TP sur l'outil d'édition d'ontologies Protégé

1. Création des classes

Créer les classes et sous-classes de l'ontologie 'Famille' selon la figure suivante :

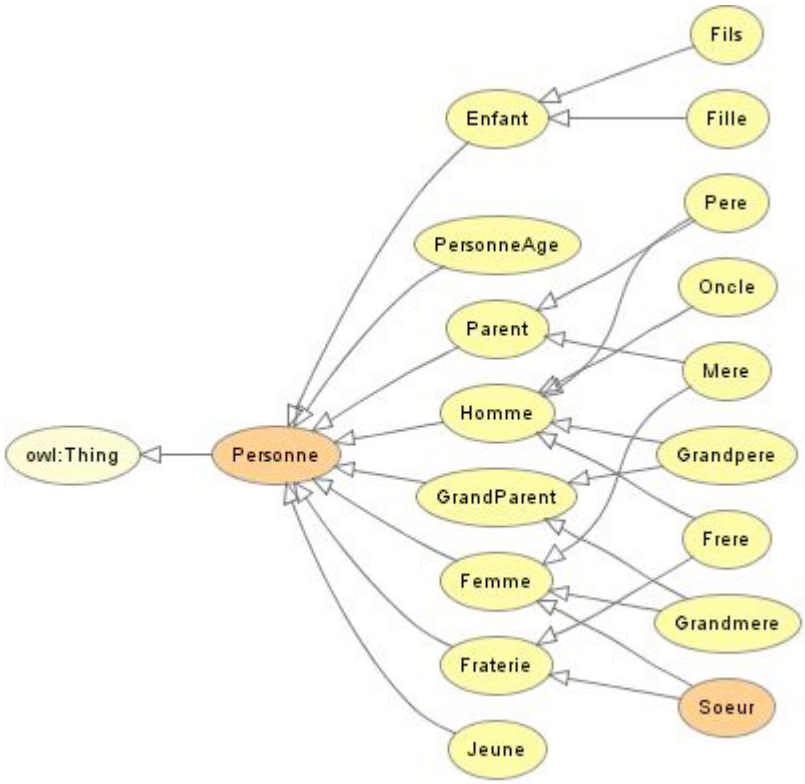

Figure 1 : L'ontologie Famille

Remarque : sur Protégé, pour qu'une classe soit sous classe de deux classes, il faut ajouter les classes 'mères' dans la restriction NECESSARY (avec le mode Logic View)

2. Création des propriétés pour les classes

1. Une personne possède un nom, un âge et une nationalité.
    a. Créer la propriété (type Datatype) *nom* avec le domaine **Personne** et le range xsd:String
    b. Créer la propriété (type Datatype) *age* avec le domaine **Personne** et le range xsd:int
    c. Créer la propriété (type Datatype) *nationalite* avec le domaine Personne et le range xsd:String
2. Deux personnes peuvent se marier
    a. Créer la propriété (type Object) *se_marier_avec* avec le domaine **Personne** et le range **Personne**

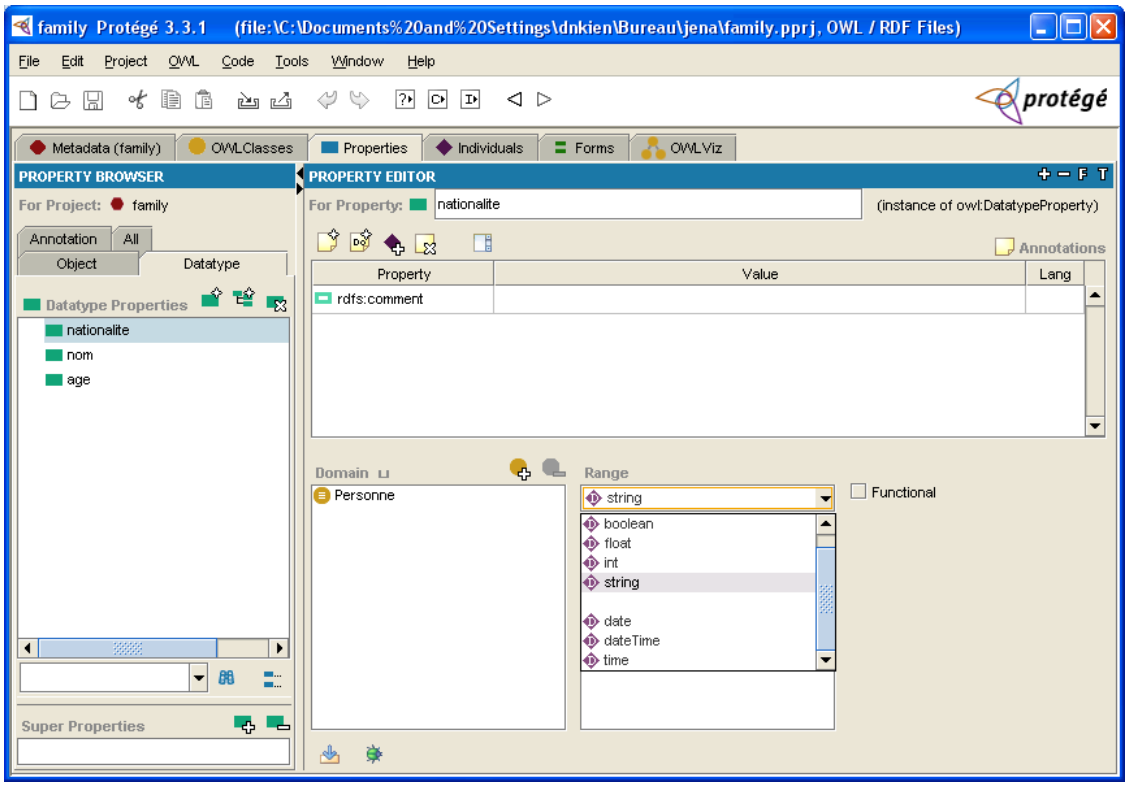

Figure 2 : La propriété DataType

3. Une personne est le parent d'une autre personne
    a. Créer la propriété (type Object) estParentDe avec le domaine Personne et le range Personne
4. Un homme est le père d'une personne
    a. Créer la propriété (type Object) *estPereDe* qui est la sous propriété de estParentDe avec le domaine **Homme** et le range **Personne**
5. Une femme est la mère d'une personne
    a. Créer la propriété (type Object) *estMereDe* qui est la sous propriété de estParentDe avec le domaine **Femme** et le range **Personne**
6. Une personne appartient à la fraterie d'une autre personne
    a. Créer la propriété (type Object) *estEnRelationDeFraterieAvec* avec le domaine **Personne** et le range **Personne**
7. Un homme est le frère d'une personne
    a. Créer la propriété (type Object) *estFrereDe* qui est la sous propriété *estEnRelationDeFraterieAvec* avec le domaine **Homme** et le range **Personne**
8. Une femme est la sœur d'une personne
    a. Créer la propriété (type Object) *estSoeurDe* qui est la sous propriété *estEnRelationDeFraterieAvec* avec le domaine **Femme** et le range **Personne**
9. Une personne est un enfant d'une autre personne

a. Créer la propriété (type Object) estEnfantDe avec le domaine **Personne** et le range **Personne**

10. Un homme est le fils d'une personne
    a. Créer la propriété (type Object) *estFilsDe* qui est la sous propriété *estEnfantDe* avec le domaine **Homme** et le range **Personne**
11. Une femme est la fille d'une personne
    a. Créer la propriété (type Object) *estFilleDe* qui est la sous propriété *estEnfantDe* avec le domaine **Femme** et le range **Personne**

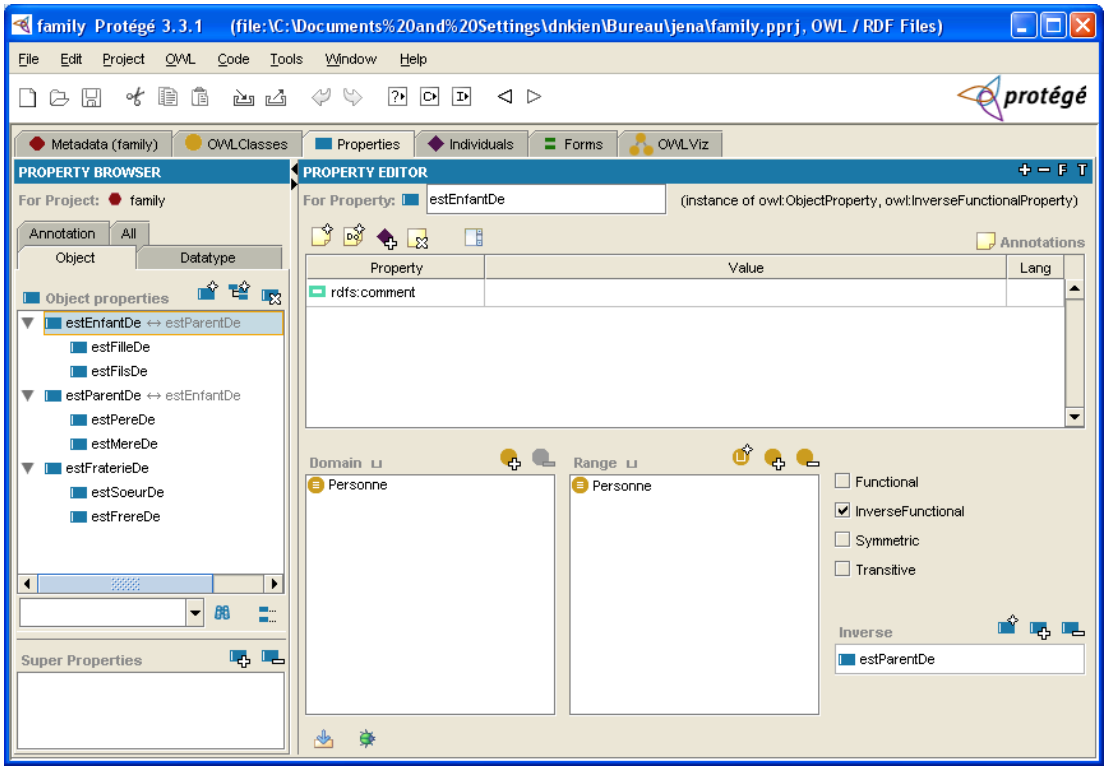

Figure 3 : La propriété Object

3. Création des restrictions sur les classes et propriétés

NECESSARY AND SUFFICIENT CONDITION :

- La classe Oncle a la restriction : *est frère d'un parent*
- La classe Grand-père a la restriction : *est père d'un parent*
- La classe Grand-mère a la restriction : *est mère d'un parent*
- La classe Pere a la restriction : La valeur de la propriété ***estPereDe*** a au moins une instance
- La classe Mere a la restriction : La valeur de la propriété ***estMereDe*** a au moins une instance
- La classe Fils a la restriction : La valeur de la propriété ***estFilsDe*** a au moins une instance
- La classe Fille a la restriction : La valeur de la propriété ***estFilleDe*** a au moins une instance
- La classe Frere a la restriction : La valeur de la propriété ***estFrereDe*** a au moins une instance
- La classe Sœur a la restriction : La valeur de la propriété ***estSoeurDe*** a au moins une instance

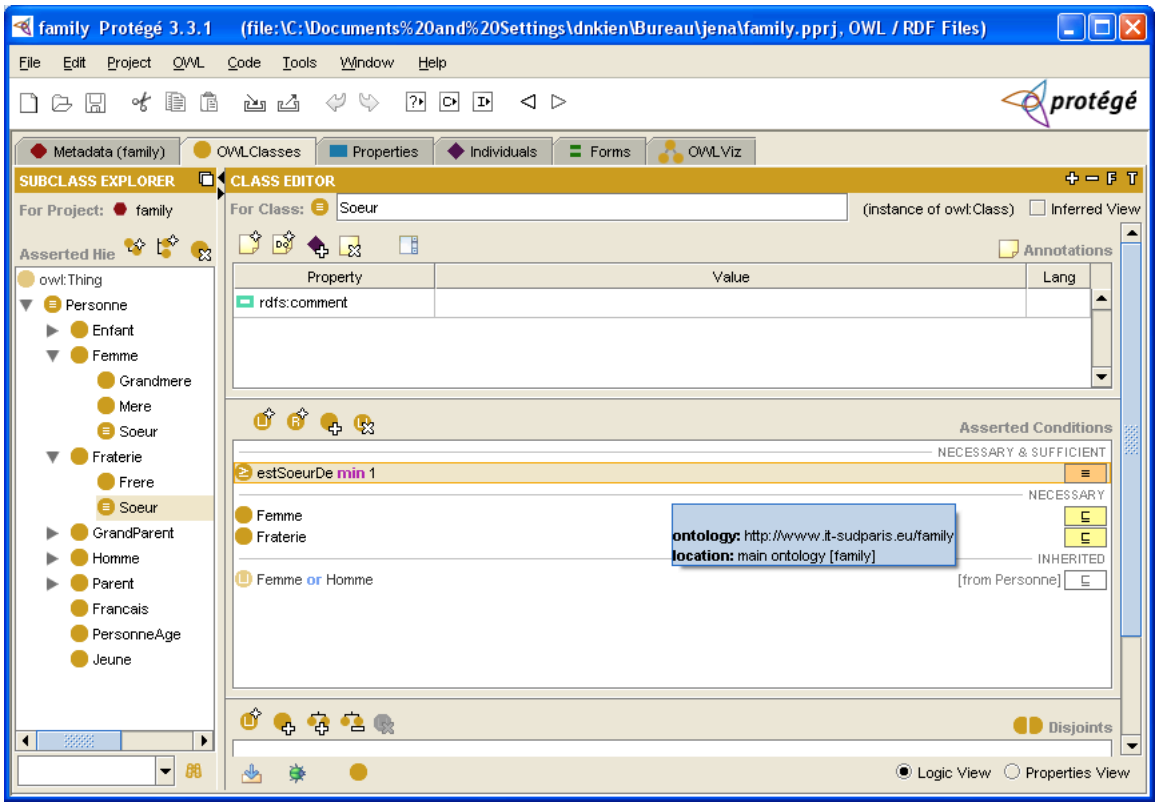

Figure 4 : La restriction sur la classe Soeur

CLASSES DISJOINTES :

- Homme et Femme sont disjointes
- Père et Mère sont disjointes
- Fils et Fille sont disjointes
- Grand père et Grand mère sont disjointes

4. Assigner les types pour les propriétés

1. La propriété *se_marier_avec* et *estEnRelationDeFraterieAvec* sont symmetriques
2. La propriété *estEnRelationDeFraterieAvec* est transitive
3. La propriété *estEnfantDe* est la propriété inverse de la propriété *estParentDe*
4. La propriété *nom, age, nationalite* sont functionnelle

5. Création des instances

1. Créer des instances pour la classe **Homme** :
    a. Peter, 70, se_marier_avec Marie, de nationalité française
    b. Thomas, 40, estFilsDe Peter, de nationalité française
    c. Paul, 38 estFilsDe Peter
    d. John, 45, de nationalité anglaise
    e. Jean, 10, estFilsDe John, anglaise
    f. Tom, 10, estFilsDe Thomas et Alex
    g. Micheal, 5, estFilsDe Thomas et Alex

2. Créer des instances pour la classe **Femme :**
    a. Marie, 69, de nationalité française
    b. Sylvie, 30, estFilleDe Marie et Peter,
    c. Chloé, 18, estFilleDe Marie et Peter,
    d. Sylvie se_marier_avec John
    e. Claude, 5, estFilleDe Sylvie, de nationalité française
    f. Alex, 25, se_marier_avec Thomas

6. Vérification d'ontologie à l'aide du moteur d'inférence PELLET

1. Configurer le moteur d'inférence : Dans le menu : **Reasonning/Pellet**
2. Vérifier la consistance d'ontologie : Dans le menu : **Reasonning /Check Consistency**
3. Inférer des instances de classes **Personne**, **Oncle, Frère, Sœur, Grand-Parent, Grand-père, Grand-mere ;** vérifier les instances générées automatiquement dans les classes Personne, Grand-Parent, Grand-père, Grand-mere. Aucune instance n'est générée dans les classes Frère, Sœur et Oncle. Pourquoi ? Comment inférer par exemple les relations frereDe ou soeurDe à partir de enfantDe ? Essayez d'exprimer ces relations avec des restrictions OWL.

Le TP suivant (Jena) va vous permettre d'exprimer ce genre de restrictions en langage de règles. Les règles d'inférences permettent en effet de pallier ce problème.