 10 : Exemple de déroulement de l'algorithme de filtrage.

A la fin de déroulement de l'algorithme nous allons obtenir le résultat suivant :

Lcd	Lci	Lce
Conception des BDD	Modélisation logique relationnelle	Définition d'une relation Attributs Degré d'une relation Clé d'une relation Le schéma d'une relation Le domaine Occurrence d'une relation Cardinalité d'une relation Les n-uplets Contrainte de domaine Contrainte de clé Contrainte d'intégrité référentielle ou d'inclusion

Tableau 11 : Résultat de l'exemple de déroulement de l'algorithme de filtrage.

La compétence disciplinaire générale « Conception des bases de données » non acquise mais l'apprenant va revoir seulement la compétence intermédiaire « Modélisation logique relationnelle » avec ses compétences élémentaires.

Conclusion :

Dans ce chapitre, nous avons proposé la démarche de modélisation suivie pour le développement de notre application, ainsi que l'architecture et le fonctionnement de cette dernière**.** Celle-ci est basée sur une ontologie de domaine de compétences. Nous nous sommes focalisés, dans le cadre de notre travail sur les fonctions de manipulation, de la mise à jour de l'ontologie et de génération de plan guide. Le chapitre suivant traitera l'implémentation proprement dite d'un prototype du système cible sous forme d'application web utilisant la technologie web sémantique.

Introduction :

Après avoir présenté la conception et le fonctionnement global de notre application dans le chapitre précèdent, ce présent chapitre sera réservé pour la description de l'environnement de développement (outils et langages) de l'application, l'exploitation de fichier OWL dans des programmes JAVA et la présentation des interfaces essentielles montrant les déférentes fonctionnalités offertes par notre application.

I. L'architecture client/serveur :

Originellement, le terme client-serveur décrivait les interactions entre deux programmes d'une architecture logicielle. Les programmes résidaient alors sur la même machine hôte. La signification du terme client-serveur a complètement changé lors de l'apparition d'outils de distribution des traitements en fonction des ressources logicielles et matérielles. Désormais, une architecture client-serveur s'appuie sur des architectures matérielles et logicielles qui interagissent. On tente alors de séparer des problèmes de communication de bas niveau des problèmes de contrôle.

Le paradigme client-serveur consiste à structurer un système en termes d'entités clientes et d'entités serveurs qui communiquent par l'intermédiaire d'un protocole de communication à travers un réseau informatique. Ces entités interagissent de manière à assurer l'ensemble des fonctionnalités que le système fournit à son environnement, ainsi :

• un service correspond à un regroupement de fonctions liées à la gestion d'une ressource applicative, physique ou logique ;

• un serveur est une entité exécutable qui réalise, seule ou en coopérant avec d'autres entités du type serveur, les fonctionnalités du service ;

• un client correspond à une entité exécutable qui émet une requête auprès d'un serveur pour utiliser les fonctionnalités d'un service. **[25]**

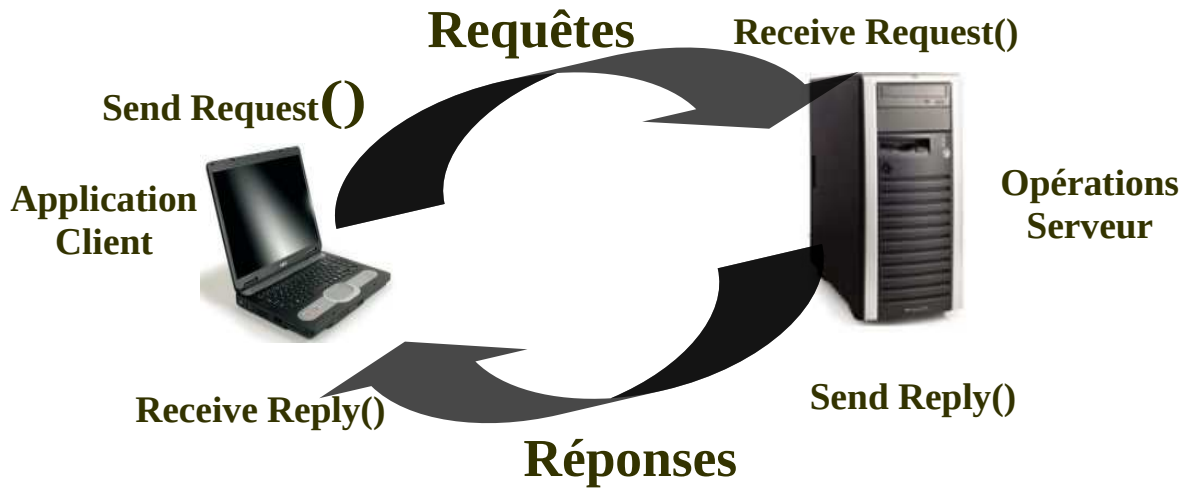

Figure 23: Schéma d'une architecture client/serveur.

I.1 Caractéristiques de l'architecture client/serveur :

L'architecture client/serveur possède plusieurs caractéristiques, à savoir:

- Le partage de ressources : plusieurs clients peuvent être « servis » simultanément.
- La transparence : la localisation des clients/serveurs est transparente aux clients et aux serveurs.
- L'échelonnage : supporte mieux une augmentation du nombre de clients.
- L'interopérabilité : les plates forme clients peuvent être hétérogènes (le lien se fait grâce à un protocole).
- La délocalisation : il y a peu ou pas de contraintes de proximité entre les clients et le serveur.

I.2 Classification des architectures client/serveur :

Plusieurs classifications des architectures client/serveur existent à ce jour. La première, est une classification issue des travaux du Gartner Group. La seconde est plus récente et prend en compte les évolutions matérielles qui influence le déploiement des applications client/serveur.

Le Gartner Group a établi un découpage en six vues distinctes montrant les différentes possibilités de répartition entre clients et serveurs des trois principales strates logicielles :

• la couche de présentation chargée de la mise en forme et de l'affichage des informations reçues;

• la couche de logique applicative mettant en œuvre les fonctions de l'application;

• la couche de gestion des données.

Une autre vision des architectures client/serveur, nous ramène à un classement en trois grandes catégories : le client/serveur de présentation, le client/serveur à deux niveaux (ou client/serveur de données) et le client/serveur à trois niveaux.

I.2.1 Le client-serveur de présentation :

Place la logique de présentation sur le poste client, mais laisse les traitements sur le serveur.

I.2.2 Architecture à deux niveaux :

L'architecture à deux niveaux (aussi appelée *architecture 2-* tiers, « tiers » signifiant *étage* en anglais) caractérise les systèmes clients/serveurs dans lesquels le client demande une ressource et le serveur la lui fournit directement. Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir le service.

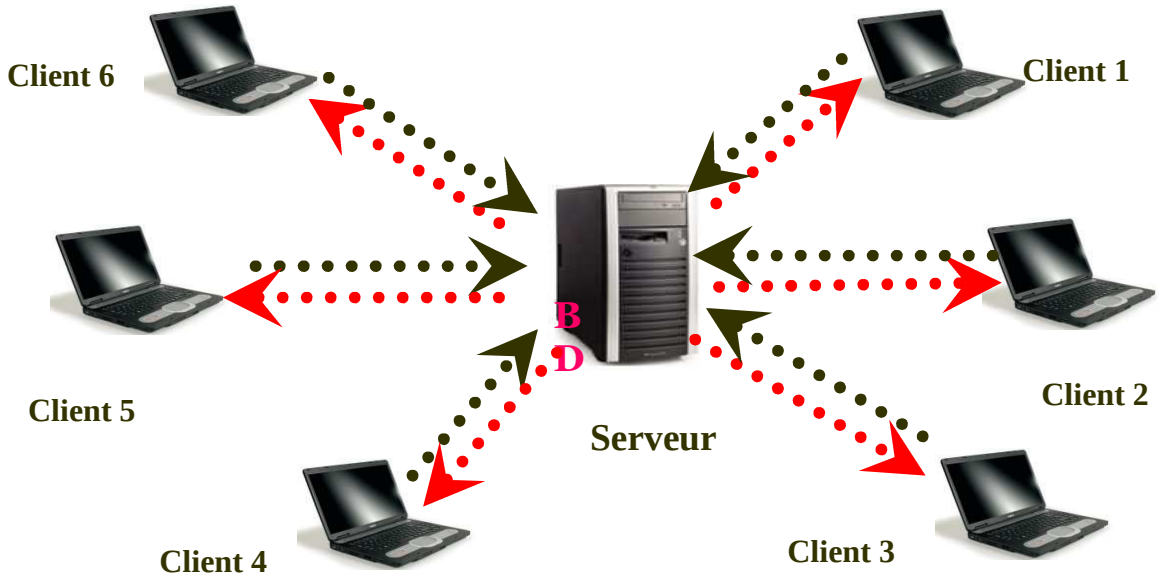

Figure 24 : L'Architecture à deux niveaux.

I.2.3 Architecture à trois niveaux :

Dans l'architecture à 3 niveaux (appelée architecture *3-tier*), il existe un niveau intermédiaire, c'est-à-dire que l'on a généralement une architecture partagée entre :

1. Un client, demandeur de ressources, équipé d'une interface utilisateur (généralement un navigateur web) chargé de la présentation ;

2. Le serveur d'application (appelé également middleware), chargé de fournir la ressource mais faisant appel à un autre serveur ;

3. Le serveur de données, fournissant au serveur d'application, les données dont il a besoin.

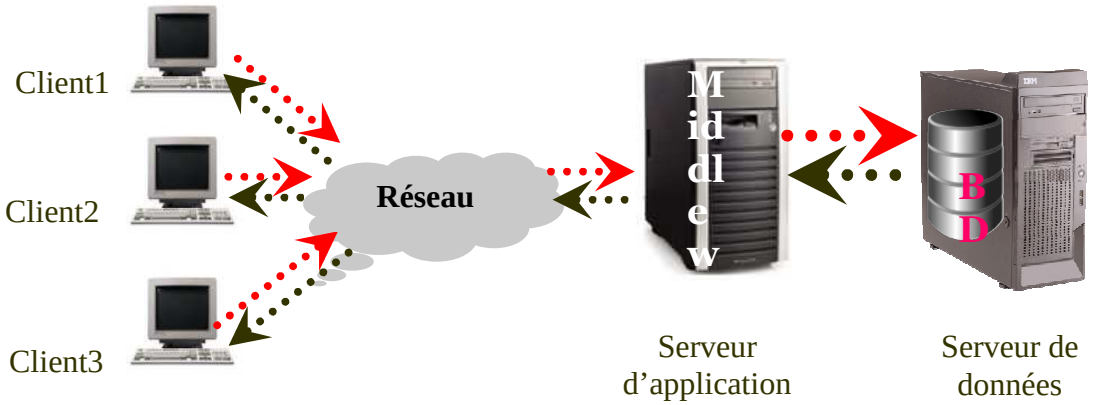

Figure 25 : L'Architecture à trois niveaux. **[25]**

II. L'architecture du système développé :

Le système a été implémenté sous forme d'une application web (un logiciel applicatif hébergé sur un serveur et accessible via un navigateur web). Dans notre cas nous avons exploité une ontologie de domaine de compétences, qui rentre dans le cadre de la nouvelle génération du web (le web sémantique). Alors pour la manipulation du fichier OWL généré à partir de cette ontologie, nous avons utilisé les Servlets qui permettent une grande flexibilité et portabilité de l'application et l'API Jena. La figure suivante présente l'architecture logicielle du système développé.

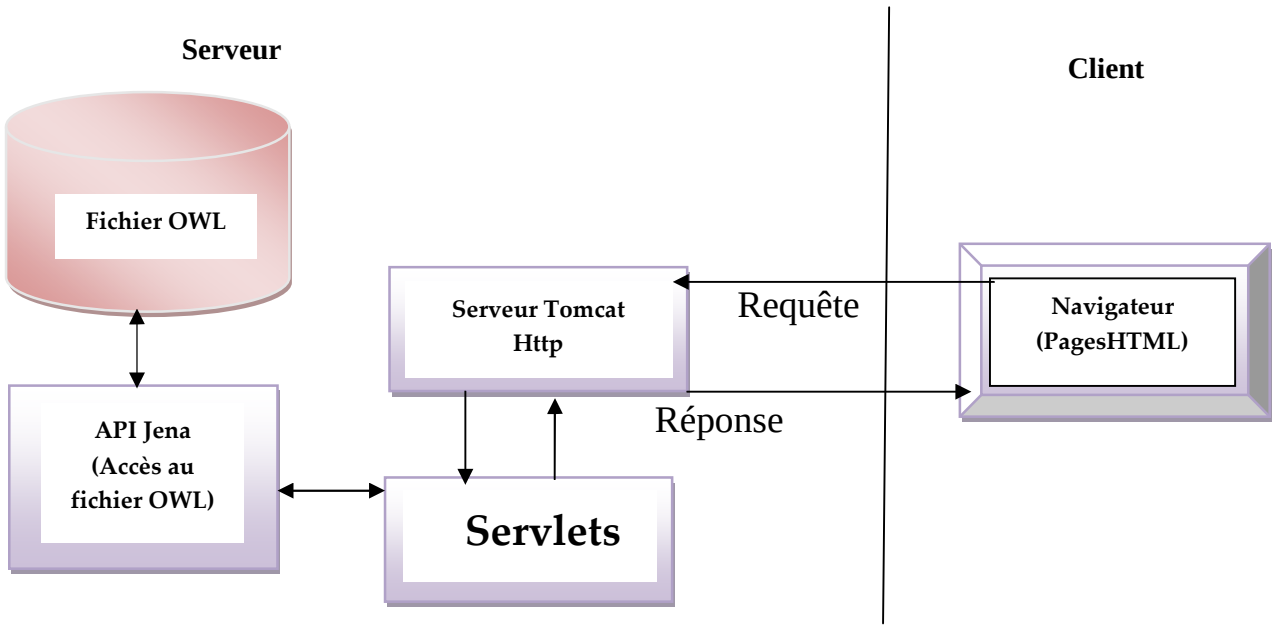

Figure 26 : L'Architecture logicielle du système développé.

III. L'environnement de développement et d'implémentation de notre application :

Dans cette section nous allons décrire l'environnement utilisé pour le développement de l'application. Le choix des différents outils et langages est fait selon le type et les fonctionnalités qui doivent être offertes par notre application.

III.1 Outils et langages :

Avant de commencer l'implémentation de l'architecture conceptuelle de notre application, nous allons tout d'abord spécifier les outils et les langages utilisés qui nous ont semblés être un bon choix de part des avantages qu'ils offrent.

III.1.1Outils :

- ✓ **Système d'exploitation :**

Notre application a été développée sous le système d'exploitation Windows 8, mais comme elle est développée en langage Java, elle peut être intégrée dans n'importe quel autre système d'exploitation supportant la machine virtuelle java (Windows 7, Windows 98/00, Linux, …).

- ✓ **Serveur d'application: Tomcat**

Le serveur d'application Apache Tomcat joue le rôle de conteneur JSP/Servlet qui permet à sa connexion avec un serveur web de délivrer du contenu dynamique aux clients. La version du serveur utilisée dans notre projet est Apache *Tomcat 8.0.9* intégré dans l'environnement Netbeans. Les raisons ayant motivé ce choix sont :

Apache Tomcat est un logiciel open source Il permet l'implémentation rapide des dernières spécifications des JSP/Servlet. Apache Tomcat est connu pour ses larges utilisations dans la communauté JAVA/J2EE en prototypage, il est ouvert et portable. En plus de ces caractéristiques le serveur apache est disponible sur pratiquement toutes les plateformes (Unix, Linux, WindowsNT et Windows 95/98 et Windows XP), gratuit et son développement est actif.

- ✓ **L'éditeur de pages web statiques et dynamiques ; Dreamweaver 8 :**

Macromedia Dreamweaver est un outil convivial et très puissant destiné à la conception, au codage et au développement de sites, de pages et d'applications Web. Quel que soit l'environnement de travail utilisé (codage manuel HTML ou environnement d'édition visuel).

Les fonctions d'édition visuelles de Dreamweaver permettent de créer rapidement des pages sans rédiger une seule ligne de code. On peut rationaliser les tâches de développement en créant et en modifiant des images dans Macromedia Fireworks ou toute autre application graphique, puis en les important directement dans Dreamweaver, ou en ajoutant des objets Flash Macromedia. Dreamweaver propose également un environnement de codage complet comprenant des outils de modification du code (comme la coloration du code et la création de balises) ainsi que des documents de référence sur les feuilles de style en cascade (CSS), Javascript, il permet aussi d'importer des documents HTML codés manuellement sans modifier le code pour pouvoir ensuite reformater ce dernier avec le style de formatage choisi par le programmeur.

Dreamweaver permet également de créer des applications Web dynamiques reposant sur des bases de données au moyen de technologies serveur comme ASP, JSP et PHP.**[27]** La figure suivante montre l'interface principale de dreamweaver 8 :

Figure 27 : Interface de Macromedia Dreamveaver.

- ✓ **L'éditeur d'ontologie :**

Protégé est un éditeur d'ontologie open source développé au département d'*Informatique Médicale de l'Université de Sandford.* L'éditeur d'ontologie utilisé pour l'édition et la génération du code OWL correspond à l'ontologie de domaine de compétences est « *Protégé version 3.4.4* ».Nous avons choisi cet éditeur pour les raisons suivantes :

Protégé est un éditeur open source et gratuit, permet d'importer et d'exporter des ontologies dans les différents langages d'implémentation (RDF-Schéma, OWL, DAML, OIL,...etc.), possède une interface modulaire ; ce qui permet son enrichissement par des modules additionnels (plugins). Il permet l'édition et la visualisation graphique des classes et des hiérarchies OWL *(Ontology Web Language)* à l'aide du logiciel *GraphViz*. Protégé contrôle la cohérence de l'ontologie par des vérifications de contraintes.**[26]**

- ✓ **L'environnement de développement ; IDE Netbeans 8.0.1 :**

Netbeans est un environnement de Développement Intégré Java (IDE) a été créé à l'initiative de Sun Microsystems. Il présente toutes les caractéristiques indispensables à un environnement de qualité, que ce soit pour développer en Java, Ruby, C/C++ ou même PHP. NetBeans est sous licence Open Source, il permet de développer et déployer rapidement et gratuitement des applications graphiques Swing, des Applets, des JSP/Servlets, des architectures J2EE, dans un environnement fortement personnalisable.

NetBeans permet de déployer des applications Web, non seulement vers Tomcat et Glassfish qui sont livrés avec le "Pack Web", mais aussi vers JBoss, WebSphere 6.1, WebLogic 9. Il détient un support de développement d'applications Web avec des améliorations pour l'édition des JSP, la gestion serveur et le support des dernières versions de Tomcat. Enfin cet IDE possède un débogueur de grande qualité ainsi qu'une interface graphique améliorée. **[9]**

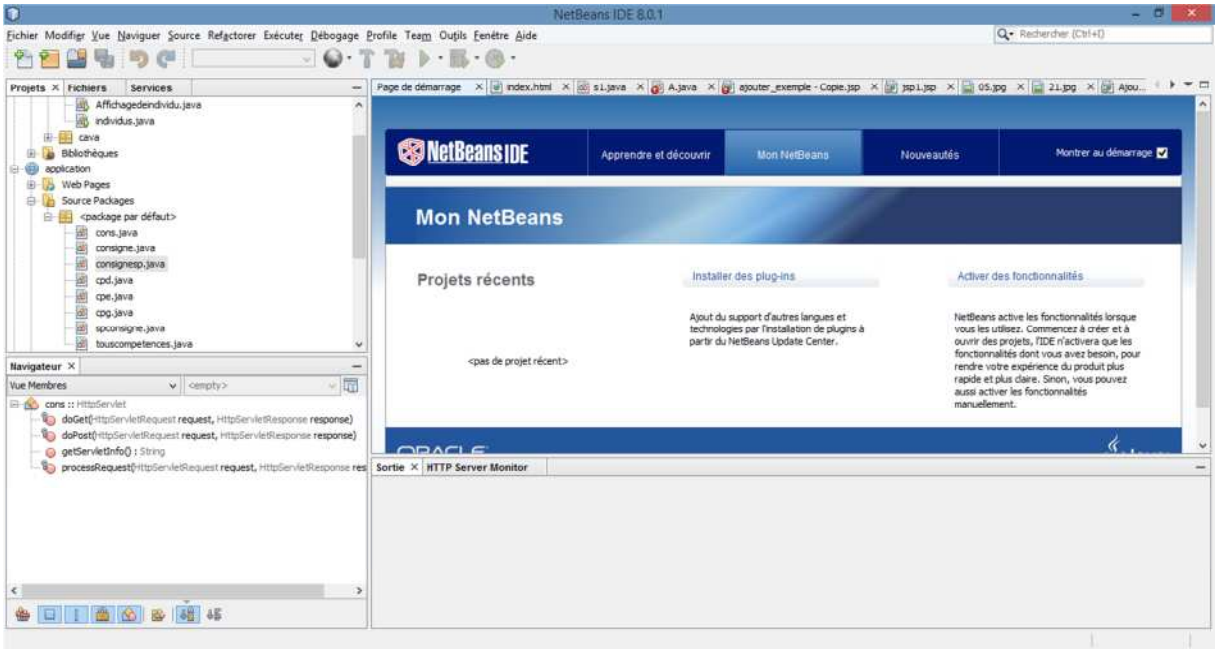

Figure 28 : L'interface Netbeans IDE8.0.1

- ✓ **JAVA Virtual Machine (JVM) :**

La machine virtuelle appelée aussi interpréteur, joue le rôle d'un traducteur. Elle traduit en ByteCode le code compilé (ensemble de fichiers de classes) qui est sous forme de pseudo-code et non en code machine, et cela afin d'être exécuté sur n'importe quelle plate-forme matérielle et logicielle : que l'on soit sur un Pentium, un PowerPC, un Sparc ou sur un Alpha, sous Windows, MacOS, Solaris ou Linux, etc. Cependant la présence de la JVM au niveau de la plate-forme est nécessaire à l'exécution des programmes JAVA sur une plate-forme quelconque.

Les JVMs sont fournies soit par le JDK (Java Developpement Kit), soit par les navigateurs ou bien par les environnements de développement spécifiques tel Borland JBuilder. Des JVM capables d'exécuter du code Java, peuvent être fournit également par certaines solutions telle Java Plug-in de Sun. La JVM peut être obtenue à partir du site de Sun Microsystems**. [27]**

- ✓ **Le Java Development Kit (JDK) :**

Le JDK est un environnement gratuit comportant un compilateur et une machine virtuelle (minimal et suffisant). Il est téléchargeable à partir du site de Sun. Il comporte l'ensemble des éléments qui ont pour but le développement, la mise au point et l'exécution des programmes Java. Il peut être considéré comme un ensemble d'outils plus un jeu de classes et de service plus un ensemble de spécifications. Il existe plusieurs versions de JDK, il est important de connaître la version employée car les classes disponibles peuvent être différentes d'une version à une autre. **[28]**

- ✓ **Application Programming Interface (API) :**

Une API fournit aux programmeurs les outils de base nécessaires afin de faciliter leur travail. Elle se compose d'un ensemble de fonctions, routines et méthodes. L'API de JAVA permet aux programmeurs d'accéder à toutes les ressources de la machine, et celles du réseau Internet. Elle est divisée en packages. Plusieurs packages sont fournis en standard avec JAVA et d'autres packages se trouvent dans d'autres applications et il faudra les importer.

L'API utilisée pour la réalisation de notre application est « Jena 2.6.2 » ; un logiciel « Open source » développé par « HP Labs Semantic Research », téléchargeable à l'adresse :http://www.hpl.hp.com/semweb/.

III.1.2 Langages :

- ✓ **HTML; HyperText Mark-up Language:**

Il a été inventé par le fondateur du web (Tim Bareners-Lee), c'est un langage universel qui s'adapte à toutes les plates forme telles que Windows, Unix, Macintosh. C'est le premier

langage utilisé pour communiquer sur le web. Le HTML n'est pas un langage de programmation dans le sens où il n'utilise pas des variables, des boucles ou des expressions conditionnelles, c'est un langage de description de documents qui utilise des marqueurs explicites (appelés aussi tags ou balises) qui précisent la structure et la mise en forme du contenu du document (texte, tableau, image, vidéo,…etc.) et établit des relations cohérentes entre ces document grâce aux liens Hypertextes.

✓ **Le langage de programmation JAVA :**

Le langage Java est issu d'un projet de Sun Microsystems datant de 1990, c'est un langage de programmation compilé et interprété, orienté objet. Il a été créé pour pallier aux contraintes que posait le C++ (développé sur un système embarqué avec des ressources limitées). La syntaxe de ce langage est assez proche du C et est plus claire que le C++. L'objectif pour lequel java a été conçu nécessite certaines caractéristiques comme : La robustesse, la compatibilité, la facilité de programmation et la petite taille du runtime ou des codes générés.
Java a donné naissance à un système d'exploitation (JavaOS), à des environnements de développement (eclipse/JDK), des machines virtuelles (MSJVM, JRE) applicatives multiplate-forme (JVM), une déclinaison pour les périphériques mobiles/embarqués (J2ME), une bibliothèque de conception d'interface graphique (AWT/Swing), des applications lourdes (Jude, Oracle SQL Worksheet, etc.), des technologies web (servlets, applets) et une déclinaison pour l'entreprise (J2EE). La portabilité du bytecode Java est assurée par la machine virtuelle Java, et éventuellement par des bibliothèques standard incluses dans un JRE. Cette machine virtuelle peut interpréter le bytecode ou le compiler à la volée en langage machine. La portabilité est dépendante de la qualité de portage des JVM sur chaque OS. **[27]**
Java est un langage :

- Orienté objet dérivé du C, simple à utiliser et plus pur que le C++, du fait qu'en java on ne peut faire que de la programmation orienté objet contrairement au C++ qui reste un langage hybride autorisant plusieurs style de programmation.
- Doté de bibliothèques de classes très riches comprenant la gestion des interfaces, la programmation multithreads (multitâches), la gestion des exceptions, les accès aux fichiers et aux réseaux.
- Doté d'un mécanisme de gestion des erreurs (les exceptions) très utile et très performant.
- Multi plates-formes : les programmes tournent sans modification sur tous les environnements où Java existe (Windows, Unix, et Mac). **[28]**

✓ Le langage SPARQL :

Comme un langage de requête, SPARQL est « orienté données » en ce sens qu'il interroge uniquement les informations détenues dans des modèles, Tout d'abord, il faut être clair sur quelles données sont interrogées. SPARQL requête des graphes RDF (un graphe RDF est un ensemble de triplets, Jena appelle les graphes RDF « modèles » et les triplets « déclarations »). Il est important de réaliser que ce sont les triplets qui importent, pas la sérialisation. La sérialisation est juste une manière de coucher par écrit les triplets.

Il est important de remarquer que les triplets dans le graphe RDF n'ont pas d'ordre particulier. Ils sont juste écrits en groupes liés, la machine ne s'en soucie pas. La Syntaxe d'une requête SPARQL est la suivante :

PREFIX nom_prefix:<prefix>
SELECT ?subject ?object
WHERE {?subject nom_prefix: predicat ?object
}

- Example d'une requête *SPARQL*

La requête suivante permet d'afficher les SP reliées à unecompétence élémentaire choisie :

PREFIX cod:<http://www.owl-ontologies.com/Ontology1431161110.owl#>
"SELECT ?A" + " WHERE {?x cod:Nom_Cpe "+"\""+classe+"\" " +". ?x cod:evalué_par ?A. }" ;

Ceci fonctionne par correspondance du modèle de triplets dans la clause WHERE sur les triplets du graphe RDF. Le prédicat et l'objet du triplet sont des valeurs fixées de sorte que le modèle va correspondre seulement aux triplets avec ces valeurs. Le sujet est une variable, et il n'y a pas d'autre restriction sur la variable. Le modèle correspond à n'importe quels triplets avec ces valeurs d'objet et de prédicat, et les faits correspondre avec les solutions pour A.

L'item inclus dans <> est une URI et l'item inclus dans "" est littéral ordinaire, les littéraux typés sont écrits avec ^^ et les étiquettes de langues peuvent être ajoutées avec @. ?x est une variable appelée x. Le ? ne forme pas une partie du nom c'est pourquoi il n'apparaît pas dans le résultat de sortie. **[29]**

- ✓ **Les Servlets :**

Une servlet est un programme java qui utilise des modules supplémentaires figurant dans l'API java. Elle s'exécute dynamiquement sur le serveur Web et permet l'extension des fonctions de serveur. Les servlets permettent donc de gérer des requêtes HTTP et de fournir au client une réponse HTTP dynamique (donc de créer des pages Web dynamiques). Une servlet s'exécute dans un conteneur de servlet utilisé pour établir le lien entre la servlet et le serveur Web, dans notre cas, on a utilisé le conteneur Tomcat version 8.0.9 Ainsi le programmeur n'a pas à se soucier des détails techniques tels que la connexion au réseau, la mise en forme de la réponse à la norme http et le retour de résultat dans différents formats (HTML, XML, RDF/XML, . . .). **[9]**

- ✓ **Les JSP (*Java Server Page*) :**

Le JSP est l'un des composants principaux de la programmation Java, qui permet de combiner le langage de marquage (HTML ou XML) avec des fragments de code Java, pour produire une page dynamique. Chaque page est automatiquement compilée en une seule Servlet (par le moteur de JSP) à sa première demande et puis exécutée. **[9]**

III.2 Présentation de quelques interfaces de l'application développée :

Après la génération du code OWL correspondant à l'ontologie de domaine de compétences, nous avons passé à son exploitation à l'aide des programmes Java dans l'environnement de développement Netbeans auquel nous avons ajouté la librairie Jena qui sert de lien entre un code OWL et un programme Java, sachant que Plus de 70% des applications de Web sémantique sont développées en Jena.

Dans ce qui suit nous allons présenter certaines interfaces de l'application illustrant les tâches réalisées par cette dernière.

III.2.1 Page d'accueil :

La page d'accueil est le point d'entrée de l'application. Elle permet à chaque type d'utilisateurs d'accéder à son espace en cliquant sur le lien qui lui correspond, ce dernier lui envoyé un formulaire d'authentification.

Figure 29 : La page d'accueil.

III.2.2 Espace expert de domaine :

L'espace expert de domaine est accessible via l'interface d'accueil. En cliquant sur le lien expert, un formulaire d'authentification s'affiche.

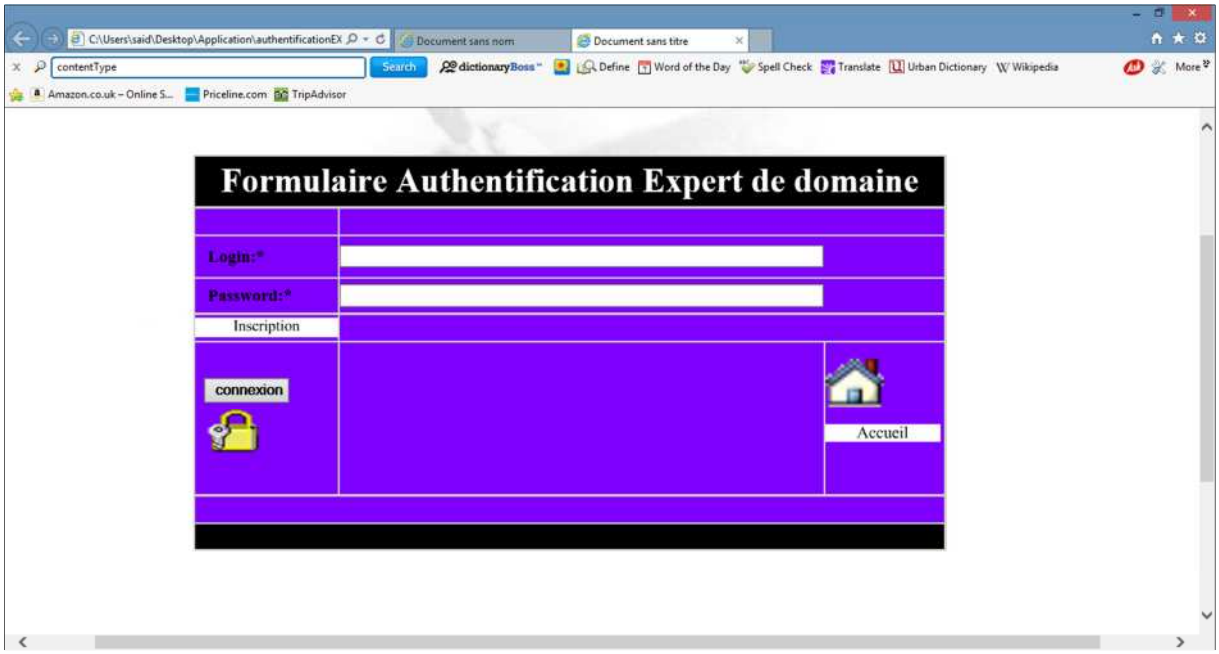

Figure 30 : Formulaire d'authentification des experts de domaine.

Si l'expert de domaine n'est pas inscrit il n'a qu'à cliquer sur le lien inscription pour s'inscrire. Un formulaire d'inscription des experts de domaine est alors affiché.

Figure 31: Espace expert de domaine.

Après avoir accéder à son espace l'expert de domaine s'occupe des tâches suivantes : La consultation et la manipulation de la base de connaissances à savoir ; l'ajout, la suppression et la modification des différents concepts (compétences disciplinaires, compétences composées, compétences élémentaires, situations problème, consignes,…etc.) ainsi que la création d'un nouveau domaine.

Figure 32 : Manipulation de la base de connaissances.

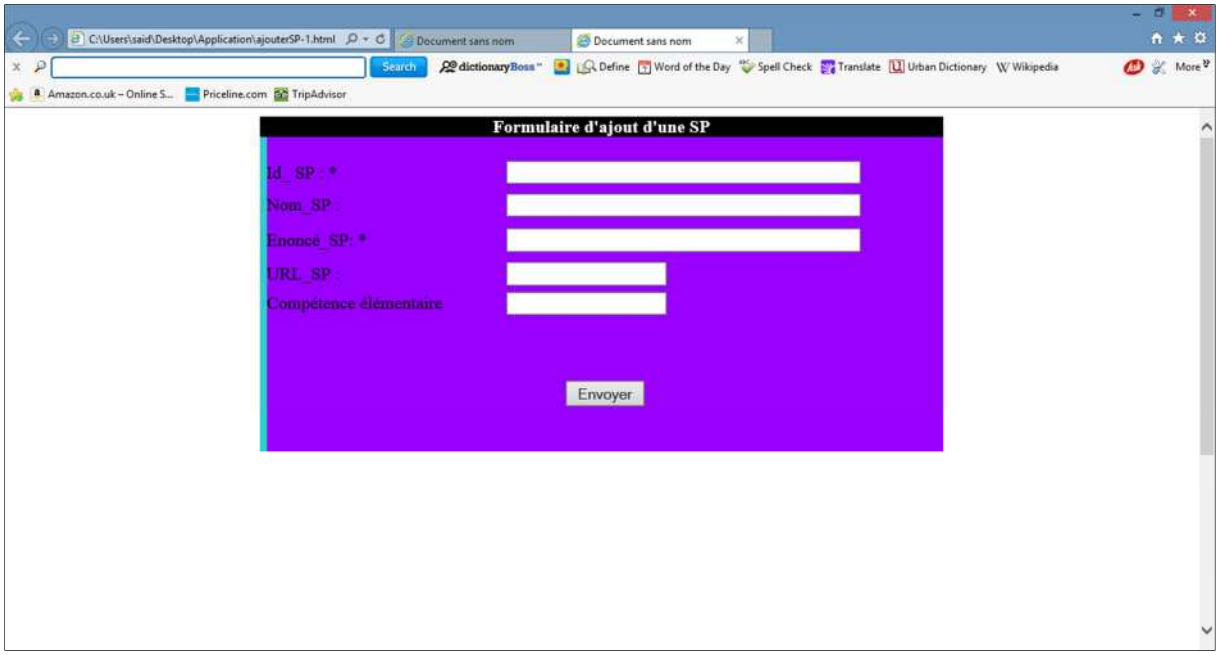

Figure 33 : Formulaire d'ajout d'une situation problème par l'expert de domaine.

Le code de la servlet suivante permet l'ajout d'un individu de la classe situation problème

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.sql.*;
import com.hp.hpl.jena.ontology.DatatypeProperty;
import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.rdf.model.Literal;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.vocabulary.OWL;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.util.Date;
public class ajoutSP extends HttpServlet {
private Object insertStatment;
private Object insertStatm;
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
```

```
String A =req.getParameter("A");
String B=req.getParameter("B");
String C=req.getParameter("C");
String D=req.getParameter("D");
String E=req.getParameter("E");
res.setContentType("text/html");
rintWriter out = res.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>La page d'ajout d'une SP</title>");
out.println("</head>");
out.println("<body>");

String IdSP=req.getParameter("A");
String NomSP=req.getParameter("B");
String EnonceSP=req.getParameter("C");
String UrlSP=req.getParameter("C");
String Nomcpe=req.getParameter("E");
file="file:/C:/Users/MEDROUH/Desktop/ontcompetence.owl";
String NL = System.getProperty("line.separator") ;


OntModel m = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM, null);
m.read(file, "RDF/XML");
try{
BufferedWriter ecrire = new BufferedWriter (new FileWriter
("C:/Users/MEDROUH/Desktop/ontcompetence.owl")) ;
OntClass SP = m.getOntClass("http://www.owl-
ontologies.com/Ontology1309777211.owl#Situation_probleme");
OntClass competence_elementaire =
m.getOntClass("http://www.owlontologies.com/Ontology1309777211.owl#
competence_elementaire");
Individual indvm = m.createIndividual("http://www.owl-
ontologies.com/Ontology1309777211.owl#"+IdSP,SP);
DatatypeProperty id = m.createDatatypeProperty("http://www.owl-
ontologies.com/Ontology1309777211.owl"+"#IdSP");
DatatypeProperty nom = m.createDatatypeProperty("http://www.owl-
ontologies.com/Ontology1309777211.owl"+"#NomSP");
DatatypeProperty enonce = m.createDatatypeProperty("http://www.owl-
ontologies.com/Ontology1309777211.owl"+"#EnonceSP");
DatatypeProperty url = m.createDatatypeProperty("http://www.owl-
ontologies.com/Ontology1309777211.owl"+"#UrlSP");
Individual indv = m.createIndividual("http://www.owl-
ontologies.com/Ontology1309777211.owl#"+Nomcpe,competence_elementaire);
DatatypeProperty nomex = m.createDatatypeProperty("http://www.owl-
ontologies.com/Ontology1309777211.owl"+"#Nomcpe");
indvm.addProperty(id,IdSP );
indvm.addProperty(nom, NomSP);
indv.addProperty(enonce, EnonceSP);
indv.addProperty(url, UrlSP);
indv.addProperty(nomex, Nomcpe);
Property est_relie_a=m.createProperty("http://www.owl-
ontologies.com/Ontology1309777211.owl","#est_relie_a");
```

Création d'un modèle d'ontologie

Lecture du fichier owl dans le modèle

L'emplacement physique du fichier après instanciation

Récupération de la classe SP

```
Property possede_exercice=m.createProperty("http://www.owl-
ontologies.com/Ontology1309777211.owl","#possede_exercice");

Model writeAll = m.write(ecrire, "RDF/XML-ABBREV");
out.println("<h1> la SP est ajoutee</h1>");
while (individuals.hasNext()) {
Individual individual = (Individual) individuals.next();
System.out.println(individual);*/
ecrire.close();

} catch (Exception e) {
System.out.println("ERROR :"+e.getMessage());
}
}}
```

Figure 34 : Le code de la servlet permet l'ajout d'une SP.

Figure 35 : Les classes et sous classes de l'ontologie consulter par l'expert de domaine.

III.2.3 Espace Apprenant :

Tout comme l'expert de domaine l'apprenant possède un espace accessible après authentification, s'il est inscrit sinon une inscription doit être effectuée. La figure ci-dessous montre l'espace apprenant.

Figure 36 : L'espace apprenant

Dans son espace l'apprenant peut interroger la base de connaissances pour voir les différentes compétences à acquérir dans sa discipline, comme il peut s'informer de son état d'avancement à l'aide du plan guide.

Figure 37 : L'interrogation de la base de connaissances.

Figure 38 : Choix d'une compétence disciplinaire générale pour voir ses compétences composées.

Figure 39 : Les compétences composées d'une compétence disciplinaire générale choisie.

Figure 40 : Génération du plan guide.

Les figures suivantes illustrent le cheminement suivi pour afficher une situation problème d'une compétence choisi par l'apprenant.

Figure 41 : Le choix d'une compétence élémentaire pour l'évaluer.

Figure 42: Affichage des SP liée à une compétence donnée.

Le code de la servlet suivante permet d'afficher les situations problème liées à une compétence élémentaire donnée :

```
import java.io.IOException;
import javax.servlet.ServletException;
import java.io.PrintWriter;
import java.util.ArrayList;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.vocabulary.RDF;
public class spliecpe extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException{
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>page1</title>");
out.println("</head>");
out.println("<body background=\"image/bkg_marb.jpg\" text=\"#000FFF\"
link=\"#0000FF\"vlink=\"#0000FF\" alink=\"#FF0000\">");
out.println("<img src=\"image/bienvenu_entre.jpg\" width=\"1235\" height=\"180\" border=\"0\"alt=\"\">");
```

```
String  classe= request.getParameter("classes");
out.println( "<center>");
String tod_file = "ontcompetence.owl";
String NL = System.getProperty("line.separator") ;
Model model = ModelFactory.createDefaultModel() ;
model.read("file:/E:/ontcompetence.owl");
String tod ="http://www.owl-ontologies.com/Ontology1309777211.owl#";
String prolog1 = "PREFIX tod: <"+tod+">" ;
String prolog2 = "PREFIX rdf: <"+RDF.getURI()+">" ;

String queryString = prolog1 + NL + prolog2 + NL +
"SELECT ?B"  + " WHERE {?x tod:Nom_cpe   "+"\"" +"classe"+"\"    " +".   ?x    tod:Est_evalue
?B.}";
Query query = QueryFactory.create(queryString) ;
query.serialize() ;
System.out.println() ;
QueryExecution qexec = QueryExecutionFactory.create(query, model) ;
out.println("<marquee behavior=\"alternate\"><h1><center>les items de connaissance :"+classe+"sont :
</center></h1></marquee>") ;
out.println(" <div align=\"center\"><table border=\"0\" cellpadding=\"10\"cellspacing=\"80\">");
out.println("<tr> <td><b> ");
try {
ResultSet rs = qexec.execSelect() ;

for ( ; rs.hasNext() ; ) {
QuerySolution rb = rs.nextSolution() ;
RDFNode y = rb.get("B");
Resource z = (Resource) rb.getResource("B");
out.println(z.getLocalName());
out.println("<br>");
}}finally {
}qexec.close() ;
out.println( "</center>");
out.println("</head>");
out.println("</html>");
}}
```

Requête SPARQL permettant l'affichage des SP liée à une compétence élémentaire donnée

Figure 43 : Le code de servlet qui permet d'afficher les SP liées à une compétence élémentaire donnée.

Figure 44 : Le choix d'une compétence pour voir ses prérequis.

Figure 45 : Affichage des prérequis de la compétence sélectionnée.

Conclusion :

Dans ce chapitre nous avons présenté l'implémentation de notre système qu'est une application web. Nous avons tout d'abord montré l'environnement de développement ainsi que les différents outils utilisés, puis nous avons donné une description détaillée de celle-ci à travers les interfaces qui sont conçues de manière à être conviviales et simples à utiliser. Cette étape nous a aussi permis de nous familiariser avec les outils employés pour le développement de notre application.

Conclusion générale :

Ce mémoire de fin d'études a présenté les principaux langages développés par le World Wide Web Consortium dans le cadre du Web Sémantique ainsi les outils utilisés par ce dernier tel que les ontologies apparaissant comme une clé pour la manipulation automatique de l'information à l'aide des technologies du web sémantique. Nous avons ensuite étudié l'approche par compétences, qui se manifeste comme nouvelle approche pédagogique adoptée dans les systèmes de l'enseignement et de l'apprentissage, cette étude avait pour objectif de rechercher et de comprendre les différents concepts (classes et relations) utilisés lors de la modélisation du domaine de compétences à l'aide d'une ontologie de domaine.

La finalité de notre travail est la manipulation et l'exploitation d'une ontologie de domaine de compétences avec la technologie de web sémantique, le travail est débuté par la réorganisation du diagramme de classes UML modélisant le domaine de compétences selon nos objectifs, ensuite nous avons passé à l'édition de l'ontologie de domaine, en utilisant l'éditeur d'ontologie Protégé afin de pouvoir générer automatiquement le code OWL. Ce fichier OWL est produit par la création des classes, des relations (ObjectProperty) et des attributs (DatatypeProperty). Dans notre travail nous avons intéressé aux compétences disciplinaires pour le module de bases de données, la génération de la base de connaissances est basée sur l'instanciation (création des individus) du modèle ontologique proposé.

Une fois la base de connaissances est créée, nous avons passé à son exploitation dans le cadre d'une application web destinée à l'acquisition de connaissances. Cette application est conçue dans l'environnement de programmation NetBeans après l'intégration de l'API Jena qui permet l'interrogation et la manipulation de la base de connaissances. Cette application permet essentiellement de :

- L'interrogation de la base de connaissances à l'aide des requêtes SPARQL qui se portent sur les compétences disciplinaires, les compétences composées, les compétences élémentaires, les situations problème, les obstacles et les consignes ;
- L'instanciation de la base de connaissances (création de nouveaux individus) ;
- La filtration des compétences selon les relations « prérequis » et « acquis ».

Ce travail permet de donner une vue globale sur les ontologies et les technologies du Web Sémantique qui peuvent apporter un très grand avantage pour les applications web destinées

surtout pour l'enseignement et l'apprentissage. Cependant ce travail peut être amélioré sur plusieurs axes à savoir :

- L'exploitation des règles d'inférence dans le raisonnement pour la recherche de ressources ;
- La visualisation graphique de l'ontologie de domaine de compétences dans un environnement Java.

Bibliographie :

[1] Anne-Sophie Feyaerts. Raisonnement sur les ontologies spatiales. Faculté des Sciences Appliquées Département CoDE. Université libre Bruxelles. Université d'Europe, 2009/2010.

[2] Jérôme Euzenat, Raphaël Troncy. Web sémantique et pratiques documentaires. 1INRIA Rhône-Alpes, INA. France (2005).

[3] introduction au web sémantique. http://en.wikipedia.org/wiki/Semantic_Web.

[4] Introduction au web sémantique. Michel Gagnon. École polytechnique. Montréal.

[5] www.futura-sciences.com/magazines/.../internet-**web-semantique**-3993/ .

[6] Sabri Boutemedjet. Web sémantique et e-Learning. Cours IFT6261. Université de montreal, hiver 2004.

[7] Alain Léger, Jean Charlet. Application du web sémantique. France Telecom R&D, Mission de recherche STIM, AP-HP & INSERM ERM 0202.

[8] http://dspace.univ-tlemcen.dz/bitstream/112/1062/5/ChapitreI.pdf.

[9] Medrouh Kahina, Yahi Djamila. Exploitation d'une ontologie de domaine d'enseignement en utilisant les outils du web sémantique. Université Mouloud Mammeri de Tizi-Ouzou, 2010/2011.

[10] Jean Charlet, Bruno Bachimont, Raphaël Troncy. Ontologies pour le web sémantique. Mission de recherche STIM, AP-HP & INSERM ERM 0202, Institut National de l'Audiovisuel, Université Technologique de Compiègne ISTI-CNR.

[11] Anastassis Kozanitis. Les principaux courants théoriques de l'enseignement et de l'apprentissage : un point de vue historique. École Polytechnique, Bureau d'appui pédagogique, Septembre 2005.

[12] Gérard Barnier, formateur. Théories de l'apprentissage et pratiques d'enseignement. IUFM d'Aix-Marseille.

[13] Berkane, T. Bouarab-Dahmani, F. Modélisation ontologique des compétences pour un hypermédia adaptatif éducatif, 2013.

[14] Jean-Marie De Ketele. François-Marie Gerard. La validation des épreuves d'évaluation selon l'approche par les compétences, 2005.

[15] https://pedagogieuniversitaire.wordpress.com/2010/03/22/les-grilles-devaluation-criteriees/

[16] http://esr.csdccs.edu.on.ca/programmes-du-ib/bi-programme-du-ppcs/evaluation-criteriee/

[17] Cylia BOUHADDA, Tania MAKRI. Développement d'un service web pour l'aide à la gestion d'évaluation critériée. Université Mouloud Mammeri de Tizi-Ouzou, 2013/2014.

[18] Denis Berthiaume, Jérôme David et Thomas David. Réduire la subjectivité lors de l'évaluation des apprentissages à l'aide d'une grille critériée : repères théoriques et applications à un enseignement interdisciplinaire, Revue internationale de pédagogie de l'enseignement supérieur, 2011.

[19] Mahiddine Mehanna , Missoum Mehenna . Conception et réalisation d'une ontologie dans le domaine des hydrocarbures pour la recherche d'information. I.N.I, 2007.

[20] Michel Héon. Transformation of Semi-formal Models into. Télé-Université du Québec, Ontologies, michel.heon@licef.teluq.uqam.ca.

[21] http://protege.cim3.net/cgi-bin/wiki.pl%3FSWRLLanguageFAQ

[22] Dana Torres, Luis Gonzalez, Sara Tassini. Cours de Bases de Données. http://tecfaetu.unige.ch/staf/staf-h/tassini/staf2x/Heidi/last_bd.htm.

[23] Guillaume Cabanac. Programmation et administration des bases de données. Université Toulouse III, version Decémbre 2014.

[24] http://web-semantique.developpez.com/tutoriels/jena/arq/introduction-sparql.

[25] Olivier GLÜCK. Architecture et communications Client/Serveur. Université LYON 1/Département Informatique 2014.

[26] Véronique Giudicelli. Ontologies et l'éditeur Protégé - Application à la formalisation des concepts de description d'IMGT-ONTOLOGY, Mai 2010.

[27] http://www.iandickinson.me.uk/articles/jena-eclipse-helloworld/

[28] Yahia Mohamed, Bechiri Marzouk. Développement d'un système d'aide à l'apprentissage de compétences disciplinaires. Université Mouloud Mammeri de Tizi-Ouzou, 2013/2014.

[29] http://web-semantique.developpez.com/tutoriels/jena/arq/introduction-sparql

Le code OWL correspond à l'ontologie de domaine de compétence est le suivant :

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:tod="http://www.owl-
ontologies.com/Ontology1431161110.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xml:base="http://www.owl-ontologies.com/Ontology1431161110.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="obstacle_general">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="obstacle"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="consigne"/>
  <owl:Class rdf:ID="competence_elementaire">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="competence_disciplinaire"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="competence_composee">
    <rdfs:subClassOf rdf:resource="#competence_disciplinaire"/>
  </owl:Class>
  <owl:Class rdf:ID="obstacle_discilplinaire">
    <rdfs:subClassOf rdf:resource="#obstacle"/>
  </owl:Class>
  <owl:Class rdf:ID="situation_probleme"/>
  <owl:Class rdf:ID="evaluation_criteriee">
    <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="possede2"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="a_obstacle_disciplinaire"/>
        </owl:onProperty>
      </owl:Restriction>
```

```
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="est_relie1"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="prerequis"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="evite"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="est_compose_de2"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="possede1"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
```

```
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="est_defini2"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="est_relie2"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="est_evite"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="evalue_par"/>
        </owl:onProperty>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="est_defini1"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="est_compose_de1"/>
        </owl:onProperty>
        <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

```
  <owl:ObjectProperty rdf:about="#prerequis">
    <rdfs:domain rdf:resource="#competence_disciplinaire"/>
    <rdfs:range rdf:resource="#competence_disciplinaire"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#est_defini1">
    <rdfs:range rdf:resource="#competence_elementaire"/>
    <rdfs:domain rdf:resource="#situation_probleme"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#est_defini2">
    <rdfs:domain rdf:resource="#consigne"/>
    <rdfs:range rdf:resource="#situation_probleme"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#evite">
    <rdfs:domain rdf:resource="#consigne"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#est_evite"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#obstacle"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#a_obstacle_disciplinaire">
    <rdfs:range rdf:resource="#obstacle_discilplinaire"/>
    <rdfs:domain rdf:resource="#competence_disciplinaire"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#est_relie2">
    <rdfs:domain rdf:resource="#obstacle_discilplinaire"/>
    <rdfs:range rdf:resource="#competence_disciplinaire"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#est_evite">
    <rdfs:range rdf:resource="#consigne"/>
    <rdfs:domain rdf:resource="#obstacle"/>
    <owl:inverseOf rdf:resource="#evite"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#est_relie1">
    <rdfs:domain rdf:resource="#obstacle"/>
    <rdfs:range rdf:resource="#situation_probleme"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#est_compose_de2">
    <rdfs:domain rdf:resource="#competence_composee"/>
    <rdfs:range rdf:resource="#competence_elementaire"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#est_compose_de1">
    <rdfs:range rdf:resource="#competence_composee"/>
    <rdfs:domain rdf:resource="#competence_composee"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#evalue_par">
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#situation_probleme"/>
          <owl:Class rdf:about="#evaluation_criteriee"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:domain rdf:resource="#competence_elementaire"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#possede1">
```

```
    <rdfs:domain rdf:resource="#situation_probleme"/>
    <rdfs:range rdf:resource="#obstacle"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#possede2">
    <rdfs:domain rdf:resource="#situation_probleme"/>
    <rdfs:range rdf:resource="#consigne"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="id_cn">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#consigne"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="degre_imp">
    <rdfs:domain rdf:resource="#evaluation_criteriee"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_sp">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#situation_probleme"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nom_cpd">
    <rdfs:domain rdf:resource="#competence_disciplinaire"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="enonce_sp">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#situation_probleme"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nom_obg">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#obstacle_general"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="desc_cpc">
    <rdfs:domain rdf:resource="#competence_composee"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_obg">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#obstacle_general"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_cpd">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#competence_disciplinaire"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="desc_cpe">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#competence_elementaire"/>
```

```
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="url_cn">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#consigne"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="enonce_cn">
    <rdfs:domain rdf:resource="#consigne"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nom_sp">
    <rdfs:domain rdf:resource="#situation_probleme"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="desc_obd">
    <rdfs:domain rdf:resource="#obstacle_discilplinaire"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_ob">
    <rdfs:domain rdf:resource="#obstacle"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nom_ob">
    <rdfs:domain rdf:resource="#obstacle"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nom_obd">
    <rdfs:domain rdf:resource="#obstacle_discilplinaire"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nom_cpc">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#competence_composee"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="desc_obg">
    <rdfs:domain rdf:resource="#obstacle_general"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nom_cn">
    <rdfs:domain rdf:resource="#consigne"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_obd">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#obstacle_discilplinaire"/>
  </owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:ID="id_cpc">
    <rdfs:domain rdf:resource="#competence_composee"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="url_sp">
    <rdfs:domain rdf:resource="#situation_probleme"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id_cpe">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#competence_elementaire"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="nom_cpe">
    <rdfs:domain rdf:resource="#competence_elementaire"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="degre_per_max">
    <rdfs:domain rdf:resource="#evaluation_criteriee"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <tod:competence_composee rdf:ID="algebre_relatinnel">
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >'algebre_relatinnel</tod:nom_cpc>
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</tod:id_cpc>
  </tod:competence_composee>
  <rdf:Description rdf:ID="competence_elementaire_95">
    <tod:id_cpe rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >31</tod:id_cpe>
  </rdf:Description>
  <tod:competence_composee rdf:ID="controle_des_droits_des_acces">
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >11</tod:id_cpc>
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >controle_des_droits_d'acces</tod:nom_cpc>
  </tod:competence_composee>
  <tod:competence_composee rdf:ID="les_dependances_fonctionnelles">
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="DF_directe">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >DF_directe</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >26</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="proprietes_des_DF">
```

```
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >23</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >proprietes_des_DF</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="DF_canonique">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >DF_canonique</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >25</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="definition_de_une_DF">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >definition_de_une_DF</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >21</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >5</tod:id_cpc>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="couverture_minimale">
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >29</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >couverture_minimale</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="fermeture_transitive">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >fermeture_transitive</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >28</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >les_dependances_fonctionnelles</tod:nom_cpc>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="DF_elementaire">
        <tod:id_cpe
```

```
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >DF_elementaire</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="graphe_des_DF">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >graphe_des_DF</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >27</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="cle_de_une_relation">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >cle_de_une_relation</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >22</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
  </tod:competence_composee>
  <tod:competence_composee rdf:ID="controle_des_acces_concurrents">
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >12</tod:id_cpc>
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >controle_des_acces_concurrents</tod:nom_cpc>
  </tod:competence_composee>
  <tod:situation_probleme rdf:ID="situation_probleme_2">
    <tod:id_sp rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >2</tod:id_sp>
    <tod:enonce_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Soit la base de donnees representee par les relations suivantes
: Boutique (NomB, VilleB, QuartierB) Client (NomC, rueC, villeC)
Employe (NomE, NomB, salaireE) Achat (NomC, NomB, NomE, montant,
date, heure) 1. Designer la cle primaire, la(es) cle(s) secondaire
(s) eventuelle (s) et la ou les cles etrangeres pour chaque relation
? 2. Formuler les expressions algebriques pour repondre aux requetes
suivantes : ï€  La liste des boutiques du quartier Â« Champlain Â»
     La liste des noms des boutiques de la ville Sainte-Foy     Nom
des clients de la rue Rene-levesque et qui sont employe de une
boutique   Afficher les noms des boutiques qui ont vendu quelque
chose      La liste des noms des clients qui nâ€™ont
</tod:enonce_sp>
    <tod:nom_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >situation_probleme_2</tod:nom_sp>
  </tod:situation_probleme>
```

```
  <tod:competence_composee rdf:ID="demarche_de_conception_des_bdd">
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >demarche_de_conception_des_bdd</tod:nom_cpc>
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >2</tod:id_cpc>
    <tod:desc_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></tod:desc_cpc>
  </tod:competence_composee>
  <rdf:Description rdf:ID="competence_elementaire_94">
    <tod:id_cpe rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >30</tod:id_cpe>
  </rdf:Description>
  <rdf:Description rdf:ID="competence_elementaire_82">
    <tod:id_cpe rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >17</tod:id_cpe>
  </rdf:Description>
  <tod:competence_composee rdf:ID="modelisation_conceptuelle">
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="les_entites">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >les_entites</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >6</tod:id_cpe>
        <tod:evalue_par>
          <tod:situation_probleme rdf:ID="situation_probleme_1">
            <tod:possede1>
              <tod:obstacle_discilplinaire
rdf:ID="obstacle_discilplinaire_53">
                <tod:est_relie1
rdf:resource="#situation_probleme_1"/>
                <tod:id_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >5</tod:id_obd>
                <tod:desc_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                ></tod:desc_obd>
                <tod:nom_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >obstacle_discilplinaire_53</tod:nom_obd>
                <tod:est_evite>
                  <tod:consigne rdf:ID="consigne_5">
                    <tod:est_defini2
rdf:resource="#situation_probleme_1"/>
                    <tod:evite
rdf:resource="#obstacle_discilplinaire_53"/>
                    <tod:id_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                    >0</tod:id_cn>
                    <tod:enonce_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    ></tod:enonce_cn>
                    <tod:nom_cn
```

```
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    ></tod:nom_cn>
                  </tod:consigne>
                </tod:est_evite>
              </tod:obstacle_discilplinaire>
            </tod:possede1>
            <tod:possede1>
              <tod:obstacle_discilplinaire
rdf:ID="obstacle_discilplinaire_35">
                <tod:nom_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >obstacle_discilplinaire_35</tod:nom_obd>
                <tod:id_ob
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >0</tod:id_ob>
                <tod:est_evite>
                  <tod:consigne rdf:ID="consigne_3">
                    <tod:id_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                    >3</tod:id_cn>
                    <tod:enonce_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >3.    Extraire les liens qui attachent entre
deux ou plusieurs entitÃ©s prÃ©sentant un intÃ©rÃªt pour le rÃ©el
perÃ§u.</tod:enonce_cn>
                    <tod:evite
rdf:resource="#obstacle_discilplinaire_35"/>
                    <tod:nom_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >consigne_3</tod:nom_cn>
                    <tod:est_defini2
rdf:resource="#situation_probleme_1"/>
                  </tod:consigne>
                </tod:est_evite>
                <tod:est_relie1
rdf:resource="#situation_probleme_1"/>
                <tod:desc_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >3.  Comment savoir quelles sont les associations
reliant les entitÃ©s.</tod:desc_obd>
                <tod:id_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >3</tod:id_obd>
              </tod:obstacle_discilplinaire>
            </tod:possede1>
            <tod:est_defini1 rdf:resource="#les_entites"/>
            <tod:enonce_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Un editeur souhaite installer une BDD pour memoriser
les informations suivantes : Les livres sont identifies par leur
numero ISBN. Un livre possede un titre et un prix de vente. Il est
ecrit par un ou plusieurs auteurs, chaque livre est tire en une ou
plusieurs editions, datees et identifiees par leur ordre. Chaque
edition comporte un certain nombre de exemplaires. Les auteurs sont
identifies par leur nom, prenom et peuvent avoir un pseudonyme. Pour
chaque livre, un auteur perÃ§oit des droits de auteur, calcules
```

```
comme pourcentage du prix de vente. Les librairies identifiees par
leur nom et adresse ; peuvent envoyer des commandes de un ou
plusieurs livres en quantite quelconque.       Elaborer le schema
entite-association pour la BDD de la maison de edition ?
</tod:enonce_sp>
            <tod:est_defini1>
              <tod:competence_elementaire rdf:ID="proprietes">
                <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >proprietes</tod:nom_cpe>
                <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >7</tod:id_cpe>
                <tod:evalue_par
rdf:resource="#situation_probleme_1"/>
              </tod:competence_elementaire>
            </tod:est_defini1>
            <tod:possede2>
              <tod:consigne rdf:ID="consigne_2">
                <tod:enonce_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >2.  Lâ€™identifiant est une propriÃ©tÃ© ou groupe
de propriÃ©tÃ©s dâ€™une entitÃ© ou dâ€™une association qui permet de
distinguer (voir dâ€™une faÃ§on unique) ces
derniÃ¨res</tod:enonce_cn>
                <tod:id_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >2</tod:id_cn>
                <tod:nom_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >consigne_2</tod:nom_cn>
                <tod:evite>
                  <tod:obstacle_discilplinaire
rdf:ID="obstacle_discilplinaire_28">
                    <tod:est_relie1
rdf:resource="#situation_probleme_1"/>
                    <tod:id_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                    >2</tod:id_obd>
                    <tod:desc_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >2.    Comment dÃ©terminer lâ€™identifiant.
</tod:desc_obd>
                    <tod:nom_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >obstacle_discilplinaire_28</tod:nom_obd>
                    <tod:est_evite rdf:resource="#consigne_2"/>
                  </tod:obstacle_discilplinaire>
                </tod:evite>
                <tod:est_defini2
rdf:resource="#situation_probleme_1"/>
              </tod:consigne>
            </tod:possede2>
            <tod:possede2>
              <tod:consigne rdf:ID="consigne_1">
                <tod:est_defini2
```

```
rdf:resource="#situation_probleme_1"/>
                <tod:nom_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >consigne_1</tod:nom_cn>
                <tod:evite>
                  <tod:obstacle_discilplinaire
rdf:ID="obstacle_discilplinaire_11">
                    <tod:nom_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >obstacle_discilplinaire_11</tod:nom_obd>
                    <tod:id_ob
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                    >0</tod:id_ob>
                    <tod:est_evite rdf:resource="#consigne_1"/>
                    <tod:est_relie1
rdf:resource="#situation_probleme_1"/>
                    <tod:nom_ob
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    ></tod:nom_ob>
                    <tod:desc_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >1.    Comment extraire les entitÃ©s et quelles
sont les entitÃ©s quâ€™on va prendre en considÃ©ration.
</tod:desc_obd>
                    <tod:id_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                    >1</tod:id_obd>
                  </tod:obstacle_discilplinaire>
                </tod:evite>
                <tod:id_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >1</tod:id_cn>
                <tod:enonce_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >1.  Extraire tous les objets concrets et abstraits
qui ont une existence propre et prÃ©sentant un intÃ©rÃªt pour le
rÃ©el perÃ§u.</tod:enonce_cn>
              </tod:consigne>
            </tod:possede2>
            <tod:possede1
rdf:resource="#obstacle_discilplinaire_28"/>
            <tod:est_defini1>
              <tod:competence_elementaire rdf:ID="cardinalites">
                <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >9</tod:id_cpe>
                <tod:evalue_par
rdf:resource="#situation_probleme_1"/>
                <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >cardinalites</tod:nom_cpe>
              </tod:competence_elementaire>
            </tod:est_defini1>
            <tod:possede2 rdf:resource="#consigne_5"/>
            <tod:id_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
```

```
            >1</tod:id_sp>
            <tod:possede1>
              <tod:obstacle_discilplinaire
rdf:ID="obstacle_discilplinaire_44">
                <tod:desc_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >4.  Comment dÃ©finir les
cardinalitÃ©s.</tod:desc_obd>
                <tod:est_evite>
                  <tod:consigne rdf:ID="consigne_4">
                    <tod:evite
rdf:resource="#obstacle_discilplinaire_44"/>
                    <tod:est_defini2
rdf:resource="#situation_probleme_1"/>
                    <tod:enonce_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >4.    La cardinalitÃ© est le nombre de fois
minimum et le nombre de fois maximum quâ€™une mÃªme occurrence
dâ€™une entitÃ© peut intervenir dans les occurrences dâ€™une
association.</tod:enonce_cn>
                    <tod:id_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                    >4</tod:id_cn>
                    <tod:nom_cn
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >consigne_4</tod:nom_cn>
                  </tod:consigne>
                </tod:est_evite>
                <tod:id_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >4</tod:id_obd>
                <tod:est_relie1
rdf:resource="#situation_probleme_1"/>
                <tod:nom_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >obstacle_discilplinaire_44</tod:nom_obd>
              </tod:obstacle_discilplinaire>
            </tod:possede1>
            <tod:nom_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >situation_probleme_1</tod:nom_sp>
            <tod:possede1
rdf:resource="#obstacle_discilplinaire_11"/>
            <tod:possede2 rdf:resource="#consigne_4"/>
            <tod:possede2 rdf:resource="#consigne_3"/>
            <tod:est_defini1>
              <tod:competence_elementaire
rdf:ID="schema_conceptuel">
                <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >11</tod:id_cpe>
                <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >schema_conceptuel</tod:nom_cpe>
                <tod:evalue_par
rdf:resource="#situation_probleme_1"/>
```

```
              </tod:competence_elementaire>
            </tod:est_defini1>
            <tod:est_defini1>
              <tod:competence_elementaire rdf:ID="les_associations">
                <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >les_associations</tod:nom_cpe>
                <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >8</tod:id_cpe>
                <tod:evalue_par
rdf:resource="#situation_probleme_1"/>
              </tod:competence_elementaire>
            </tod:est_defini1>
            <tod:est_defini1>
              <tod:competence_elementaire rdf:ID="identifiant">
                <tod:evalue_par
rdf:resource="#situation_probleme_1"/>
                <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >10</tod:id_cpe>
                <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >l'identifiant</tod:nom_cpe>
              </tod:competence_elementaire>
            </tod:est_defini1>
          </tod:situation_probleme>
        </tod:evalue_par>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:id_cpd rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</tod:id_cpd>
    <tod:est_compose_de2 rdf:resource="#cardinalites"/>
    <tod:est_compose_de2 rdf:resource="#proprietes"/>
    <tod:est_compose_de2 rdf:resource="#les_associations"/>
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >modelisation_conceptuelle</tod:nom_cpc>
    <tod:est_compose_de2 rdf:resource="#identifiant"/>
    <tod:est_compose_de2 rdf:resource="#schema_conceptuel"/>
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</tod:id_cpc>
  </tod:competence_composee>
  <tod:competence_elementaire rdf:ID="competence_elementaire_118"/>
  <tod:competence_composee rdf:ID="les_sgbd_relationnels">
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >10</tod:id_cpc>
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >les_sgbd_relationnels</tod:nom_cpc>
  </tod:competence_composee>
  <rdf:Description rdf:ID="situation_probleme_105">
    <tod:enonce_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Soit la base de donnÃ©es reprÃ©sentÃ©e par les relations
suivantes : Boutique (NomB, VilleB, QuartierB) Client (NomC, rueC,
```

```
villeC) EmployÃ© (NomE, NomB, salaireE) Achat (NomC, NomB, NomE,
montant, date, heure) 1.  DÃ©signer la clÃ© primaire, la(es)
clÃ©(s) secondaire (s) Ã©ventuelle (s) et la ou les clÃ©s
Ã©trangÃ¨res pour chaque relation ? 2.    Formuler les expressions
algÃ©briques pour rÃ©pondre aux requÃªtes suivantes : ï€  La liste
des boutiques du quartier Â« Champlain Â»  ï€  La liste des noms
des boutiques de la ville Sainte-Foy ï€  Nom des clients de la rue
RenÃ©-levesque et qui sont employÃ© dâ€™une boutique ï€  Afficher
les noms des boutiques qui ont vendu quelque chose ï€ La liste des
noms des clients qui nâ€™ont </tod:enonce_sp>
  </rdf:Description>
  <tod:competence_composee
rdf:ID="modelisation_logique_relationnelle">
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="schema_de_une_relation">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >schema_d'une_relation</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >19</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="le_domaine">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >le_domaine</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >12</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="contrainte_de_cle">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >contrainte_de_cle</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >17</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >modelisation_logique_relationnelle</tod:nom_cpc>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="relation_ou_table">
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >15</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >relation_ou_table</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
```

```
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >4</tod:id_cpc>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="contrainte_Integrite">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >contrainte_d'integrite</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >18</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="attributs">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        ></tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >13</tod:id_cpe>
        <tod:evalue_par rdf:resource="#situation_probleme_1"/>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="contrainte_de_domaine">
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >16</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >contrainte_de_domaine</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="les_n_uplets">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >les_n_uplets</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >14</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
  </tod:competence_composee>
  <tod:competence_composee rdf:ID="la_normalisation_de_relations">
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >6</tod:id_cpc>
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >la_normalisation_de_relations</tod:nom_cpc>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="premiere_forme_normale">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >premiere_forme_normale</tod:nom_cpe>
        <tod:id_cpe
```

```
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >30</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="forme_normale_de_BetC">
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >34</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >forme_normale_de_BetC</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="troisieme_forme_normale">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >troisieme_forme_normale</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >33</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="deuxieme_forme_normale">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >deuxieme_forme_normale</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >31</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
  </tod:competence_composee>
  <tod:competence_composee rdf:ID="introduction_aux_bdd">
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >introduction_aux_bdd</tod:nom_cpc>
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</tod:id_cpc>
  </tod:competence_composee>
  <rdf:Description rdf:ID="competence_elementaire_88">
    <tod:id_cpe rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >24</tod:id_cpe>
  </rdf:Description>
  <rdf:Description rdf:ID="competence_elementaire_91">
    <tod:id_cpe rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >27</tod:id_cpe>
  </rdf:Description>
  <tod:situation_probleme rdf:ID="situation_probleme_3">
    <tod:enonce_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Soit R(A, B, C, D) avec F={Aï,®B, Bï,®C, Cï,®D}. Est-ce que
Aï,®D ? Trouver F+ la fermeture transitive ? </tod:enonce_sp>
    <tod:nom_sp
```

```
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >situation_probleme_3</tod:nom_sp>
    <tod:id_sp rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</tod:id_sp>
  </tod:situation_probleme>
  <tod:competence_disciplinaire rdf:ID="administration_des_bdd">
    <tod:prerequis>
      <tod:competence_disciplinaire rdf:ID="manipulation_de_bdd">
        <tod:nom_cpd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >manipulation_de_bdd</tod:nom_cpd>
        <tod:id_cpd
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >3</tod:id_cpd>
        <tod:prerequis>
          <tod:competence_disciplinaire rdf:ID="conception_de_bdd">
            <tod:id_cpd
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
            >2</tod:id_cpd>
            <tod:nom_cpd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >conception_de_bdd</tod:nom_cpd>
          </tod:competence_disciplinaire>
        </tod:prerequis>
      </tod:competence_disciplinaire>
    </tod:prerequis>
    <tod:prerequis rdf:resource="#conception_de_bdd"/>
    <tod:id_cpd rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</tod:id_cpd>
    <tod:nom_cpd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >administration_des_bdd</tod:nom_cpd>
  </tod:competence_disciplinaire>
  <tod:obstacle_discilplinaire rdf:ID="obstacle_discilplinaire_119">
    <tod:nom_obd
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >obstacle_discilplinaire_119</tod:nom_obd>
    <tod:id_ob rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >6</tod:id_ob>
  </tod:obstacle_discilplinaire>
  <tod:competence_composee
rdf:ID="traduction_de_schema_conceptuel_en_schema_relationnel">
    <tod:est_compose_de2>
      <tod:competence_elementaire
rdf:ID="traduction_de_la_relation_binaire_un_a_un">
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >0</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        ></tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire
rdf:ID="traduction_de_la_relation_binaire_un_a_plusieurs">
```

```
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >39</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>traduction_de_la_relation_binaire_un_a_plusieurs</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire
rdf:ID="traduction_de_la_relation_binaire_plusieurs_a_plusieurs">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>traduction_de_la_relation_binaire_plusieurs_a_plusieurs</tod:nom_cp
e>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >38</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire
rdf:ID="autres_attributs_de_la_table">
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >autres_attributs_de_la_table</tod:nom_cpe>
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >37</tod:id_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="cle_de_la_table">
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >36</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >cle_de_la_table</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:est_compose_de2>
      <tod:competence_elementaire rdf:ID="nom_de_la_table">
        <tod:id_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >35</tod:id_cpe>
        <tod:nom_cpe
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >nom_de_la_table</tod:nom_cpe>
      </tod:competence_elementaire>
    </tod:est_compose_de2>
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >7</tod:id_cpc>
    <tod:nom_cpc
```

```
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

>traduction_d'un_schÃ©ma_conceptuel_en_schema_relationnel</tod:nom_c
pc>
  </tod:competence_composee>
  <tod:situation_probleme rdf:ID="situation_probleme_4">
    <tod:nom_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></tod:nom_sp>
    <tod:id_sp rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</tod:id_sp>
    <tod:enonce_sp
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></tod:enonce_sp>
  </tod:situation_probleme>
  <tod:competence_composee rdf:ID="langage_relationnel_sql">
    <tod:id_cpc rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >9</tod:id_cpc>
    <tod:nom_cpc
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >angage_relationnel_sql</tod:nom_cpc>
  </tod:competence_composee>
  <rdf:Description rdf:ID="competence_elementaire_92">
    <tod:id_cpe rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >28</tod:id_cpe>
  </rdf:Description>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.4.4, Build 579)
http://protege.stanford.edu -->
```

Tableau 12: Le code OWL généré pour l'ontologie de domaine de competences.