

TP

Ingénierie des connaissances

Exemple de définition d'une
ontologie via Protégé

Objectif de la modélisation

- Structuration & analyse des données d'une collection bibliographique en vue de :
 1. Caractériser finement les auteurs et leurs œuvres en fonction de différents critères (type & thématique des œuvres, popularité, ...).
 2. Distinguer des recommandations d'œuvres et d'auteurs pour des publics divers.
 3. Distinguer des manques éventuels dans la collection bibliographique.

Les concepts centraux

- Œuvre
 - *a* un nom
 - *est écrite par* un Auteur
 - *est d'un* Type (livres, essais, romans, BD,...)
 - *traite de* Thèmes (Math, Info, Physique, Manga...)
 - *a une popularité*, e.g. nombre de ventes
- Auteur
 - *A une* nationalité, ..., un sexe, une date de naissance...
 - *écrit* des œuvres

Les relations importantes

(Domaine) *relation* (Co-domaine)

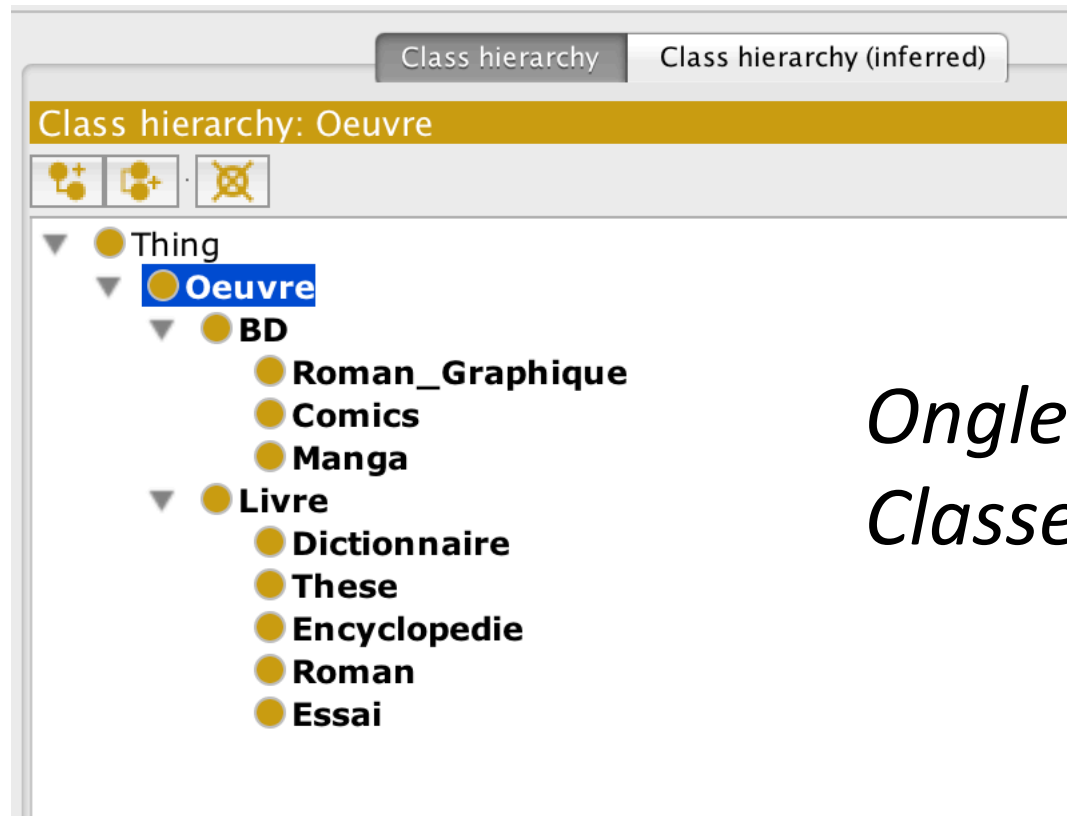
- Œuvre
 - (Œuvre) *aPourLabel* (Littéral)
 - (Œuvre) *aPourAuteur* (Auteur)
 - (Œuvre) *estUn* (Type Œuvre)
 - (Œuvre) *aPourTheme* (Thème)
 - (Œuvre) *nombreDeVente* (Integer)
- Auteur
 - (Auteur) *aPourNationalité* (Pays)
 - (Auteur) *estAuteurDe* (Œuvre)

Les relations importantes

- Deux types de relations
 - *Object properties* reliant deux instances
 - (Œuvre) *aPourAuteur* (Auteur)
 - (Œuvre) *estUn* (Type Œuvre)
 - (Œuvre) *aPourTheme* (Thème)
 - *Data Properties* reliant une instance est une valeur de type primitif (integer, string, etc.)
 - (Œuvre) *aPourLabel* (Littéral)
 - (Œuvre) *nombreDeVente* (Integer)

Taxonomie des types d'Oeuvres

- (Œuvre) *estUn* (Type Œuvre)
- Définition d'une taxonomie de types d'œuvres



*Onglet
Classes*

Les relations

- (Auteur) *estAuteurDe* (Œuvre)

Description: estAuteurDe

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

Ranges (intersection) +

topObjectProperty

aPourAuteur

Auteur

Oeuvre

*Onglet
Object Properties*

Les relations

- (Œuvre) *nombreDeVentes* (Integer)

Description: nombreDeVentes

Equivalent To +

SubProperty Of +

Domains (intersection) +

Oeuvre

Ranges +

int

Disjoint With +

*Onglet
Data Properties*

*Sélectionner
Built in Datatypes*

Exemple d'Instances

- Auteurs
 - Karl_Marx
 - Voltaire
 - Edgar_Morin
 - Luc_Random
- Œuvres
 - Das_Kapital
 - Candide
 - La_Methode
 - Les_primates

Exemple d'assertions (faits)

- Author(Karl_Marx)
- estAuteurDe(Karl_Marx, Das_Kapital)
- nombreDeVentes(Das_Kapital, 36500)

- Author(Edgar_Morin)
- estAuteurDe(Edgar_Morin, La_Methode)
- nombreDeVentes(La_Methode, 9800)

- Author(Voltaire)
- estAuteurDe(Voltaire, Candide)
- nombreDeVentes(Candide, 53000)

- Author(Luc_Random)
- estAuteurDe(Luc_Random, Les_Primates)
- nombreDeVentes(Luc_Random, 357)

Exemple d'assertions

- Author(Voltaire)
- estAuteurDe(Voltaire, Candide)
- nombreDeVentes(Candide, 53000)

The screenshot displays the Protégé ontology editor interface. The top navigation bar includes tabs for Active Ontology, Entities, Classes, Object Properties, Data Properties, Annotation Properties, Individuals, OWLViz, DL Query, OntoGraf, Ontology Differences, and SPARQL. The 'Individuals' tab is currently selected.

On the left, the 'Class hierarchy' panel shows a tree structure starting with 'Thing', followed by 'Auteur' (highlighted), 'Oeuvre', 'BD' (with sub-classes 'Comics', 'Manga', 'Roman_Graphique'), and 'Livre' (with sub-classes 'Conte', 'Dictionnaire', 'Encyclopedie', 'Essai', 'Roman', 'These').

The main area is divided into three panels:

- Members list: Voltaire:** A list of individuals belonging to the 'Voltaire' class, including 'Edgar_Morin', 'Karl_Marx', 'Luc_Random', and 'Voltaire' (which is highlighted).
- Annotations: Voltaire:** A panel for viewing and adding annotations for the 'Voltaire' individual.
- Description: Voltaire:** A panel showing the types of the 'Voltaire' individual, currently displaying 'Auteur'.
- Property assertions: Voltaire:** A panel for viewing and adding property assertions for the 'Voltaire' individual. It shows an object property assertion 'estAuteurDe' with the value 'Candide'.

Utiliser la connaissance de notre domaine pour construire des classes complexes

- La classe *Bestseller* :
 - Une œuvre avec un nombre de ventes > 30000

→ New concept, «Equivalent To», puis class expression editor :

`Oeuvre and nombreDeVentes some integer[≥ 30000]`

Utilisez la syntaxe OWL Manchester

Le classe *Bestseller* :

Description: Bestseller

Equivalent To +

● **Oeuvre** and **nombreDeVentes** some integer[≥ 30000]

SubClass Of +

● Oeuvre

General class axioms +

SubClass Of (Anonymous Ancestor)

Members +

◆ Candide

◆ Das_Kapital

Utiliser la connaissance de notre domaine pour
construire des classes complexes

TopAuthor

estAuteurDe some BestSeller

Description: TopAuteur

Equivalent To +

● **estAuteurDe some BestSeller**

SubClass Of +

● Auteur

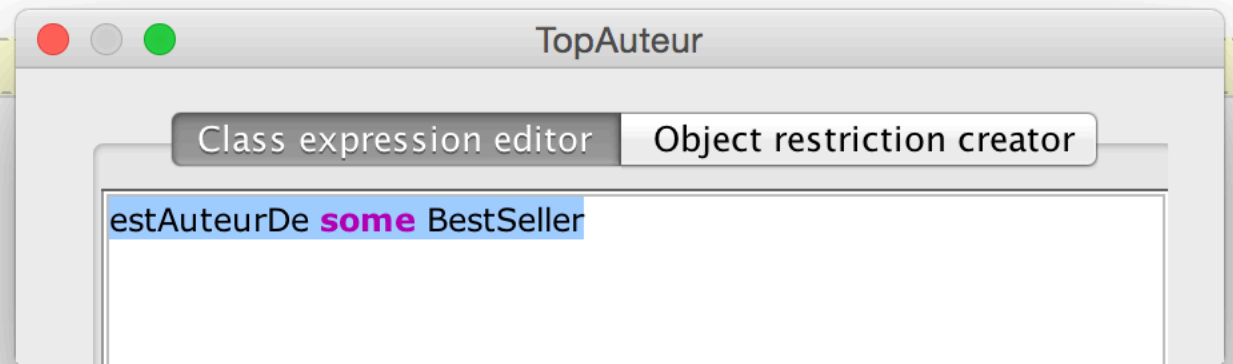
General class axioms +

SubClass Of (Anonymous Ancestor)

Members +

◆ Karl_Marx

◆ Voltaire



- Notez aussi la possibilité de définir des restrictions graphiquement.
- Pensez à utiliser le raisonneur pour vérifier la cohérence de votre modèle.
- Pensez aussi à sauvegarder différentes versions

Références

- La documentation de référence:
 - <http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/>
- Protégé
 - <http://protege.stanford.edu/>
 - <http://protegewiki.stanford.edu/wiki/Protege4UserDocs>
- OWL Manchester syntax
 - <http://www.w3.org/2007/OWL/wiki/ManchesterSyntax>
 - <http://www.semantic-web-book.org/w/images/7/75/W2011-08-OWL-Syntax.pdf>
 - <http://www.w3.org/TR/owl2-manchester-syntax/>