WEB Sémantique

Rémi Gilleron
Inria Lille - Nord Europe & LIFL & Univ Lille 3

Cours donné en master 2 "informatique et document"
Cours accompagné de nombreux TPs
Cours inspiré de "Programming the Semantic Web" de Segaran, Evans et Taylor

septembre 2013

Les prémisses du Web

Les origines

- réflexions sur les documents électroniques
- architectures et protocoles réseaux

WEB $\equiv$ couche applicative sur le réseau Internet

Dates clés

- 90–92 : HTTP, HTML 1.0, NEXTSTEP
- 92–97 : définition des URL, naissance du W3C, MOSAIC puis IE, HTML intègre images, tableaux, applets.

La maturité du Web

Séparer forme et contenu

La nécessité de traiter et échanger les données du WEB font apparaître le besoin de séparer les données des traitements et de la présentation dans le navigateur.

WEB $\equiv$ grande base de données de documents structurés

Dates clés

- 96 : première définition des feuilles de style,
- 98 : définition de XML 1.0,
- 98–05 : langages de schémas DTD, XML SCHEMA, RELAX NG, langages XHTML, SVG, MATHML, ..., langages de manipulation XPATH, XLINK, XPOINTER, XSL, XSLT, XQUERY.

La situation actuelle du Web

Buzzword WEB 2.0

généralisation du modèle à tous les systèmes d'information, clients légers (le navigateur), syndication et push, XHTML 5.0, petits objets portables communicants, réseaux sociaux

WEB $\equiv$

- ensemble de ressources de très grande taille (WEB et WEB caché),
- outillé pour le traitement et la représentation de données et documents,
- outillé pour l'adressage, la publication et l'échange de données,
- outillé pour la recherche par l'humain de ressources,
- permettant l'interaction des utilisateurs dans des réseaux.

Le futur du web

WEB $\equiv$? ? ?

Les évolutions sont très difficiles à prévoir car fortement dépendantes des réactions des communautés à des propositions d'évolutions technologiques ou applicatives.

Récemment les petits objets portables communicants, les données multimedia, les capteurs, les réseaux sociaux

Demain la parole ? Nouvelles interfaces ? Nouveaux modèles de communautés ?

Un besoin : des utilisateurs désireux de services de plus en plus sophistiqués. Alors pourquoi pas le WEB 3.0 $\equiv$ Semantic WEB ?

Plan

RI sur le WEB

La situation

- Le mode d'interaction de l'utilisateur avec le WEB passe prioritairement par un moteur de RI,
- L'interrogation et la recherche sont faites de façon syntaxique
- L'utilisateur humain interprète les résultats, i.e. leur attribue une sémantique, et reformule sa requête au besoin

Exemple de requête

- "Hugo"
- liste ordonnée de documents du Web contenant la chaîne de caractères Hugo sur des critères syntaxiques
- L'utilisateur affine sa requête selon qu'il cherche un titre de roman de Victor Hugo, la date de naissance de Victor Hugo, ou encore un calecon Hugo Boss.

Services sur le WEB

La situation

La situation est identique car l'utilisateur humain doit lui-même décomposer sa demande en des recherches de services et appels de services WEB .

Exemple de service

- "3 jours à Rome le week end de Toussaint"
- géolocaliser le point de départ, connaître les moyens de transport et d'hébergement, appeler les services correspondants de transport et d'hébergement, comparer les offres, ...

Que faut-il ajouter au WEB actuel ?

- Il faudrait que les programmes (les services) puissent interpréter les données : ce document correspond à un hôtel, un hôtel est un mode d'hébergement, dans un hôtel on peut réserver des chambres, une chambre ... Il serait alors possible d'appeler les services, de les combiner, d'ordonner les réponses à la demande de service, ...
- Un prérequis est de représenter les connaissances liées aux données pour faire des inférences : cette page représente un hôtel, un hôtel est un mode d'hébergement donc cette page représente un mode d'hébergement.

Le WEB sémantique

Les challenges

Représenter les connaissances dans un monde ouvert (tout type de connaissance), à l'échelle du WEB et avec une très large variété de protocoles, de langues, de culture, ... pour pouvoir manipuler les connaissances liées au contenu du WEB.

Dates clés

- 94 : idées émises par Tim Berners Lee,
- 98 : formalisation des idées au W3C,
- 98– : langage de description RDF99, langage de schéma RDFS04, et de raisonnement OWL, langage de requête SPARQL, ...

La pile du WEB sémantique

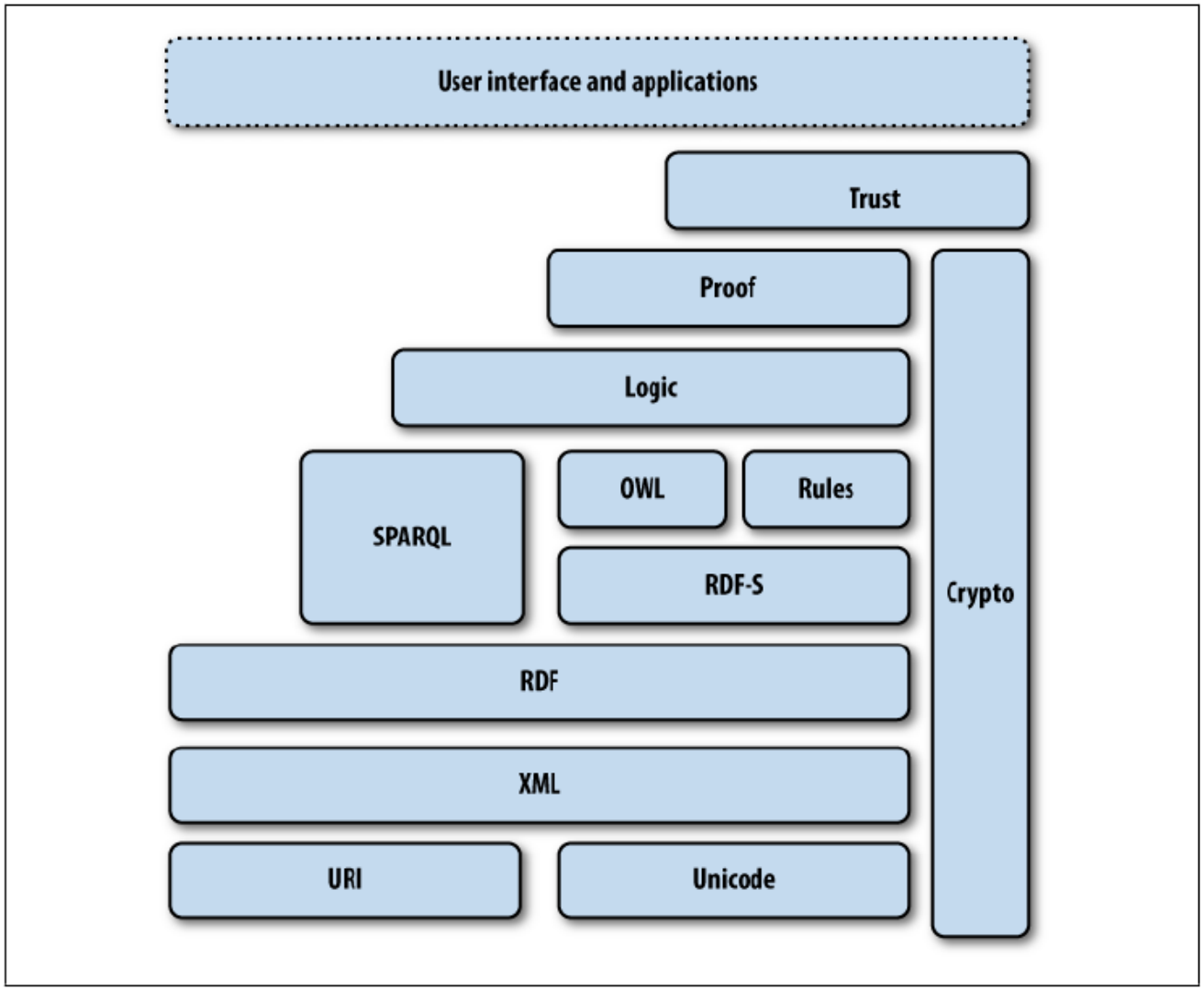


Figure 11-1. W3C semantic web technology stack

La pile réelle du WEB sémantique

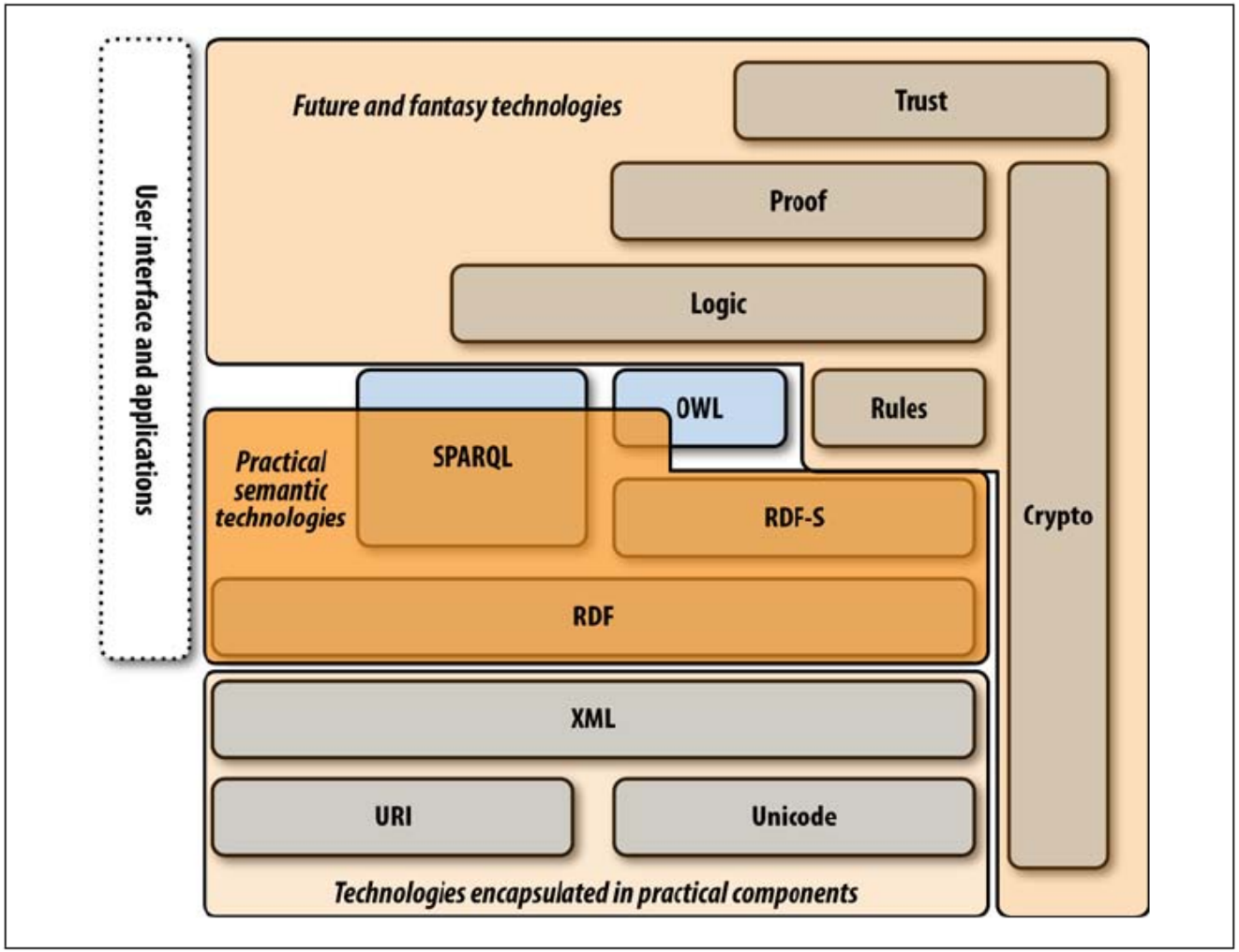


Figure 11-2. A practical view of the semantic stack

Le WEB sémantique

Le cours

Beaucoup de cours sont orientés “représentation des connaissances “ et “intelligence artificielle”. Ce cours aura plutôt une approche pragmatique partant des données et de l'existant. Le cours est basé sur Programming the Semantic Web de Segaran, Evans et Taylor.

Les questions

- Des exemples d'utilisation ?
- Idées dès 96, va-t-il enfin percer ?
- Les entreprises vont-elles publier leurs données ?
- Éthique et vie privée ?

Plan

Le modèle de données du WEB sémantique

Modèles de données usuels

Nous allons revoir les modèles de données les plus classiques : tabulaire, relationnel et arborescent et montrer leurs limites dans le contexte du WEB sémantique sur les plans représentation des connaissances et évolutivité.

Un modèle évolutif de données et connaissances

Nous allons introduire le modèle de données graphes, encore appelé modèle de triplets.

Modèle de données tabulaires

Exemple

	A	B	C	D	E
16		nom	sexe	age	ville
17		Dupont	1	35	Lille
18		Gilleron	1	38	Lomme
19		Lemoine		30	Leers

Critiques du modèle

- simplicité qui en fait un modèle utilisable par tout utilisateur,
- des outils de recherche (filtres) et de mise à jour,
- des tableurs avec outils de calcul et d'analyse,
- limité en taille, difficulté à gérer plusieurs tables, mono-utilisateur,
- sémantique externe et schéma peu évolutif

Le modèle relationnel

Dans un contexte où les données sont en volume important, où les données évoluent, où les utilisateurs sont nombreux avec des accès concurrents, il faut s'orienter vers des bases de données. Le modèle le plus utilisé est le modèle relationnel

- basé sur des tables vérifiant des propriétés de formes normales implanté dans
- des SGBDR (systèmes de gestion de bases de données relationnelles) tels que Oracle, PosGres, SQLite, ..., et utilisant
- un langage d'interrogation standardisé SQL et
- des transactions vérifiant les propriétés ACID : Atomicité, Cohérence, Isolation et Durabilité.

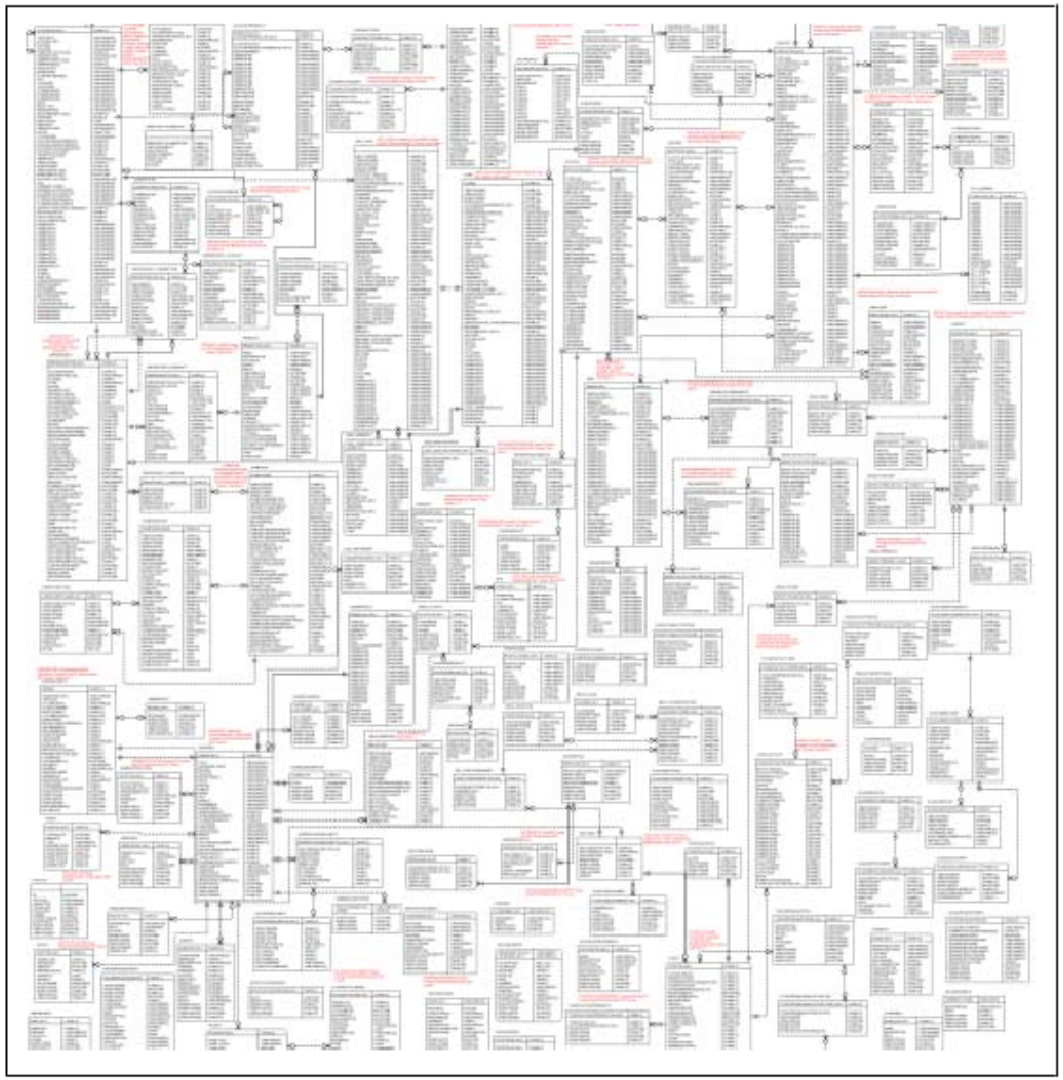
BD relationnelle

- Une base de données relationnelle est une base de données qui a été définie relativement au modèle relationnel en satisfaisant les propriétés de forme normale. La phase de conception est une tâche complexe et de haut niveau.
- une BD relationnelle est définie par un schéma relationnel :
  - les relations (ou tables). Une relation est définie par la liste des attributs, le domaine de chaque attribut, la précision de la clé primaire ;
  - les associations (ou jointures) naturelles entre les tables ;
  - les contraintes d'intégrité : de domaine qui expriment des conditions remplies par les valeurs d'un attribut, de structure sur les clés et de référence qui expriment des propriétés pour les associations et des contraintes liées à la dénormalisation.

Base restaurants

- Les relations : `Restaurants(numrestaurant, nomrestaurant, ...)`, `Typecuisine(numtypecuisine, ...)`, ...
- Les associations : `Restaurants.reftypecuisine` avec `Typecuisine.numtypecuisine`, ...
- Les contraintes d'intégrité : : `Restaurants.numrestaurant` toujours défini et unique, `Typecuisine.intitule` prend ses valeurs dans la liste ("chinoise", "rapide", ...), `Restaurants.reftypecuisine` peut être NULL mais doit prendre ses valeurs dans les valeurs de `Typecuisine.numtypecuisine`
- La sémantique est donnée par les relations, les noms et contenus des champs, les associations ou jointures et précisée dans un dictionnaire des données,
- Il est difficile de faire évoluer un modèle (voir exercices).

Une base réelle



Conclusion sur le modèle relationnel

Critiques du modèle

- modèle le plus répandu de bases de données avec des outils très murs et très performants,
- un langage expressif et normalisé (avec des dialectes),
- capable de maintenir la cohérence des données dans un cadre multi-utilisateurs (avec des accès concurrents),
- modèles de données complexes, souvent plusieurs centaines de tables,
- les schémas sont peu évolutifs et l'intégration de données et l'évolution de schémas sont un secteur d'activités,
- la sémantique reste externe au modèle et est parfois difficile à appréhender.

Retour au Web sémantique

Rappel des challenges

Représenter les connaissances dans un monde ouvert (tout type de connaissance), à l'échelle du WEB et avec une très large variété de protocoles, de langues, de culture, ... pour pouvoir manipuler les connaissances liées au contenu du WEB.

Le besoin pour les données

- les données doivent contenir leur sémantique pour pouvoir être réparties et utilisées indépendamment de leur définition,
- comme le Web évolue, il faut pouvoir constamment ajouter de nouvelles données et de nouvelles connaissances.

Retour sur les restaurants

Modèle relationnel

- Modèle avec une sémantique globale au modèle
- Modèle difficile à maintenir et faire évoluer

Reprenons l'étude et les besoins

- on souhaite modéliser des restaurants et des bars ayant des propriétés comme type de cuisine et spécialité,
- on souhaite aussi pouvoir retrouver leur localisation géographique,
- on souhaite modéliser les heures d'ouverture.

Des triplets pour les restaurants et lieux

Des triplets pour les bars et restaurants

- Supposons que l'on puisse identifier les bars et restaurants,
- on peut alors représenter les différentes propriétés par des triplets de la forme :
  (Venue1, name , "Deli Llama")
  (Venue1, cuisine , "Deli")
  (Venue1, price , "$")

Des triplets pour les lieux

- Supposons que l'on puisse identifier les lieux,
- on obtient alors :
  (Loc1, containedBy , "San Francisco")
  (Loc1, name , "North Beach")

Un modèle de triplets pour les restaurants

On peut alors relâcher la sémantique de la première colonne et dire que cette première colonne identifie un objet du monde (un restaurant, un lieu, un artiste, ...). On peut reprendre le modèle précédent et même ajouter la localisation de nos restaurants et bars. On obtient alors un modèle de triplets :

(S1, name , "Deli Llama")
(S1, cuisine , "Deli")
(S1, price , $)
(S4, containedBy , "San Francisco")
(S4, name , "North Beach")
(S1, hasLocation , S4)

Un modèle de triplets pour les restaurants

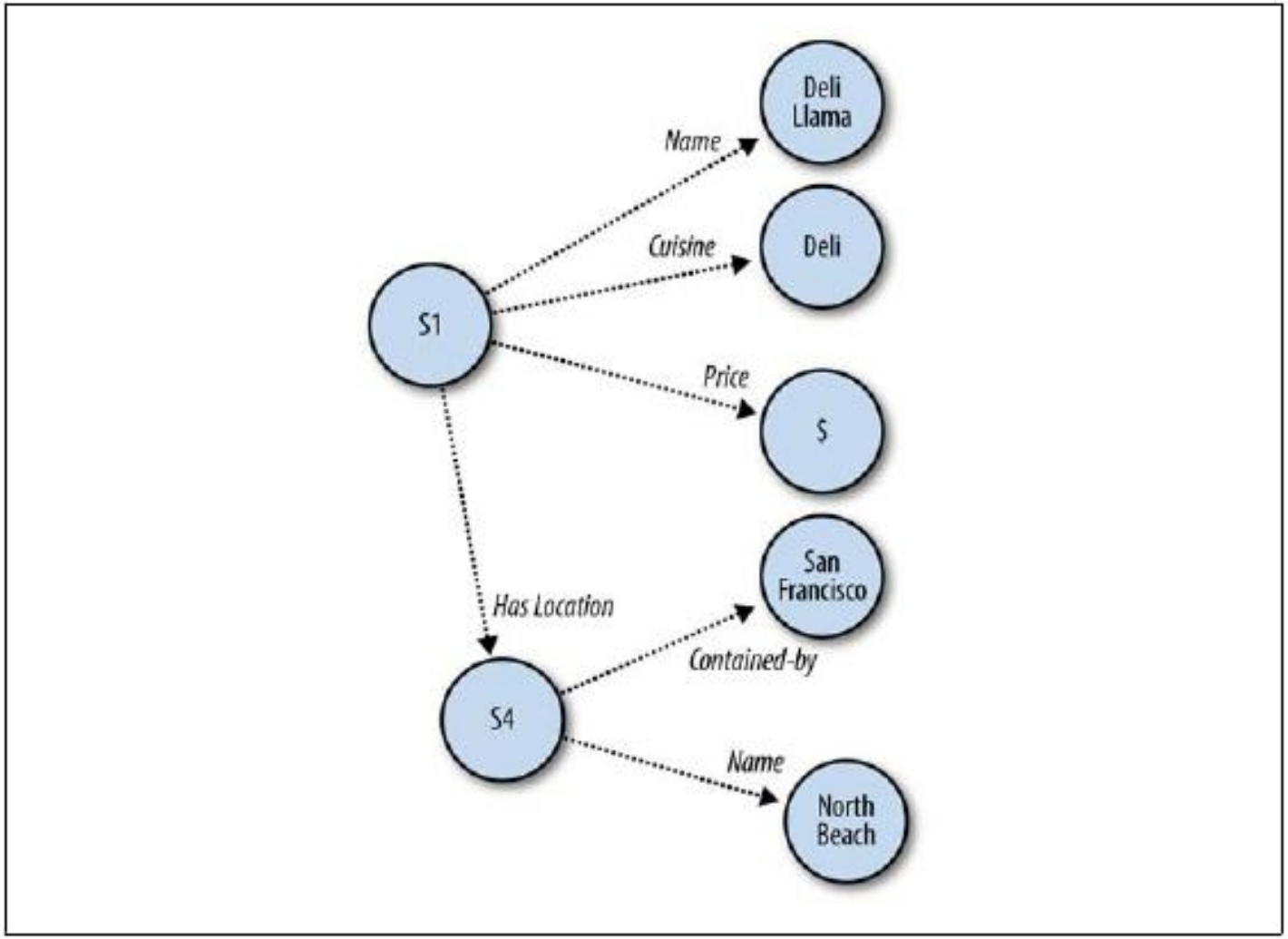


Figure 2-2. A graph of triples showing information about a restaurant

Le modèle de triplets

Un triplet est de la forme

(sujet, prédicat, objet)

où

- le sujet est une entité du monde représentée par son identifiant,
- le prédicat est une propriété de l'entité,
- l'objet est la valeur de la propriété de l'entité désignée en sujet et de la propriété donnée en prédicat. L'objet peut être une valeur ou un identifiant d'entité.

Le modèle de triplets

spo : (sujet, prédicat, objet)

Premières réflexions sur le modèle

- Le modèle de triplets peut être représenté par un graphe car sujet et objet peuvent désigner des entités. On parle aussi de modèle graphe.
- La sémantique est contenue dans les triplets : l'entité désignée par S1 sert de la cuisine indienne, ...
- Les données sont auto-décrites avec pour seules règles la forme de triplets et la notion d'identification d'entité.
- Le modèle est évolutif car on peut ajouter des entités, des nouvelles propriétés, ...

Le modèle de triplets

spo : (sujet, prédicat, objet)

Premières réflexions sur le modèle

- Le modèle de triplets peut être représenté par un graphe car sujet et objet peuvent désigner des entités. On parle aussi de modèle graphe.
- La sémantique est contenue dans les triplets : l'entité désignée par S1 sert de la cuisine indienne, ...
- Les données sont auto-décrites avec pour seules règles la forme de triplets et la notion d'identification d'entité.
- Le modèle est évolutif car on peut ajouter des entités, des nouvelles propriétés, ...
- MAIS la redondance n'est pas gérée, les entrepôts seront de grande taille et il faut gérer l'efficacité, le modèle dépend fortement de la notion d'identifiant sur les entités du monde.

Plan

Introduction

Nous avons introduit le modèle de données en triplets de la forme

spo : (sujet, prédicat, objet)

qui permet d'exprimer de la connaissance comme dans une phrase sujet verbe complément. Les entrepôts peuvent être interrogés, reliés, enrichis et évoluer dans le temps. Il faut passer à l'échelle du WEB.

Les questions

- Comment gérer les identifiants à l'échelle du WEB ?
- Comment savoir quelles propriétés (quels prédicats) utiliser ?
- Comment savoir ce que représente une valeur, c'est-à-dire son type, l'unité, la langue, ...) ?
- Comment publier, échanger, interroger les données ?

Contenu de la section : RDF, SPARQL

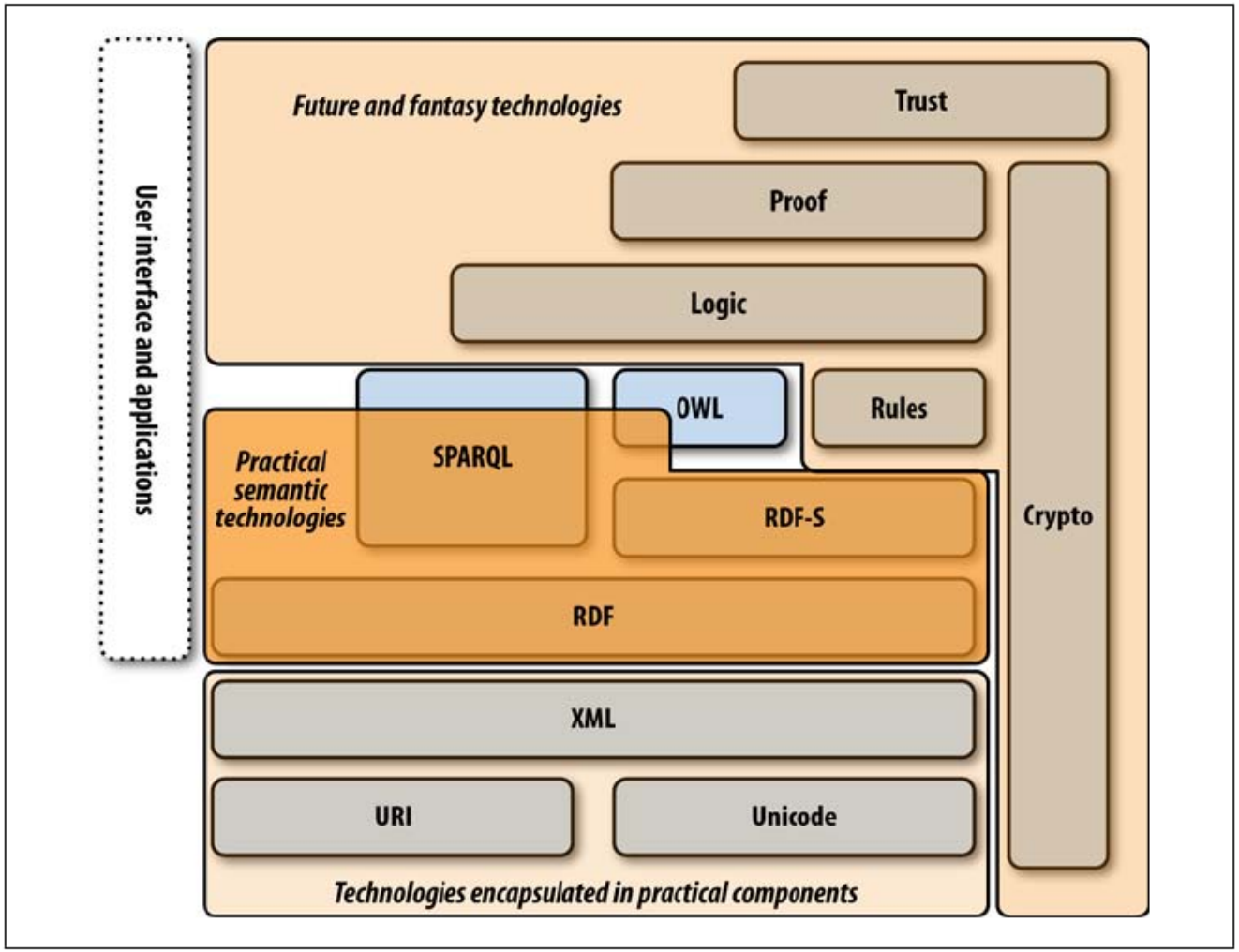


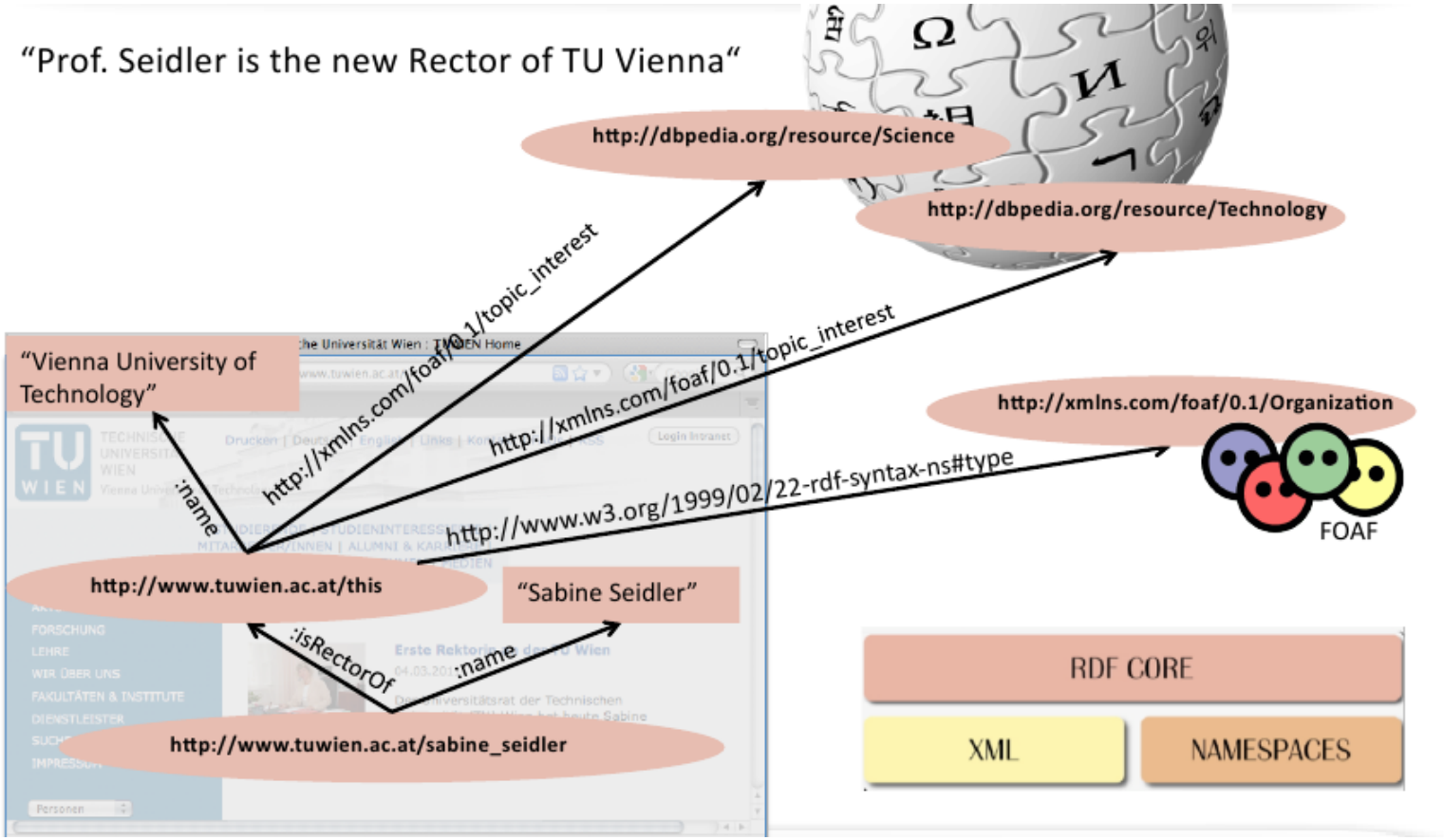
Figure 11-2. A practical view of the semantic stack

RDF : langage de description du Web sémantique

- RDF pour Resource Description Framework développé par le W3C et basé sur le
- RDF data model qui est le modèle introduit précédemment, c'est-à-dire le modèle d'entrepôts de triplets de la forme (s,p,o).
- La norme RDF spécifie également la façon de décrire les ressources (sujets et objets) et les prédicats,
- la norme RDF spécifie également les formats d'échange de fichiers de ressources sur le WEB (sérialisation)

RDF : un langage pour rendre les machines intelligentes

RDF sur un exemple



RDF : identification des ressources

La norme RDF conçoit toute chose de l'univers comme une ressource qui peut être identifiée par une URI

Les URIs

- une URI (Uniform Resource Identifier) identifie toute ressource qu'elle soit accessible électroniquement ou pas. Exemple : http ://fr.dbpedia.org/page/Lille,
- toute URL est une URI,
- L'URI n'est pas l'objet mais l'identifiant d'un objet qui peut être représenté dans un graphe RDF
- Les URIs sont faites pour les machines, cependant les espaces de nom (éventuellement locaux) vont permettre de simplifier les écritures.

RDF : et si la ressource n'a pas d'URI ?

Il se peut que l'on ne dispose pas d'URI pour la ressource dont on souhaite parler. Par exemple, dans les réseaux sociaux, on ne dispose pas d'URI pour les membres. On crée alors un noeud vide (noeud blanc, noeud anonyme, "blank node").

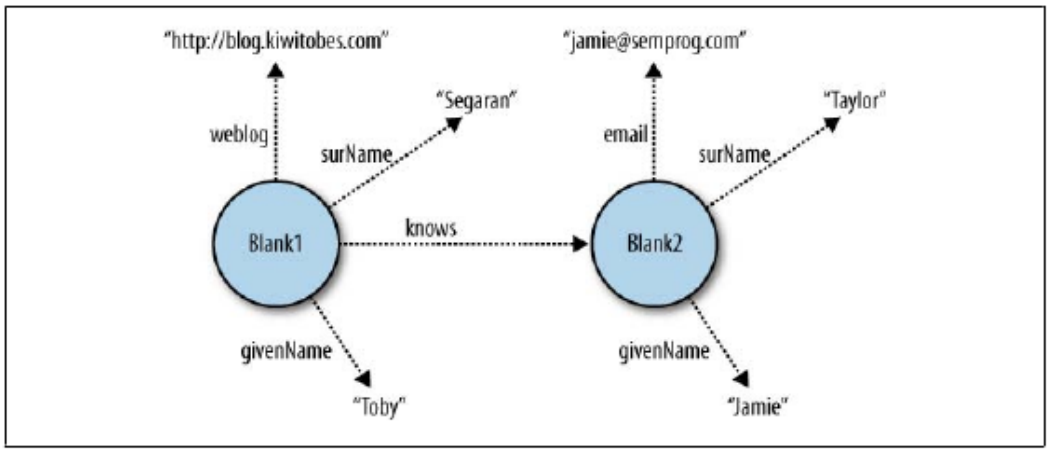


Figure 4-2. Blank nodes in a social graph

Une personne de nom Toby Segaran auteur du blog kiwitobes connaît une personne nommée Jamie Taylor dont l'email est jamie@semprog.com

sémantique et syntaxe pour les noeuds vides

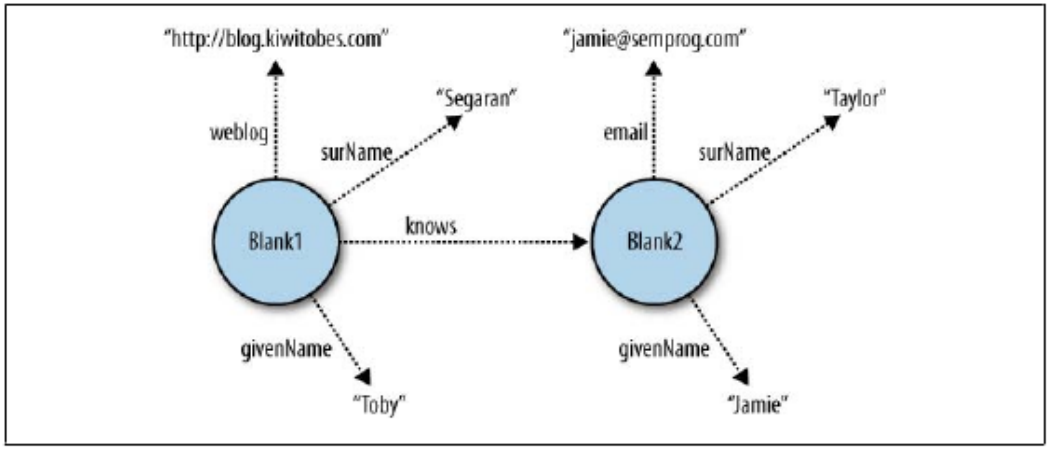


Figure 4-2. Blank nodes in a social graph

Les noeuds vides

- La sémantique : $\exists X$, $\exists Y$, (X, givenName, "Toby"), ..., (X, knows, Y), ..., (Y, email, "jamiesemprog.com")
- La syntaxe : noeuds vides abstraits par un nom de variable `_:id`. Ce qui donne (`_:person1`, givenName, "Toby"), ..., (`_:person1`, knows, `_:person2`), ..., (`_:person2`, email, "jamiesemprog.com")

Autre utilisation des noeuds vides

Les noeuds vides sont également utilisées pour grouper des informations comme dans l'exemple ci-dessous

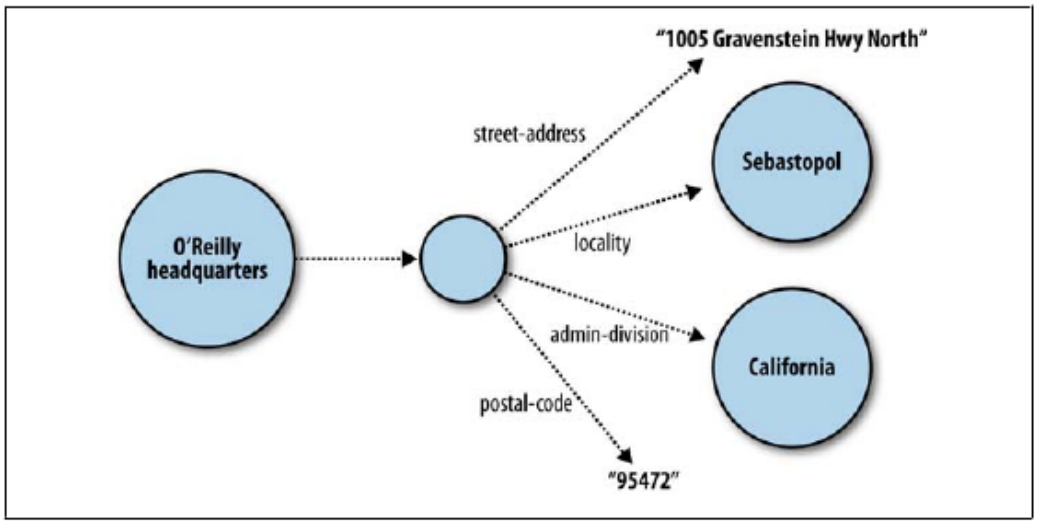


Figure 4-3. Using a blank node to model a mailing address

Les valeurs en RDF

Les valeurs littérales

- peuvent être typées
- avec la spécification de XML schema. Voir par exemple http ://www.w3.org/2001/XMLSchema#int,
- elles peuvent être exprimées dans une langue, en général,
- en respectant souvent la norme ISO 639 (en, fr, ja, ...).

Résumons

Un modèle de données en triplets de la forme spo : (sujet, prédicat, objet) représenté avec le langage

RDF

dans lequel

- les ressources sont identifiées par des URIs, les URIs peuvent être simplifiées par des espaces de nom, les URIs peuvent référer à des vocabulaires (rdf, foaf, dbpedia, ...),
- ou représentées par des noeuds vides avec des variables locales `_:id`,
- les valeurs peuvent être typées et associées à une langue.

Voyons maintenant comment échanger des entrepôts de triplets avec RDF

Sérialisation en RDF

Pour échanger des données de triplets, il faut les sérialiser dans des fichiers en respectant des contraintes communes pour émetteur et receveur.

Les principaux formats

- N-triples : les triplets sont échangés dans leur entièreté. Tout est représenté par URI sauf les valeurs et les noeuds vides
- N3, Turtle : on condense l'écriture avec l'utilisation de préfixes, des factorisations et des raccourcis de notation
- RDF/XML : syntaxe XML pour décrire les fichiers RDF. Cette syntaxe est très utilisée mais pas facile à lire (par l'humain).

Illustration par l'exemple

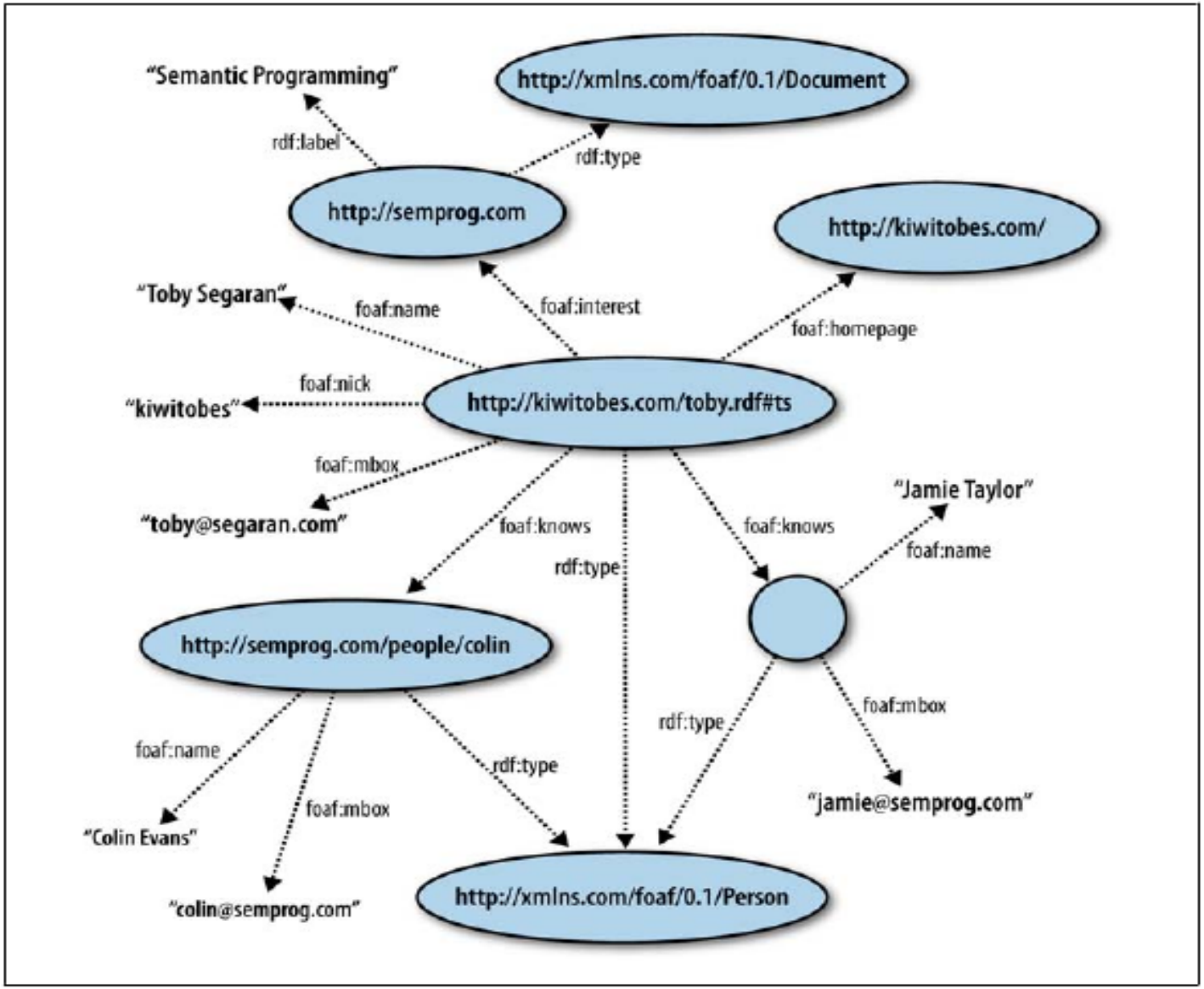


Figure 4-4. Toby's FOAF graph

Format Ntriples

Consulter le fichier exemple au format ntriples qui contient les triplets séparés par des .

Format N3

Consulter le fichier exemple au format n3. On constate

- la définition de préfixes et leur utilisation dans le document,
- la factorisation de sujet avec le ;. On peut factoriser sujet, prédicat avec la ,
- les noeuds vides peuvent ne pas être désignés en sujet,
- a est un raccourci pour `rdf:type` et = est un raccourci pour `owl:sameAs`,
- le typage des littéraux avec `^^`, par exemple `^^`xs :string,
- la précision de la langue avec `@`.

Format RDF/XML

Consulter le fichier exemple au format RDF/XML . On constate que

- l'entête permet de déclarer des espaces de nom,
- on a ensuite une suite de descriptions, une description décrit un chemin dans le graphe RDF, une description commence par un noeud en utilisant `rdf:Description`, l'URI est spécifiée par un `rdf:about`
- le choix de description du graphe par des chemins est libre
- `<foaf:Person rdf:about="..."> ... </foaf:Person>` est un raccourci pour `<rdf:Description rdf:about="..."> <rdf:type><foaf:Person> ... </rdf:Description>`,
- le typage et précision du langage "à la XML",
- et beaucoup d'autres trucs et astuces qui le rendent peu lisible.

résumons

Un modèle de données en triplets de la forme spo : (sujet, prédicat, objet) représenté avec le langage

RDF

pour lequel

- les ressources sont identifiées par des URIs, les URIs peuvent être simplifiées par des espaces de nom, les URIs peuvent être définis dans des vocabulaires (rdf, foaf, dbpedia, ...)
- ou représentées par des noeuds vides avec des variables locales `_:id`,
- on peut associer un type et une langue,
- il existe des formats (Ntriples, N3, XML/RDF) de sérialisation et d'échange de données RDF.

Voyons maintenant comment interroger des entrepôts de triplets

Un langage de requêtes de bases de données graphes

- SPARQL ("Simple Protocol And Rdf Query Language") est le langage standardisé d'interrogation de graphes RDF ,
- comme SQL est le langage d'interrogation de bases de données relationnelles.
- Alors que SQL est basé sur la notion de calcul relationnel et les requêtes sont exprimées dans un langage logique de description du résultat : les champs sélectionnés, les jointures, les filtrages, les groupes et les opérations de groupe,
- le langage SPARQL est basé sur la notion de graphes de triplets et les requêtes sont décrites par des motifs (patterns) et des variables.

Forme générale d'une requête SPARQL

Trois requêtes principales

`SELECT` pour interroger, `CONSTRUCT` pour ajouter de nouveaux triplets, et `ASK` pour tester une propriété

L'instruction SELECT

```
BASE prefix :<namespace-URI >
PREFIX prefix :<namespace-URI >
SELECT variable-list
FROM source-list
WHERE pattern
      ORDER BY expression
      LIMIT integer > 0
      OFFSET integer > 0
```

où prologue, head et body

Proloque et Head d'une requête SPARQL

Prologue

- PREFIX suivi d'une déclaration d'un raccourci pour des espaces de nom. Il est courant d'avoir plusieurs préfixes.
  Exemple : PREFIX fb :<http ://rdf.freebase.com/ns/>
- BASE suivi d'un raccourci pour une URI de base auxquelles les URIs seront concaténées. Une seule base par requête.
  Exemple : BASE <http ://www.grappa.univ-Lille3/data/>

Head

- SELECT suivi d'une liste de variables dont on souhaite connaître les valeurs répondant à la requête. Note : la requête peut utiliser d'autres variables.
- CONSTRUCT suivi d'un template de triplets pour construire un entrepôt des triplets répondant à la requête.
- ASK pour tester s'il existe des solutions vérifiant les conditions de la requête.

Une première requête SPARQL

Exemple courant

Base Freebase de films avec les prédicats
film.film.directed_by entre identifiant de film et de personne,
film.film.starring entre identifiant de film et de personne,
film.film.initial_release_date entre identifiant de film et date,
type.object.name entre identifiant d'objet et son nom
préfixés par <http ://rdf.freebase.com/ns/>

Une requête

```
PREFIX fb :<http ://rdf.freebase.com/ns/>
SELECT ?who ?film
WHERE{
        ?film fb :film.film.directed_by ?who .
        ?film fb :film.film.starring ?who }
```

qui fait quoi ?

Le corps d'une requête SPARQL

- FROM est optionnel et permet de spécifier différents graphes de triplets dans lesquels il faut chercher. Sinon, on suppose qu'un graphe a été spécifié au processeur SPARQL.
- WHERE suivi d'une déclaration de pattern (motif) qui peut contenir plusieurs patterns, plusieurs patterns de base, des filtres, des unions, des présences optionnelles.
- ORDER BY comme son nom l'indique
- LIMIT pour donner un nombre maximal de réponses
- OFFSET pour commencer à partir d'un numéro de réponse

Patterns de base en SPARQL

sont des triplets contenant des noms de variables.
Syntaxe : les triplets sont séparés par le symbole .
Sémantique : un tuple de variables satisfait le pattern si tous les triplets sont satisfaits par le tuple. On parle de requête conjonctive

Exemple

- Afficher les noms des artistes jouant dans un film dirigé par "Ron Shelton". Note : on peut spécifier Distinct dans le SELECT.
- Même question avec la date (de parution du film)
- Afficher les noms des artistes ayant joué dans un film dont ils étaient le directeur

Les contraintes OPTIONAL et FILTER

- OPTIONAL permet d'utiliser l'information requise si elle est présente et de ne pas éliminer les solutions où elle est absente. Très important pour le WEB sémantique
- FILTER permet de poser des conditions de filtrage sur les solutions trouvées avec les opérateurs de comparaison usuels, BOUND() qui teste si une variable est liée à une valeur, REGEX() pour test d'expression régulière.

Exemple

- Afficher les noms des artistes jouant dans un film dirigé par "Ron Shelton" avec la date quand elle est présente et vide sinon
- Afficher les identifiants des films sortis après 2002
- Afficher les identifiants des films n'ayant pas de date de sortie
- Les noms des acteurs contenant "Russ"

Combinaison de patterns

Conjonction de patterns

Un pattern complexe peut contenir plusieurs patterns de base.
Syntaxe : les patterns de base sont entre accolades. Note : les patterns de base peuvent contenir des filtres.
Sémantique : les tuples solutions sont les tuples qui sont solution de tous les patterns. C'est-à-dire qu'on effectue l'intersection des solutions.

Disjonction de patterns

On peut faire l'union des solutions de patterns de base en ajoutant UNION entre les patterns de base.

Calcul d'une requête SPARQL

1. Le processeur construit le graphe correspondant aux sources du FROM,
2. il construit les tuples de variables qui satisfont dans le graphe les conditions décrites dans le pattern donné dans le WHERE,
3. si le pattern est complexe on effectue une intersection des solutions ou une union selon le cas,
4. enfin, il Restreint les résultats aux variables précisées dans le SELECT, ou plus généralement dans l'entête de la requête

Remarques importantes :

- Attention, cet ordre est important à retenir pour écrire vos requêtes SPARQL. En particulier si vous utilisez des négations,
- l'ordre des réponses ne peut pas être connu à l'avance,
- les résultats peuvent être de très grande taille. Pensez donc à utiliser LIMIT et OFFSET dans vos tests,
- on utilise des requêtes SPARQL pour explorer un entrepôt de triplets : quelles propriétés utilisées ? Quels sujets possibles pour un prédicat ? Quels objets possibles pour un prédicat ? ...

Conclusion sur SPARQL

- SPARQL intégré dans des plate-formes de WEB sémantique comme ARQ dans JENA de APACHE ou
- entrepôts gérés dans une base de données et interfaçage entre le SGBD et SPARQL
- les implantations existantes sont très diverses et, de plus,
- SPARQL est une norme qui évolue :
  - introduction de EXISTS (remplace BOUND),
  - introduction de DELETE et UPDATE
  - introduction de patterns élaborés avec des chemins (utilisation de + et *) pour enrichir la description par triplets.

Plan

Contenu de la section : RDFS, OWL et inférence

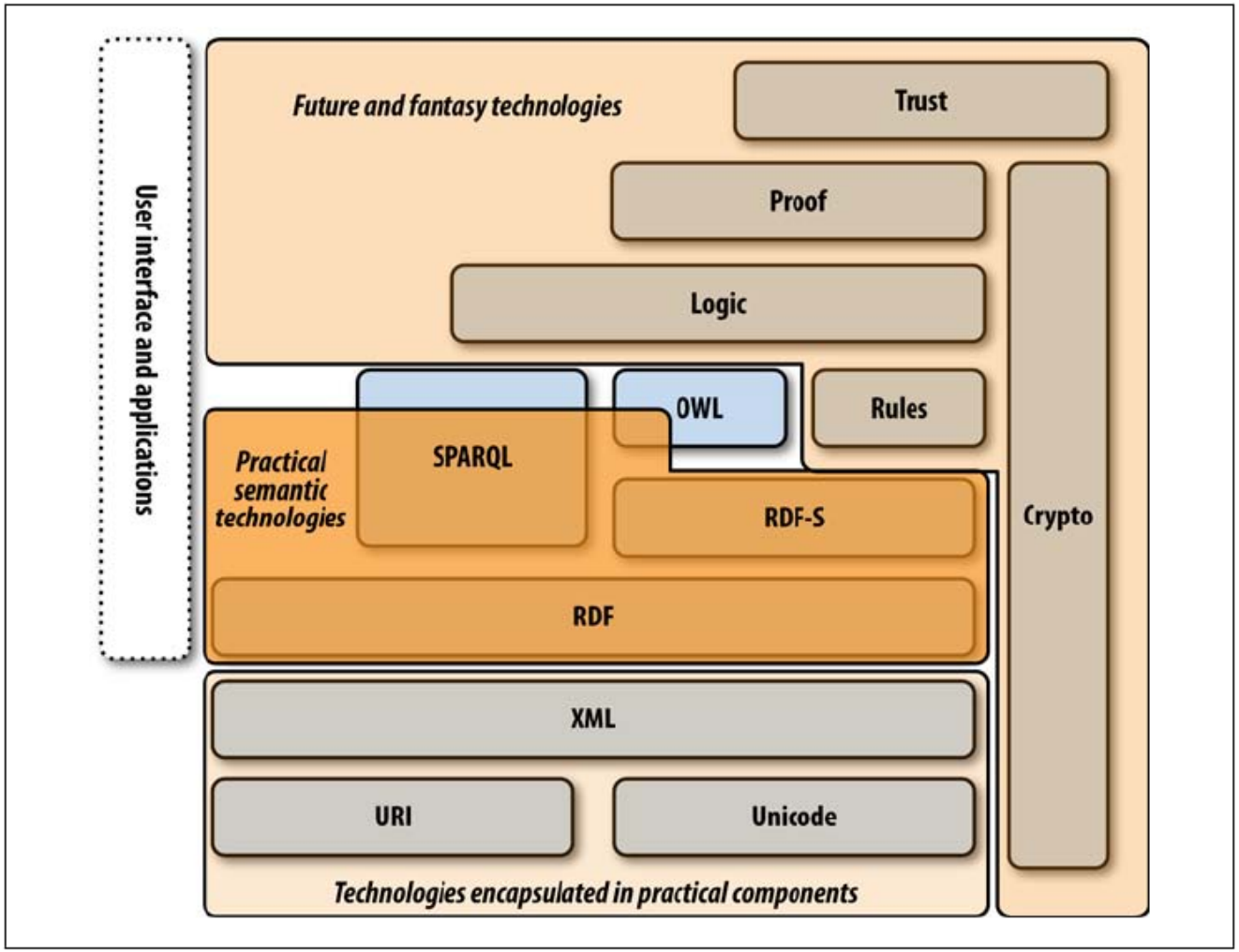


Figure 11-2. A practical view of the semantic stack

Tentative de définition

Introduction

Nous avons utilisé des définitions non formelles pour des domaines de sens commun avec une sémantique intuitive en utilisant des propriétés telles que `foaf:knows` ou `fb:film.film.starring` ou des caractérisations d'entités telles que `foaf:Person`.
Mais qu'est-ce qu'un film ? Un film de cinéma, de télévision, un court-métrage ou une video ? Comment va-t-il être identifié ? Quelles sont les propriétés que peut avoir un film ? Avec quelles valeurs de sujet ? ...

Définition

Une ontologie est une description formelle de la signification du vocabulaire utilisé dans un document RDF (ou un graphe de triplets ou un entrepôt de triplets). Le terme ontologie est issu de la philosophie.

Utilisation des ontologies

Quelques caractéristiques

- Une ontologie pour le Web sémantique concerne les propriétés, les classes des sujets et objets des triplets, les entités qui peuvent être représentées,
- une ontologie peut être vue comme un contrat entre le producteur de données et le consommateur des données,
- une bonne ontologie est utilisée, elle ne doit être ni trop simple, ni trop complexe,
- une ontologie est exprimée en RDF, souvent dans le même graphe de triplets : rdf vocabulary to describe rdf vocabularies

Ontologies et inférence

Une ontologie est une description formelle de la signification du vocabulaire. Une ontologie permet d'inférer de nouveaux triplets et d'enrichir le résultat des requêtes sous réserve du logiciel.

Des exemples d'inférence

en RDFS

- Robert Redford est typé comme `Actor`, `Actor` est une sous-classe de `Person` qui est une sous-classe de `Object`. On infère que Robert Redford est une instance des classes `Person` et `Object`.
- La relation `starring` a pour domaine `Actor`. Marion Cotillard est sujet d'un triplet avec le prédicat `starring`. On infère que Marion Cotillard est une instance de la classe `Actor`.

en OWL

- Marion Cotillard est une instance de la classe `Actor`. On infère que Marion Cotillard est une instance de la classe `Person`. Marion Cotillard est de sexe `Female`. On infère que Marion Cotillard est une instance de la classe `Woman` définie comme intersection des classes `Person` et `Female`.

Les ontologies en pratique

rdfs et owl

sont les deux langages principaux pour exprimer des ontologies

- RDFS (RDF Schema) permet de définir des vocabulaires RDF. Il est simple, d'une expressivité réduite et il permet de réaliser des inférences simples.
- OWL (Web ontology language) est la dernière norme du W3C. C'est un langage expressif (trop ?) permettant de réaliser des inférences complexes (trop ?).

Définir des ontologies

- Utilisation du logiciel Protege,
- on définit des ontologies de domaines spécialisés et on
- utilise au maximum les ontologies existantes et reconnues,
- on trouve le meilleur compromis entre simplicité et expressivité.

Les classes et propriétés en RDFS

Les classes

- Les ressources peuvent être partagées en classes,
- on précise la classe d'une ressource avec la propriété `rdf:type`,
- on peut exprimer qu'une classe est sous-classe d'une autre qui signifie que toutes les instances de la première sont instances de la seconde.

Les propriétés

- Une propriété est de type `rdf:Property` qui est une instance de `rdfs:Class`,
- on peut exprimer qu'une propriété est sous-propriété d'une autre qui signifie que toutes les paires d'instances vérifiant la première vérifient la seconde,
- on peut spécifier le type du domaine (l'ensemble des valeurs de sujet possibles) et le type du co-domaine (l'ensemble des valeurs d'objet possibles).

Syntaxe pour les classes et propriétés en RDFS

Les classes RDF et RDFS

- `rdfs:Resource` : classe de toutes les resssources
- `rdfs:Class` : classe de toutes les classes
- `rdfs:Property` : classe de toutes les propriétés
- `rdfs:XMLliteral` : classe de toutes les XML literal values
- et les types, les listes, les sacs, containers

Les propriétés RDF et RDFS

- `rdf:type` : précise la classe d'une ressource
- `rdfs:subClassOf` : propriété d'inclusion de classes
- `rdfs:subPropertyOf` : propriété d'inclusion de propriétés
- `rdfs:domain` : type du domaine
- `rdfs:range` : type du co-domaine
- et `rdfs:label`, `rdfs:comment`, `rdfs:first`, ...

Juste un peu de OWL

Le langage OWL

est très expressif et on se limite souvent à un sous-ensemble OWL-Lite.
Nous ne présentons ici que les concepts les plus utilisés.

Les classes et propriétés OWL

- `owl:Thing` et `owl:Class` : classes de tous les objets et de toutes les classes
- `owl:DatatypeProperty` et `owl:ObjectProperty` : classe des propriétés à co-domaine de valeurs typées et à co-domaine d'instances de classes
- `owl:FunctionalProperty` : sous-classe de `owl:ObjectProperty` des propriétés fonctionnelles (au plus une valeur pour un sujet donné)
- `owl:inverseOf` : pour exprimer qu'une propriété est l'inverse d'une autre
- et `owl:equivalentClass`, `owl:equivalentProperty`, `owl:disjointWith`, ...

RDFS et OWL en pratique sur movie (1)

Le monde à modéliser

On considère le domaine `movie` qui concerne des films de cinéma avec acteurs, directeurs, dates de sortie, ...

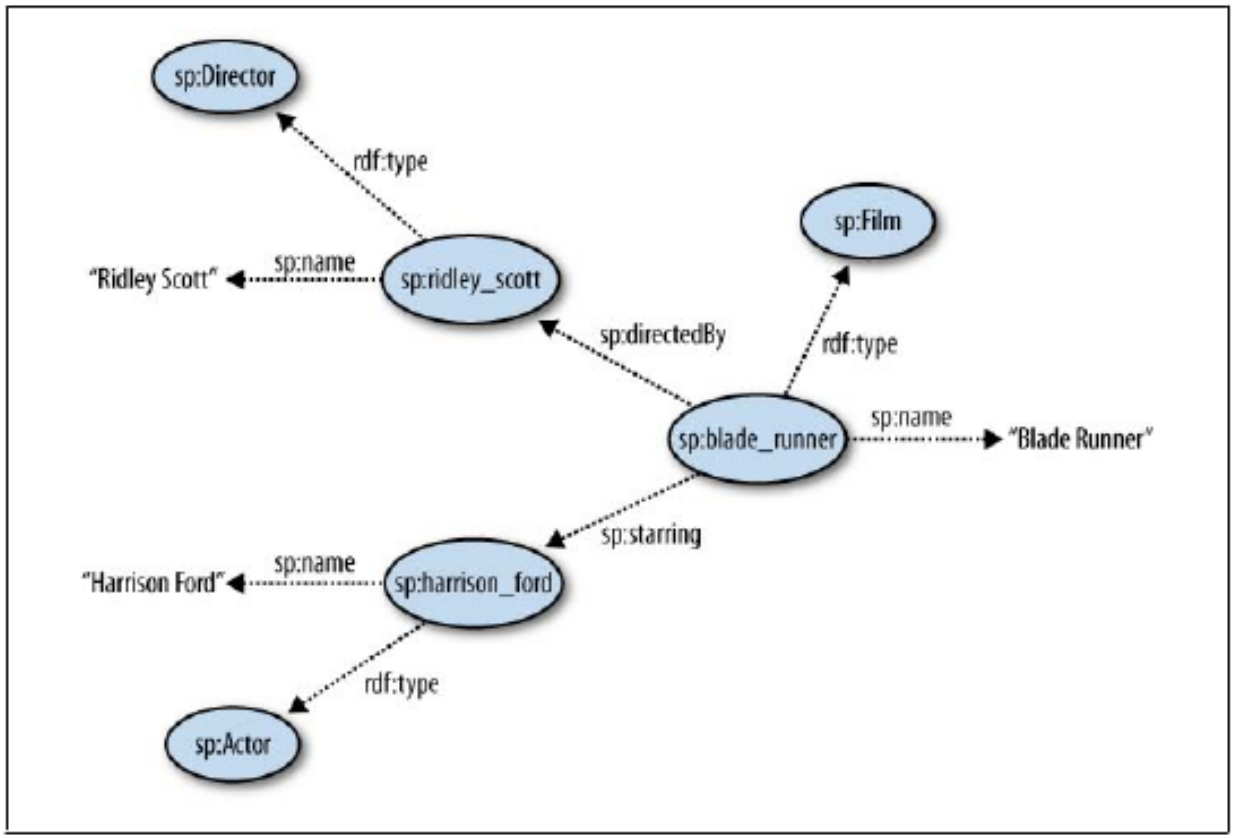


Figure 6-2. A graph describing the film Blade Runner

RDFS et OWL en pratique sur movie (2)

La hiérarchie de classes

Au vu du monde réel très simplifié, nous allons considérer les classes `Director`, `Actor` et `Film`. Dans une modélisation plus complète, vous auriez les producteurs, les professions diverses, les types de film, ... On peut les organiser dans une hiérarchie de classes.

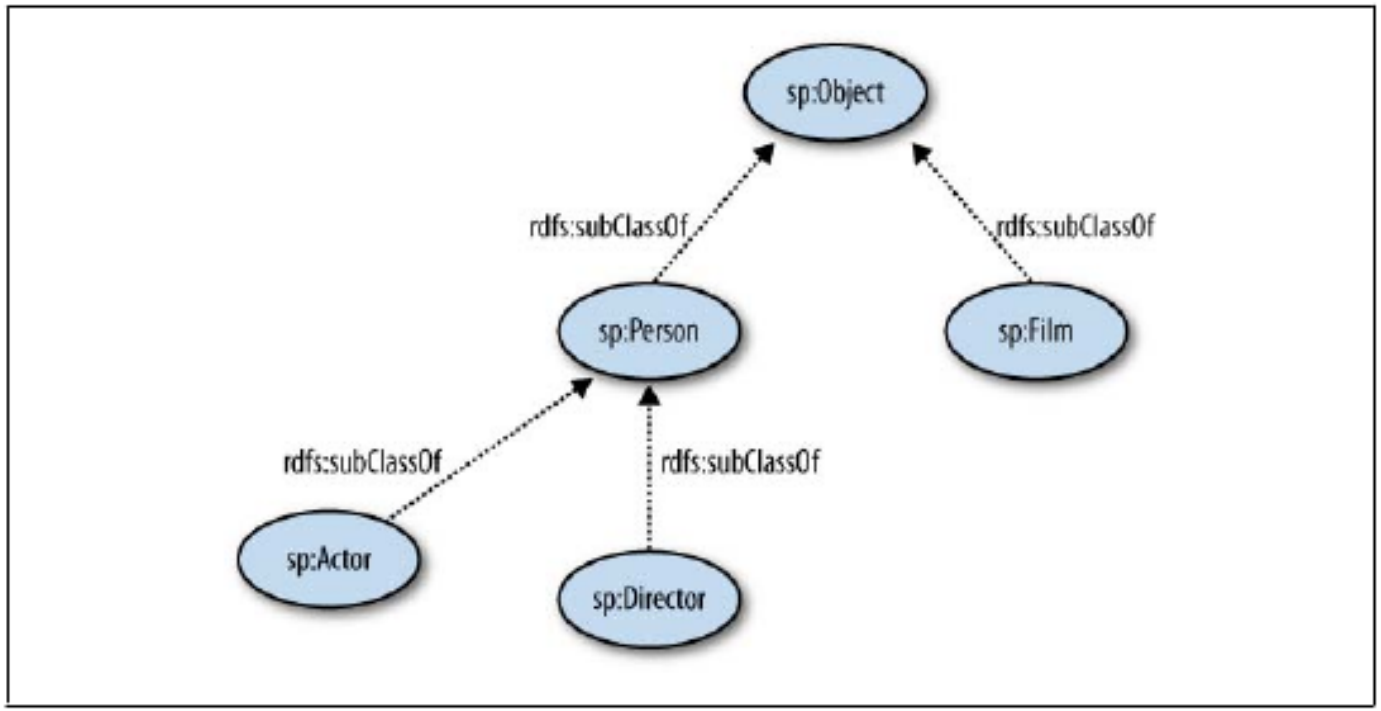


Figure 6-3. The class hierarchy for our film ontology

RDFS et OWL en pratique sur movie (3)

Les propriétés

Sur notre vue réduite du monde, nous avons les propriétés `starring`, `directedBy` et `name`. Nous pouvons alors préciser les domaines et co-domaines de ces propriétés ainsi que leur type. Note : verbosité (d'où logiciels comme Protégé), modèle dans les données, inférences.

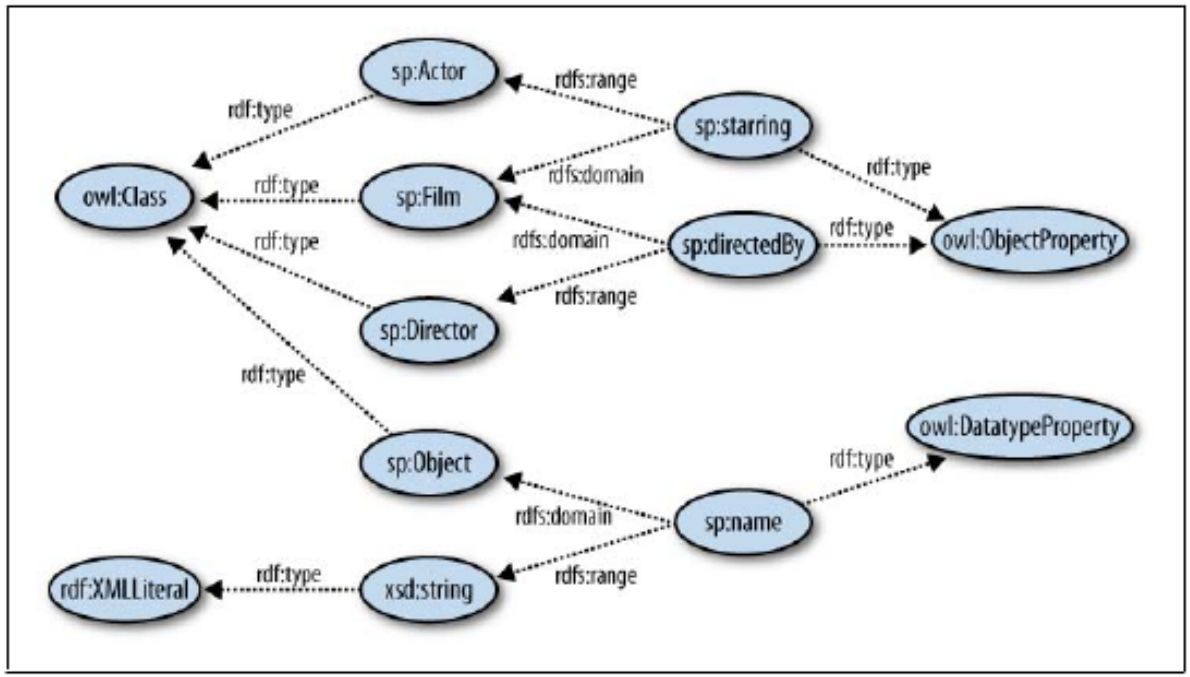
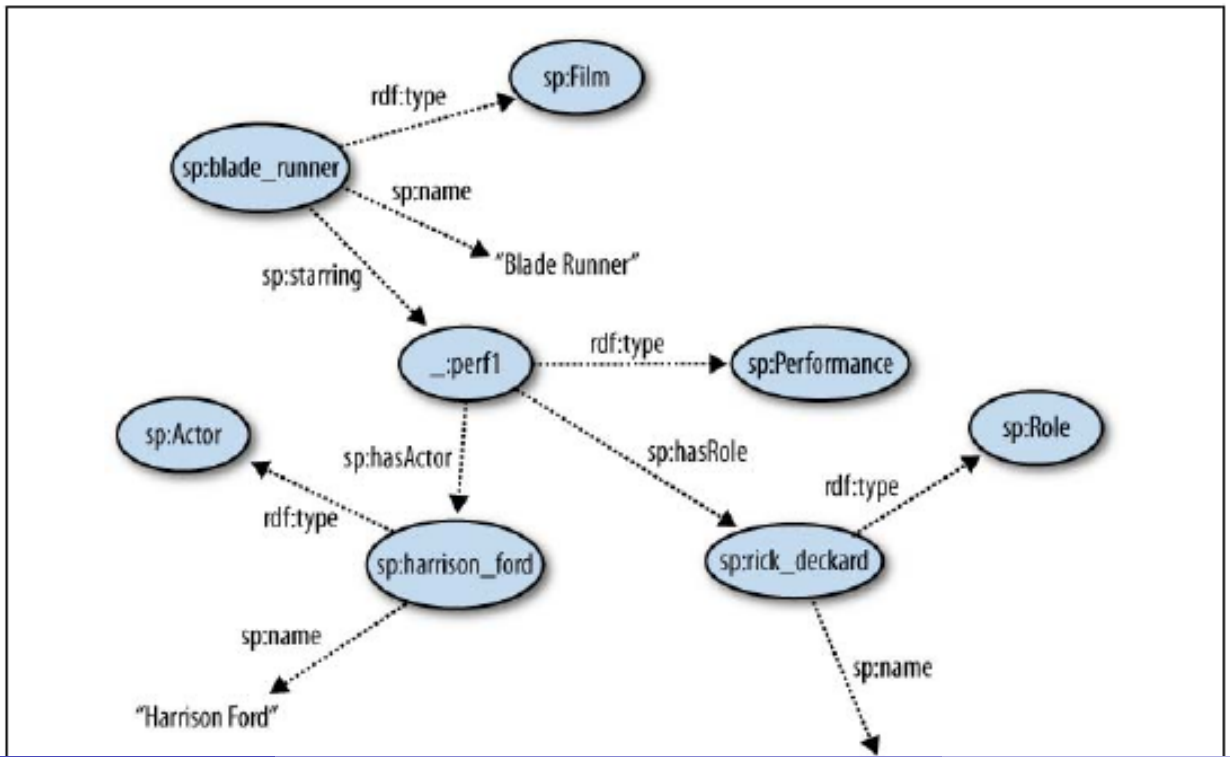


Figure 6-4. The property definitions for our film ontology

La question des associations complexes

Sur le cas movie

- Harrison Ford joue Rick Deckard dans Blade Runner. Le rôle n'est pas une propriété de l'acteur, ni du film, c'est une propriété de la performance (du jeu) de l'acteur dans le film.
- Il existe la possibilité de réification en RDF, mais on préfère modifier le schéma et introduire une classe `Performance` :



Conception d'ontologies

Logiciels

La syntaxe RDF et OWL amène assez vite un grand nombre de classes, de propriétés et de relations entre ces classes et propriétés d'où l'utilisation d'environnements logiciels comme Protégé.

Principes de base

Tâche de haut niveau comme toute modélisation

- une ontologie est pertinente si elle est utilisée,
- doit faire un compromis pour limiter le domaine de la modélisation : elle ne doit être ni trop simple, ni trop complexe,
- il faut réutiliser au maximum les ontologies de référence (Dublin Core, foaf, dbpedia, geonames, ...)
- un modèle est évolutif et on peut toujours ajouter de nouvelles classes, propriétés et relations entre elles car le Web sémantique est évolutif.

Conlusion

- Un environnement éprouvé avec en particulier la norme RDF et le langage d'interrogation SPARQL,
- des succès applicatifs comme le knowledge graph de Google, les applications avec géolocalisation pour les petits objets portables et communiquants, l'apparition de moteurs de recherche sémantique,
- des initiatives gouvernementales pour la publication de données ouvertes (démographiques, géographiques, politiques, documentaires (BNF, INA), ...).
- Questions en suspens : la publication de données individuelles (Web, CMS, ...), la publication de données par les entreprises, le développement d'applications intelligentes utilisant réellement les possibilités du Web sémantique : "1 semaine à New York pendant les prochains congés scolaires" 8-)