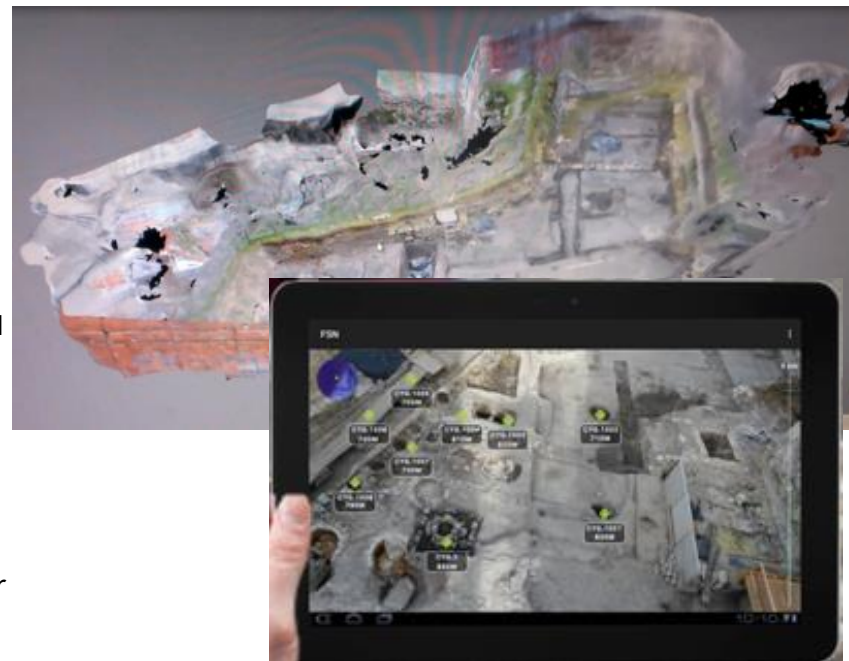


Projet Fiche Stratigraphique Numérique : des outils informatiques pour les archéologues



- La création, modification et visualisation des informations recueillies sur le site fouillé :
 - Unités stratigraphiques
 - Éléments recueillis
 - Faits archéologiques
- Un tableau de bord pour le suivi des fouilles basé sur la restitution des données
- La gestion du cycle de vie du matériel et sa conservation (éléments recueillis) avec une application de rangement sur smartphone
- Une application de recueil des données sur tablette synchronisée avec le site central
- Le contrôle de cohérence des données par analyse sémantique
- La modélisation 3D du terrain de fouilles et des éléments
- La géo localisation des constituants du site
- Un module de réalité augmentée
- Un module de gestion des documents (photos,...)



Site d'expérimentation : îlot Cygne

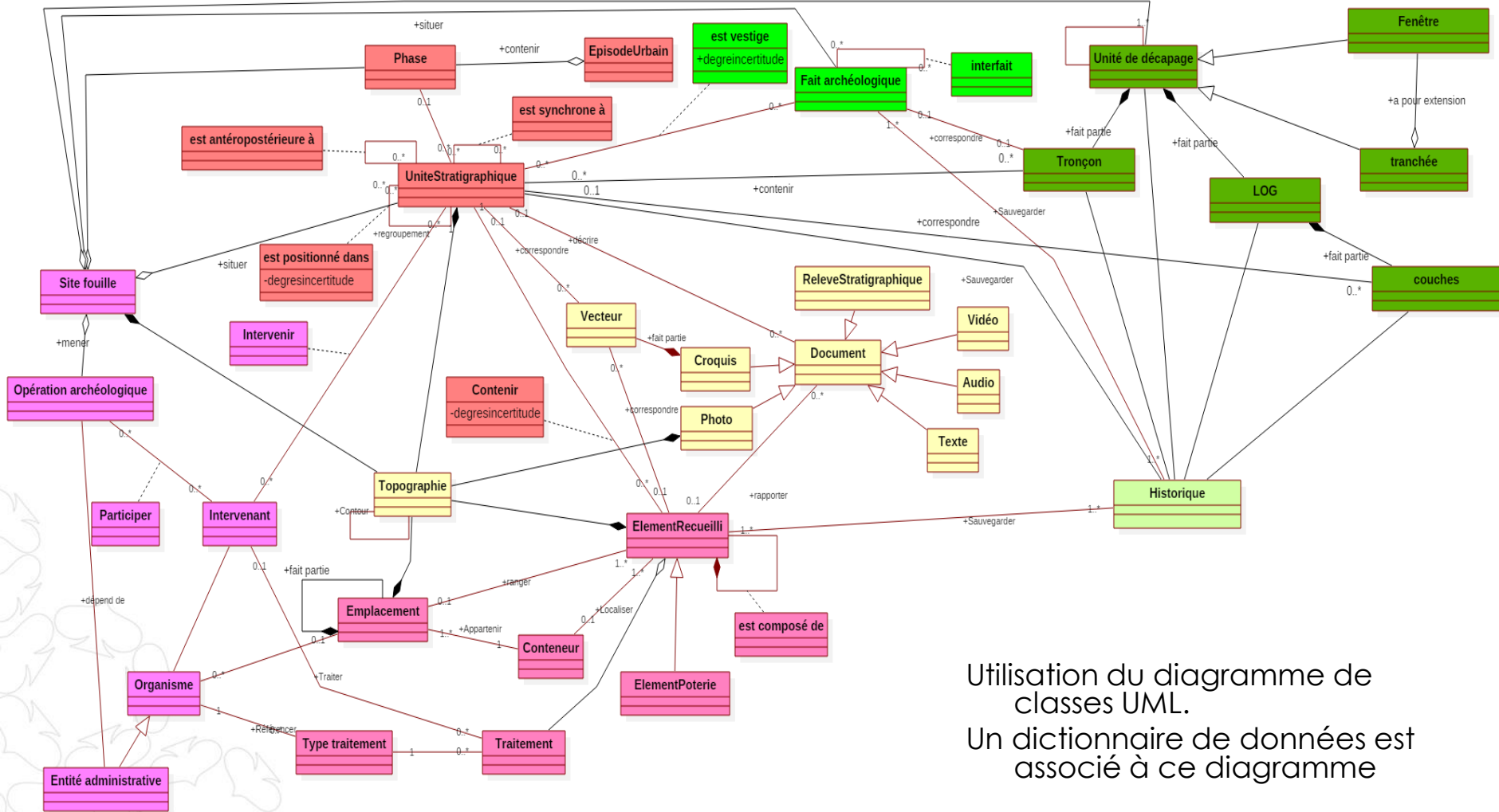
Projet de recherche et innovation
Sogeti réalisé en interne en partenariat
avec l'Unité d'archéologie de Saint-Denis

Démarrage du projet : décembre 2012

FSN : état du projet début 2017

- Mise en production de la version 3 de FSN depuis octobre 2016 pour l'Unité d'Archéologie de Saint-Denis.
- Ouverture de FSN pour d'autres unités d'archéologie : hébergement de l'application par le CNRS (Humanum)
- Lancement des travaux (version 4 et plus) sur :
 - L'amélioration du tableau de bord.
 - La réécriture et la connexion du Stratifiant.
 - La reconnaissance automatique de motifs sur des photos.
 - L'intégration d'outils statistiques : analyse multifactorielle des données.
 - L'extension de FSN aux opérations de diagnostic.
 - La gestion de nouvelles propriétés et de nouvelles relations sur les faits archéologiques.
 - La création d'un requêteur universel.

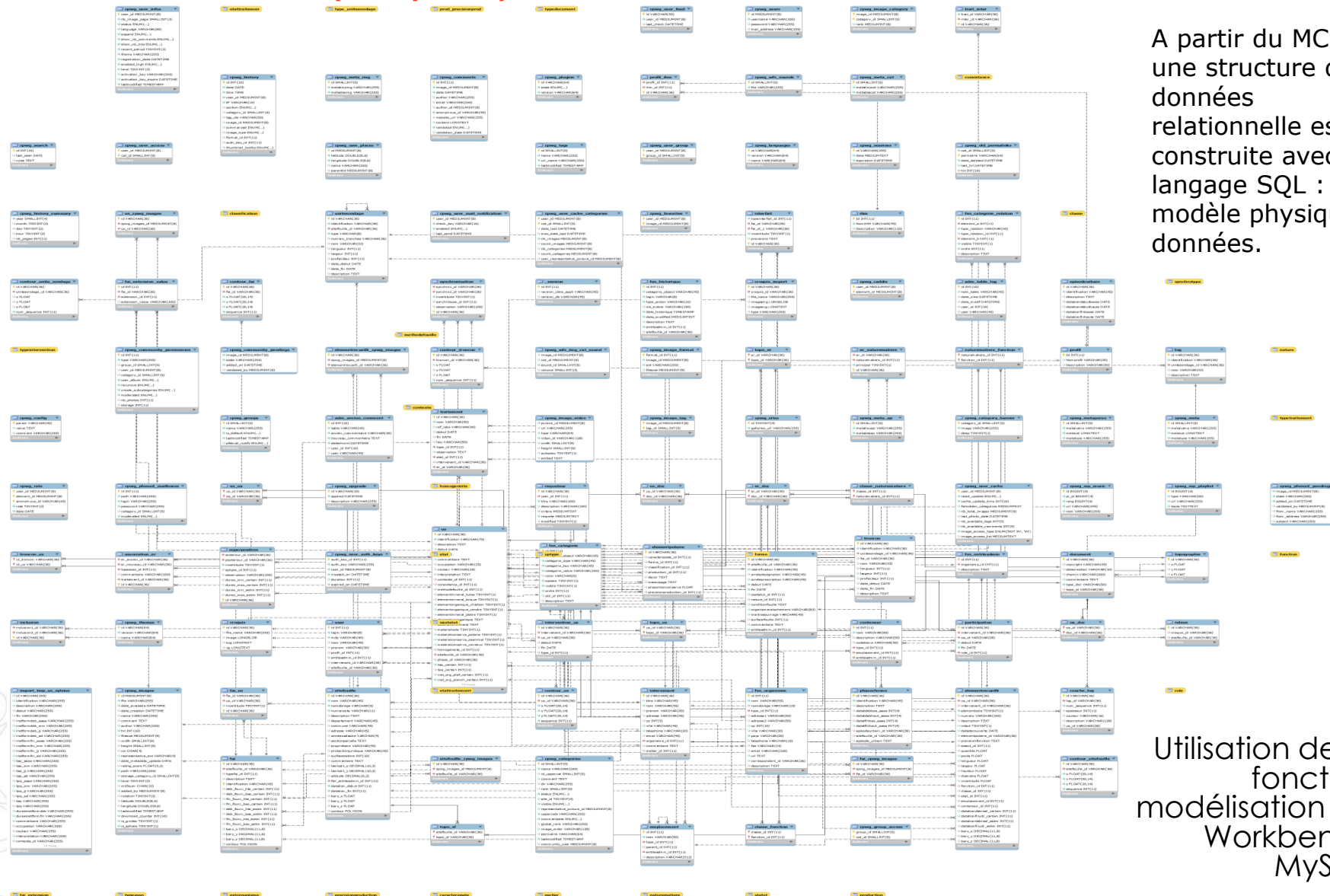
Le modèle conceptuel de données de FSN



Utilisation du diagramme de classes UML.
Un dictionnaire de données est associé à ce diagramme


Le modèle physique de données de FSN

A partir du MCD, une structure de données relationnelle est construite avec le langage SQL : modèle physique de données.



Utilisation de la fonction modélisation de Workbench MySQL


Un requêteur pour la consultation et l'échantillonnage des données



MENU PRINCIPAL

- Gestion
- Suivi de chantiers
- Statistiques
- Historique
- Outils
 - Import
 - Export
 - Export Stratifiant
 - Manuel du Site
 - Manuel de la Tablette (FSN Fouille)
 - Médiathèque
 - Requêteur

Tous les sites



Filtre Avancé Mode complet

Sélection des éléments "Élément Recueilli" de la requête

Choix élément			Valeur recherchée	Action
Élément Recueilli	Décor	contient	dec	✓ + ✖
Élément Recueilli	Poids	compris entre	10 et 100	✓ + ✖
Élément Recueilli	Identifiant US (Unité Stratigraphique)			✓ + ✖
Unité Stratigraphi	Couleur	contient	rouge	✓ + ✖

Lister les Élément Recueilli pour lesquels (Décor de Élément Recueilli associé contient 'dec') et (Poids de Élément Recueilli entre 10 et 100) et (Couleur de Unité Stratigraphique associé contient 'rouge')

```
SELECT a1.id
FROM elementrecueilli AS a1
INNER JOIN elementpoterie AS p1 ON a1.elementpoterie_id = p1.id JOIN us AS a2 ON a2.id = a1.us_id
WHERE 1 AND (p1.decor LIKE '%dec%') AND (a1.poids BETWEEN 10 AND 100) AND (a2.couleur LIKE '%rouge%')
```

Titre de la requête: Titre de la requête

Description de la requête

Enregistrer | Editer | Lancer | Annuler | Retour

Création de faits archéologiques avec des propriétés et relations inter-faits étendues en fonction du type

Détail Fait Archéologique : cercueil1

Identification

Site de Fouille	MGA01	
Identification	cercueil1	
Description		
Type	cercueil	
Datations	Estimées	Certaines
Année Fin Fourchette Haute		
Année Fin Fourchette Basse		
Année Début Fourchette Haute		
Année Début Fourchette Basse		

Extension

Sexe occupant	Homme
Taille cercueil	202
Forme cercueil	rectangulaire
intact	<input checked="" type="checkbox"/>

Relations

Multimédia

Le modèle doit être robuste dans le temps : il doit supporter les ajouts, améliorations et modifications de l'application effectués au fil des années.

La robustesse du modèle se juge sur :

1. La capacité de conserver de bonnes performances en augmentant la quantité de données,
2. la capacité d'intégrer de nouvelles fonctions sur les objets sans modifier la structure : nouvelles consultations, nouveaux calculs,...
3. La capacité d'introduire de nouveaux objets et de nouvelles relations sans modifier la structure précédente et donc la base de données existantes.
4. La capacité de modifier les objets et les relations existants en continuant à gérer les anciennes données (obtenues avec l'ancienne structure).

Avec une base de données relationnelle, ces différentes capacités peuvent être en opposition : par exemple, vouloir conserver de bonnes performances en consultation et ajouter de nombreuses associations au modèle pour affiner les restitutions.

Les données doivent être rigoureusement structurées.

Le schéma de données est non flexible.

Toute modification de structure (ajout d'une nouvelle relation par exemple) entraîne un temps de développement et donc un coût avec un risque de régression de l'application.

Les limites du modèle relationnel (2)

La gestion des bases de données relationnelles (création, consultation, modification, suppression d'éléments) se fait avec un langage universel fondé sur la logique des prédicats du premier ordre (théorie des ensembles) : le **SQL (Structured Query Language)**.

En SQL, la description d'une relation se traduit par l'ajout d'un «JOIN». Or, avec des données connectées, on obtient rapidement des requêtes avec beaucoup de JOIN, ce qui les rend complexes et donc difficilement maintenables.

Exemple : *retrouver l'emplacement actuel des éléments de classe végétaux ayant subi un traitement de type datationC14 et ayant été recueillis sur une unité stratigraphique de contexte « remblai d'occupation » vestige d'un fait archéologique de type « puits » présent sur un site situé en Picardie.*

Plus on a de données dans une table et moins bonnes sont les performances. Pour calculer le résultat, les moteurs SQL font (quasiment) le produit cartésien de chacune des tables.

Mauvaises performances pour des données très connectées
Code complexe et difficilement maintenable

Les bases de données NOSQL

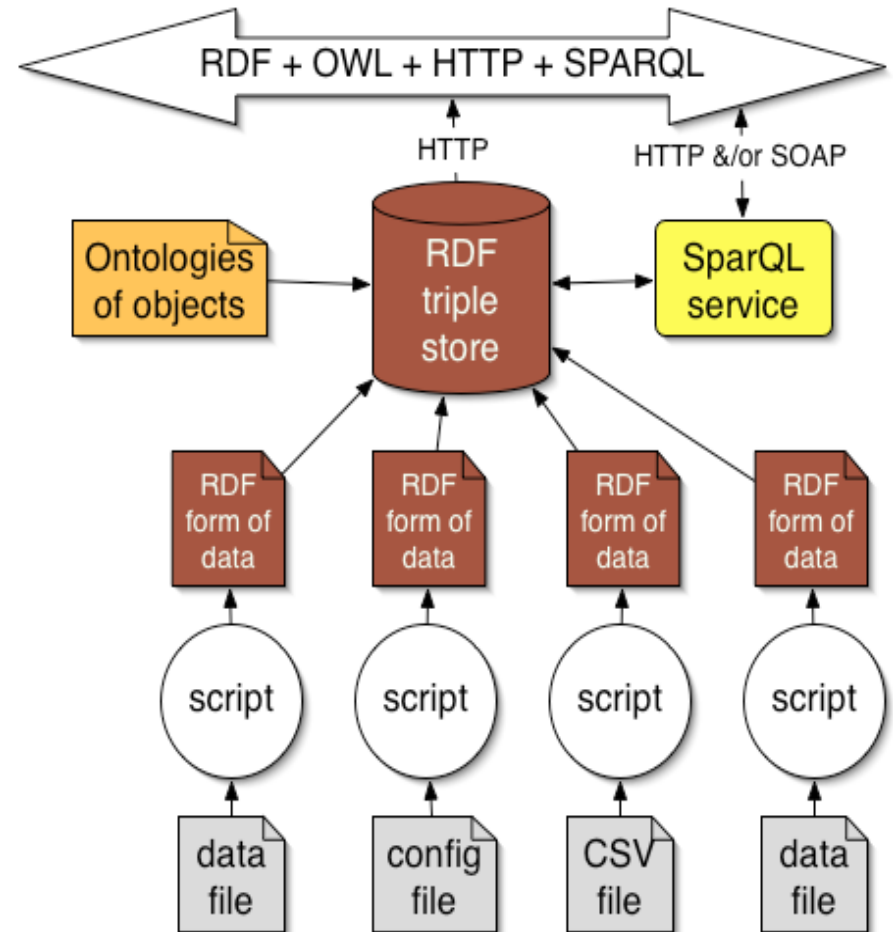
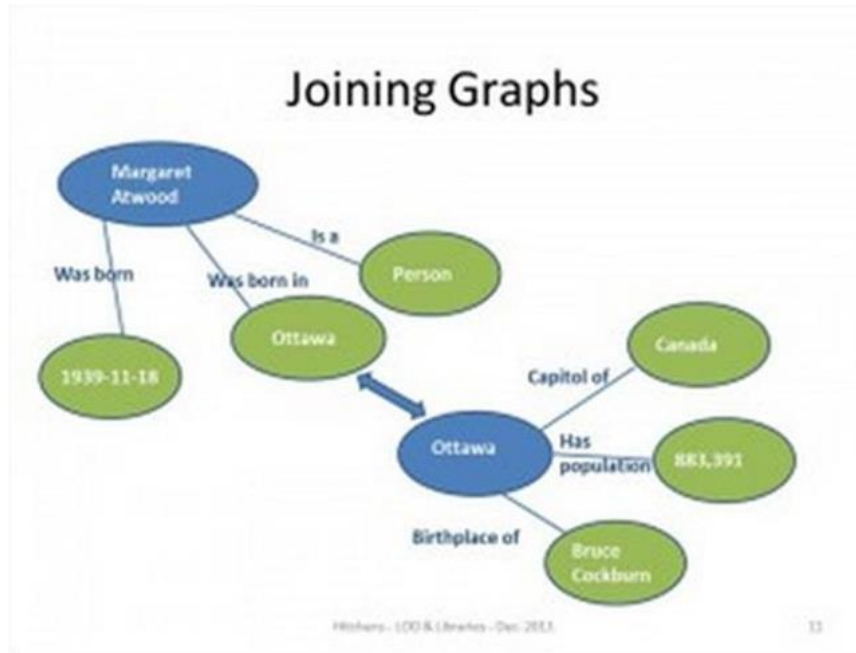
Trois familles de bases NoSQL :

0x235C	{name:Philip, UID: PPR, Groups: [CHI,SFO,BOS]}					Document DB
0xCD21	{name:Neo4j Chicago, UID: PPR, Members:[PPR,RB,NL], where:{city:Chicago, State: IL}}					MongoDB CouchDB
	Name	UID	Members	Groups	Photo	Column Family
0x235C	Philip	PPR		CHI, SFO, BOS	B75DD108A893A	HBase Cassandra
0xCD21	Neo4j Chicago	CHI	PPR,RB,NL		218758D88E901	
0x235C	Philip					Kev-Value
0xCD21	Neo4j Chicago					
0x2014	[PPR,RB,NL]					
0x3821	[CHI, SFO, BOS]					
0x3890	B75DD108A					
		membase		Riak	Redis	

Aucune de ces familles ne convient pour la gestion de données avec de nombreuses relations puisqu'il n'existe aucun moyen de modéliser, stocker ou requêter les relations. La seule manière de lier des objets est de créer des relations implicites et de réaliser les «jointures» dans le code des applications.

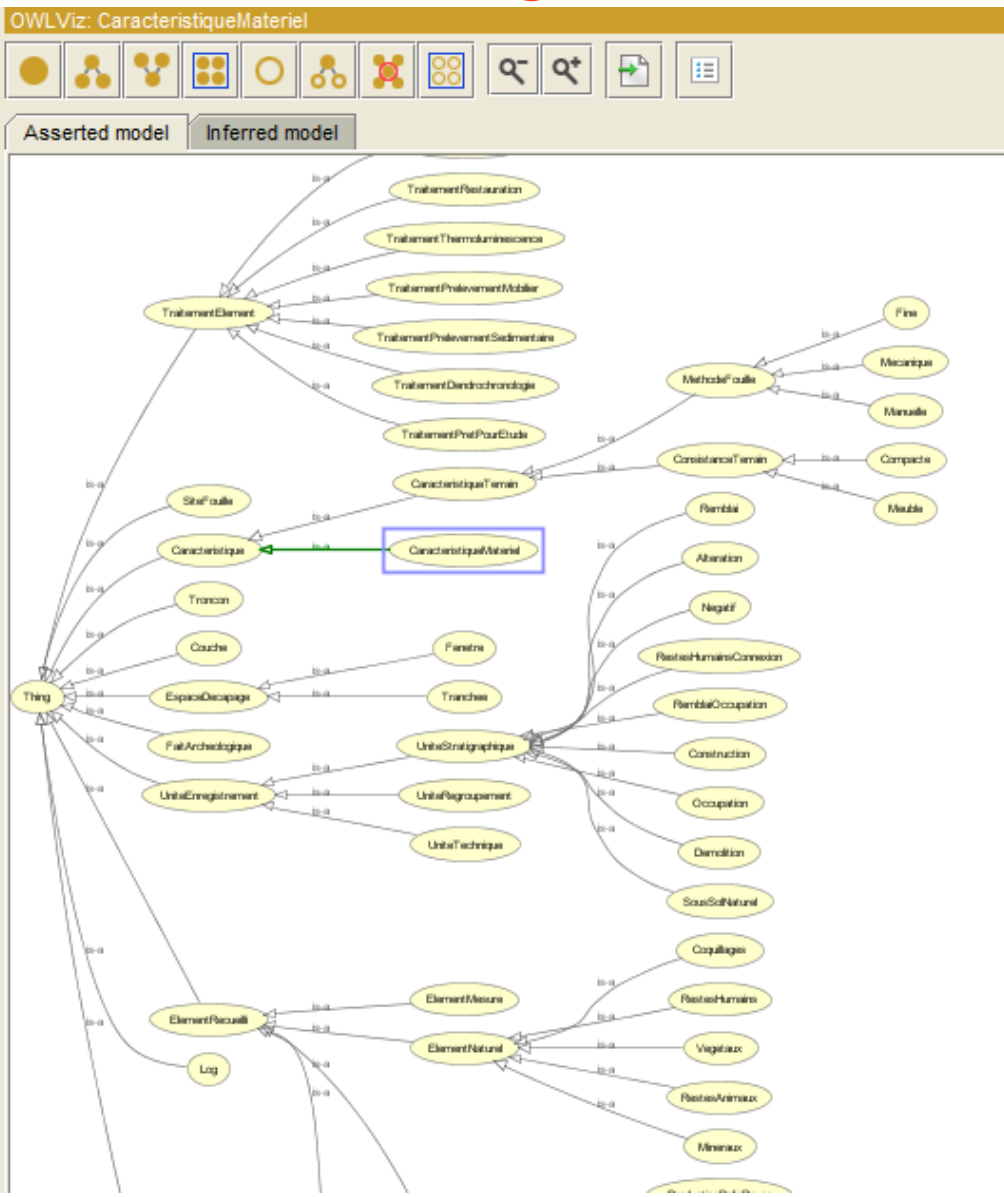
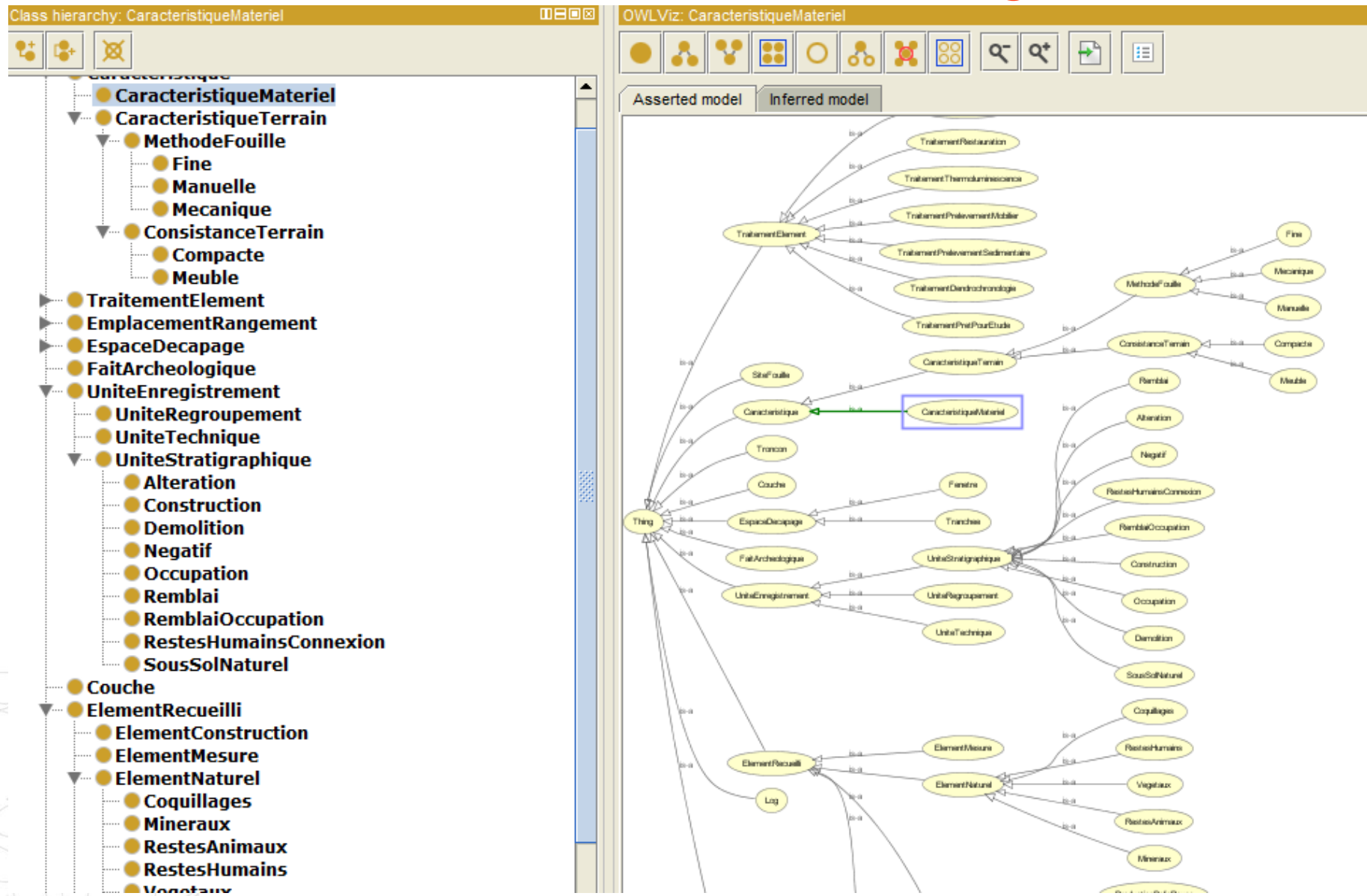
Les bases de données triplestore

Stockage et récupération de données sous un seul format : le triplet RDF (Resource Description Framework) : *(sujet, prédicat, objet)*

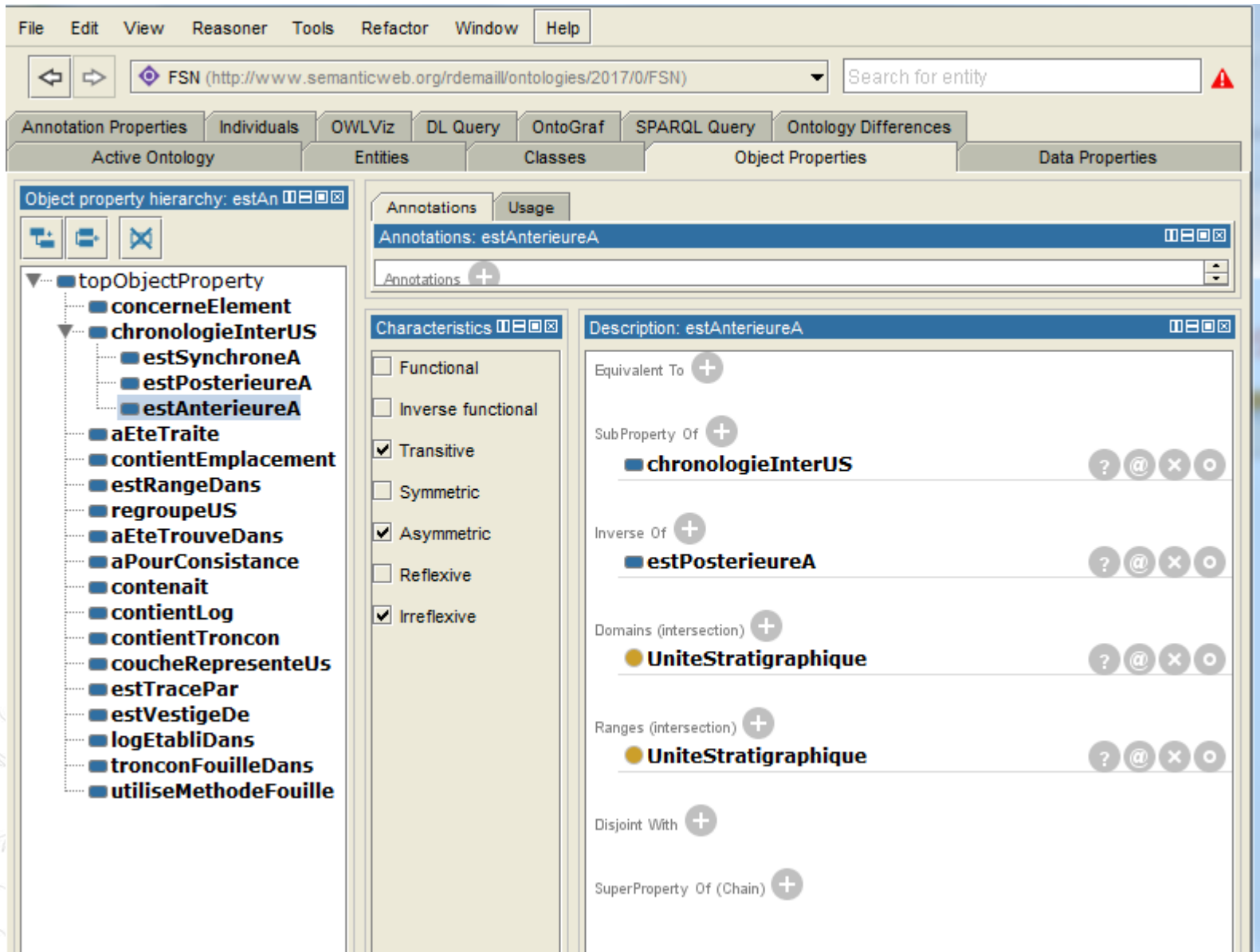


Un langage de requêtes pour les données du web : SPARQL
Prise en compte des ontologies OWL

Hiérarchie des classes OWL avec Protégé.



ObjectProperties OWL.



The screenshot shows the Protege OWL editor interface. The main window displays the configuration for the object property `estAnterieureA`. The left pane shows the object property hierarchy, with `estAnterieureA` selected under `topObjectProperty`. The right pane shows the property's characteristics and description.

Object property hierarchy: estAn

- topObjectProperty
 - concerneElement
 - chronologieInterUS
 - estSynchroneA
 - estPosterieureA
 - estAnterieureA**
 - aEteTraite
 - contientEmplacement
 - estRangeDans
 - regroupeUS
 - aEteTrouveDans
 - aPourConsistance
 - contenait
 - contientLog
 - contientTroncon
 - coucheRepresenteUs
 - estTracePar
 - estVestigeDe
 - logEtabliDans
 - tronconFouilleDans
 - utiliseMethodeFouille

Annotations: estAnterieureA

Characteristics: estAnterieureA

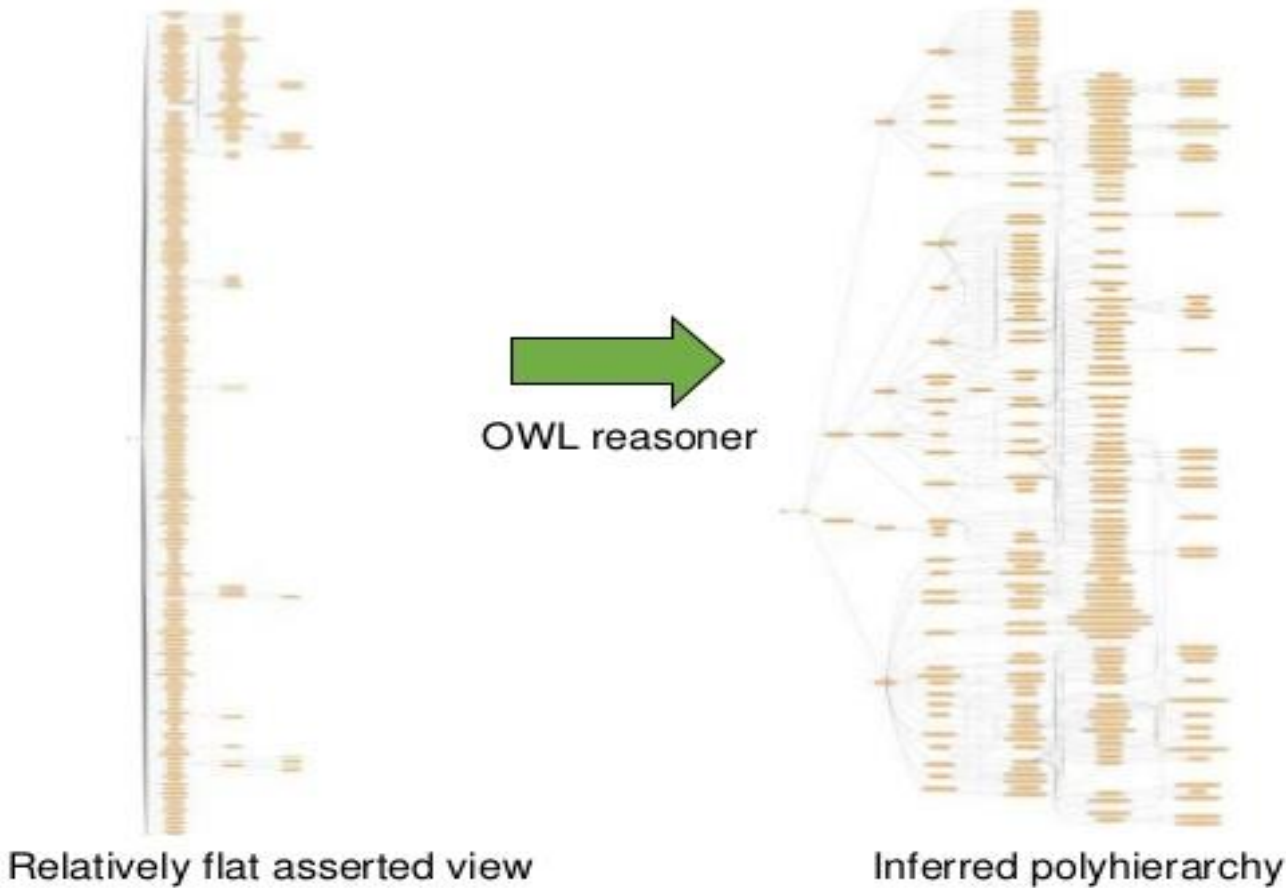
- Functional
- Inverse functional
- Transitive
- Symmetric
- Asymmetric
- Reflexive
- Irreflexive

Description: estAnterieureA

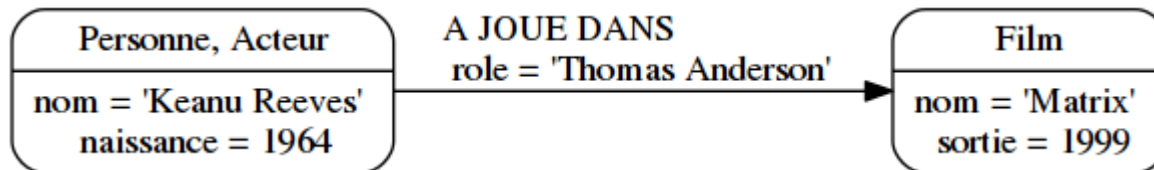
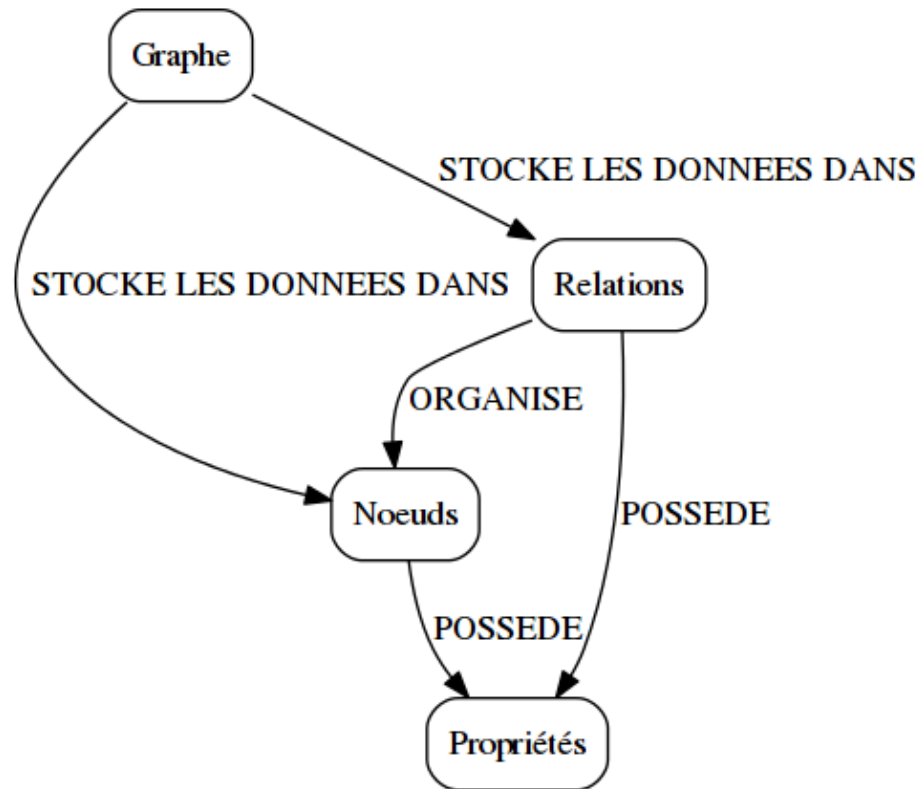
- Equivalent To +
- SubProperty Of +
 - chronologieInterUS** ? @ X O
- Inverse Of +
 - estPosterieureA** ? @ X O
- Domains (intersection) +
 - UniteStratigraphique** ? @ X O
- Ranges (intersection) +
 - UniteStratigraphique** ? @ X O
- Disjoint With +
- SuperProperty Of (Chain) +

Le raisonneur pour déduire la classification

Using a DL reasoner to infer classification



Les bases de données orientées graphe



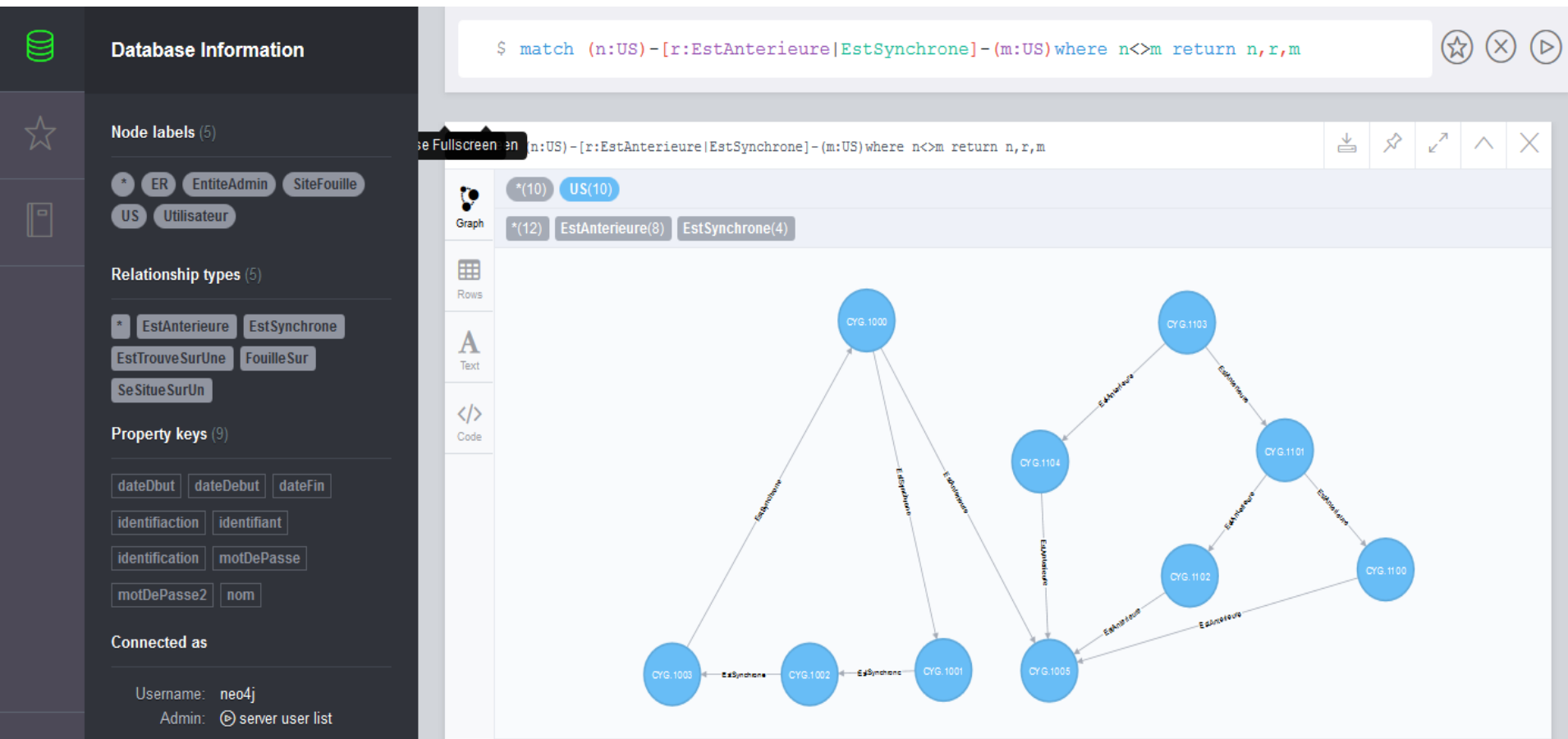
Les requêtes dans les bases de données orientées graphe

Cypher (base **Neo4j**): langage déclaratif permettant de requêter et mettre à jour le graphe.

Quelques exemples:

- `()` : n'importe quel nœud
- `(:Personne)` : un nœud avec le label `Personne`
- `(n:Personne)` : un nœud identifié dans la variable `n` avec le label `Personne`
- `(n:Personne:Acteur)` : un nœud identifié dans la variable `n` avec le label `Personne` et `Acteur`
- `(a)--(b)` : n'importe quelle relation entre le nœud `a` et `b` (peu importe la direction)
- `(a)-[:AMI]->(b)` : relation de type `AMI` depuis le nœud `a` vers le nœud `b`
- `(a)-[r:AMI|CONNAIT]->(b)` : relation identifiée dans la variable `r` de type `AMI` ou `CONNAIT` depuis le nœud `a` vers le nœud `b`

Utilisation de cypher (Neo4j) pour une consultation graphique de la base de données.



The screenshot displays the Neo4j web interface. On the left, the 'Database Information' sidebar shows node labels (ER, EntiteAdmin, SiteFouille, US, Utilisateur) and relationship types (EstAnterieure, EstSynchrone, EstTrouve SurUne, Fouille Sur, Se Situe SurUn). The main area features a Cypher query editor with the following query:

```
$ match (n:US)-[r:EstAnterieure|EstSynchrone]-(m:US)where n<>m return n,r,m
```

Below the query, the graph visualization shows 10 nodes (CYG.1000 to CYG.1100) and their relationships. The nodes are arranged in a hierarchical structure. Relationships are labeled with 'EstAnterieure' and 'EstSynchrone'. The graph shows a complex network of dependencies between the nodes.

Détection des incohérences chronologiques inter-US dans un diagramme de Harris avec cypher.

```
// creation du node bag et de ses inclusions
MATCH (u:US)-[:EstSynchrone*]-(us:US)
WITH u, collect(us) + u AS bags
UNWIND bags as bag_list
WITH u, max(id(bag_list)) as max_bags
WITH DISTINCT(max_bags) as mus_bag
MATCH (mus:US)-[:EstSynchrone*]-(muss:US)
WHERE id(mus) = mus_bag
WITH mus, muss
MERGE (mus)<-[:BAG_INCLUSION]-(bu:BAG_US)-[:BAG_INCLUSION]->(muss)
return bu,mus,muss

// creation des bag pour les us non-synchrhones
MATCH (u:US)
WHERE NOT (u)-[:EstSynchrone]-()
MERGE (u)<-[:BAG_INCLUSION]-(b:BAG_US)
return u,b

// creation des liens d'antéro-posteriorité entre les bags
MATCH (b:BAG_US)-->(us)<-[:EstAnterieure]-(usa)<-[:BAG_INCLUSION]-(ba)
MERGE (ba)-[:EstAnterieure]->(b)
return b,ba

//affichage des bags, des us et leurs relations
MATCH (b:BAG_US), (u:US) RETURN b,u

// detection d'un circuit => incohérence
MATCH (b:BAG_US)<-[:EstAnterieure*]-(b)
return b
```

Intégrer une ontologie dans une base orientée graphe.

Un langage de requêtes adapté à

- La prise en compte de la hiérarchie de classes : relation « isA »
- La prise en compte de l'héritage des relations

