Graphes de données

RDF & SPARQL

Emmanuel Coquery

RDF, SPARQL et le Web sémantique

- RDF
  - format de graphe annoté par des IRI
- SPARQL
  - Langage de requête pour RDF
- Web sémantique
  - Ensemble de (méta)données codant/ formalisant de la connaissance sur des objets
  - Résidant sur le Web

Graphes étiquetés

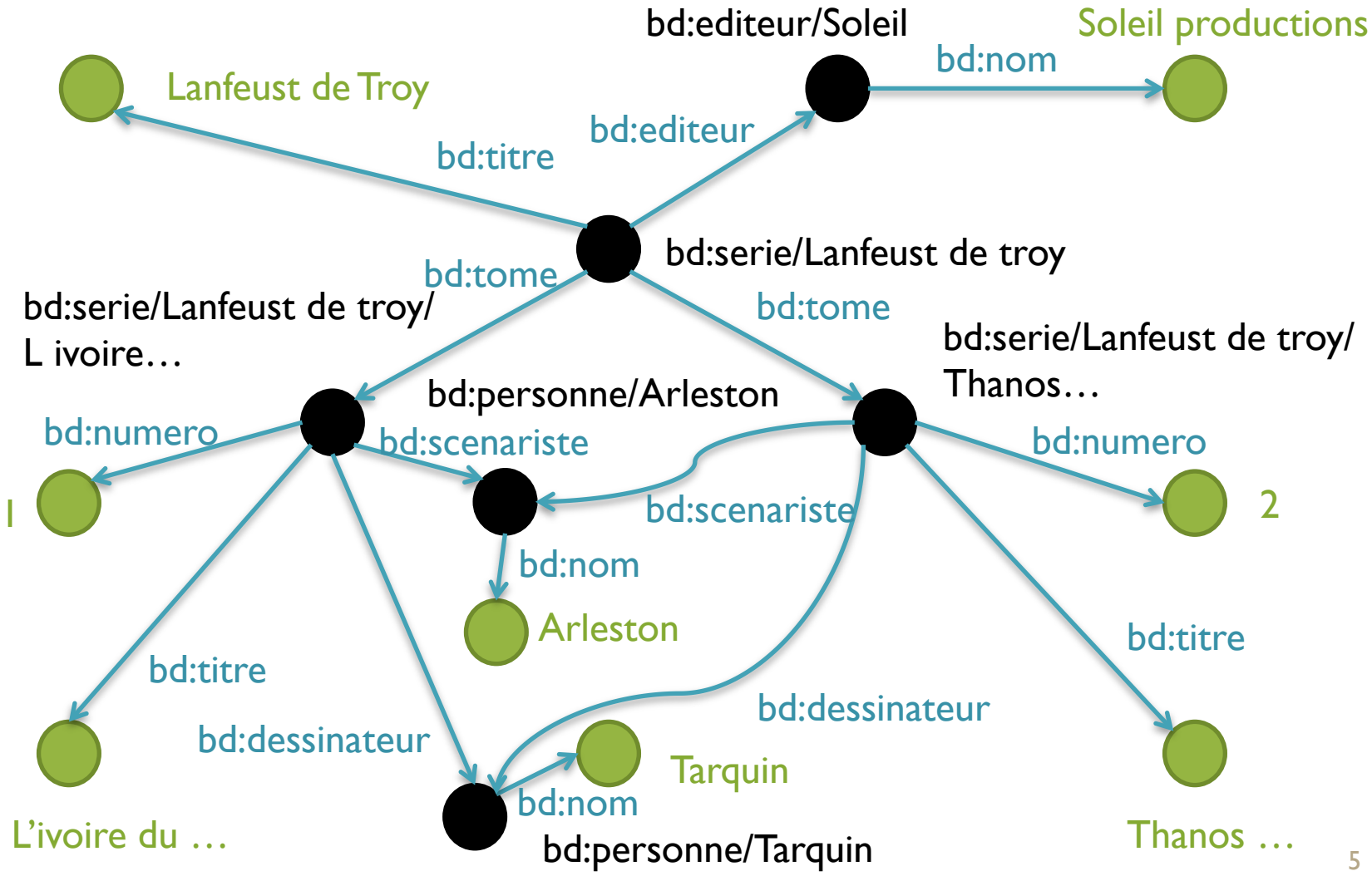
- Graphes orientés
  - Les sommets et les arêtes sont étiquetés
- Constitue un modèle de données alternatifs aux modèles:
  - Relationnel, semi-structuré, objet
- Permet de représenter aisément des liens entre des *choses* référencées par un identifiant:
  - Sommet : *chose*
  - Arête : relation entre deux *choses*

RDF: graphes pour le Web sémantique

- Standard W3C
- Graphes RDF:
  - Étiquetés (arêtes et sommets) par des IRIs (ressources)
    - Certains sommets sont étiquetés par des littéraux(valeurs)
    - Au plus 1 sommet par étiquette
    - Pas de limite sur le nombre d'arêtes par étiquette
  - La valeur de l'IRI est symbolique
    - Dans le cadre du « Linked data », on attend qu'un certain nombre d'IRI soit déréférençables

RDF: Exemple

bd: ↔ http://www.collection.com/bd/



Triplets RDF

- Description de graphe par des triplets représentant les arêtes
  - Sujet
    - Étiquette du sommet de départ
  - Prédicat (ou property)
    - Étiquette de l'arête
  - Objet
    - Étiquette du sommet d'arrivée
- Exemple:
  - (bd:Lanfeust de Troy, bd:editeur, bd:editeur/Soleil)

XML

- Syntaxe pour représenter des triplets
- rdf:Description
  - Déclaration de triplets ayant pour sujet l'IRI indiquée par l'attribut rdf:about
  - Attributs/élements:
    - Espaces de nommage + nom local = IRI du prédicat
  - Valeur/attribut rdf:resource
    - Objet
    - Littéral/IRI
  - Pour les littéraux:
    - rdf:datatype

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:bd="http://www.collection.com/bd#">

  <rdf:Description rdf:about="http://www.collection.com/bd/Lanfeust de Troy">
    <bd:tome rdf:resource="http://www.collection.com/bd/serie/Laufeust de Troy/L
ivoire du Magohamoth"/>
    <bd:tome rdf:resource="http://www.collection.com/bd/serie/Laufeust de Troy/Thanos
l incongru"/>
    <rdf:type rdf:resource="http://www.collection.com/bd/serie"/>  </rdf:Description>

  <rdf:Description rdf:about="http://www.collection.com/bd/editeur/Soleil">
    <bd:nom>Soleil Productions</bd:nom>
  </rdf:Description>

  <rdf:Description
      rdf:about="http://www.collection.com/bd/Laufeust de Troy/L ivoire du
Magohamoth">
    <bd:numero rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</bd:numero>
  </rdf:Description>
```

TURTLE

- Syntaxe alternative pour RDF
- IRI:
  - <http://www.collection.com/bd/serie>
  - bd:serie
    - PREFIX bd: <http://www.collection.com/bd#>
- Valeur:
  - "Arleston"
  - "2.5"^^xsd:float
- Triplet:
  - sujet predicat objet .
  - Sujet predicat objet; predicat objet .

Exemple

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bd: <http://www.collection.com/bd#>

<http://www.collection.com/bd/Lanfeust de Troy> bd:tome <http://
www.collection.com/bd/serie/Laufeust de Troy/L ivoire du Magohamoth> .
<http://www.collection.com/bd/Lanfeust de Troy> bd:tome <http://
www.collection.com/bd/serie/Laufeust de Troy/Thanos l incongru> .
<http://www.collection.com/bd/Lanfeust de Troy> rdf:type <http://
www.collection.com/bd/serie> .
<http://www.collection.com/bd/Lanfeust de Troy> bd:editeur <http://
www.collection.com/bd/editeur/Soleil Productions> .

<http://www.collection.com/bd/editeur/Soleil Productions> bd:nom "Soleil
Productions" .

<http://www.collection.com/bd/Laufeust de Troy/L ivoire du Magohamoth>
    bd:numero "1"^^xsd:integer ;
    bd:titre "L'ivoire du Magohamoth" ;
    bd:dessinateur <http://www.collection.com/personne/Tarquin> ;
    bd:scenariste <http://www.collection.com/personne/Arleston"> .
```

Noeuds anonymes (blank nodes)

- Pas des IRIs, ni des littéraux
- Peuvent être utilisés comme des nœuds IRI
- *Intuitivement*, deux nœuds anonymes peuvent être replacés par un même noeud

Sémantique des graphes

Pourrait être simple, mais:

- Web → monde ouvert
- Nœuds anonymes
- Interprétation des littéraux typés

Interprétations I simples

- Univers (des ressources): IR
- Propriétés: IP
- Interprétations des propriétés:
  - IEXT: IP $\rightarrow 2^{IR \times IR}$
- Interprétations des IRIS:
  - IS: IRIs $\rightarrow$ IR $\cup$ IP
- Interprétations des littéraux
  - IL: Littéraux $\rightarrow$ IR

Interprétations de graphes instanciés (sans nœuds anonymes)

- Triplet S P O
  I(S P O) = vrai si:
  - (IS(S), IS(O)) ∈ IEXT(IS(P))
- Graphe E
  I(E) = vrai si:
  - Pour tout triplet S P O ∈ E, I(S P O) = vrai

Sémantique des nœuds anonymes

- Etant données I
- Fonction A : nœuds anonymes → IR

- Extension naturelle de I(S P O)
  à [I+A](S P O)

- I(E) = vrai (***I satisfait E***) si
  - on peut trouver A t.q. [I+A](E) = vrai

Revient à trouver une valeur pour les nœuds anonymes

Types de données

Un type T:

- Espace lexical EL: ensemble de chaînes de caractères
  - *c.f.* types simples XML
- Espace (ensemble) de valeurs EV
- Fonction L2V(T): EL → EV

Ex: T = Entiers XML Schema

- EL: chaînes reconnues par `-?[0-9]+`
- EV: Z
- L2V(T): parsing des entiers

D-Interprétations

Force le « bon typage » des littéraux

- Littéraux typés: "sss"^^aaa
- IS(aaa)=T
- IL("sss"^^aaa) = L2V(T)(sss)

Exemple:

- IL("-012"^^xsd:integer) = -12
- Avec `PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>`

Implication de graphes

- Graphe E1 implique graphe E2 si
  - Pour toute I qui satisfait E1
  - I satisfait aussi E2

- Propriété:
  - E1 implique E2 ssi
  - On peut trouver une instance de E2 qui est un sous-graphe de E1

- On peut passer de E2 à E1 en
  - Remplaçant des nœuds blancs par des URIs
  - Ajoutant des triplets

Vocabulaires RDF

- Ensemble d'IRI
- Pouvant être utilisés par une application
- Ayant souvent une définition à minima informelle

- Exemple FOAF:
  - foaf:firstName (prénom)
  - foaf:knows (connaît)

Vocabulaire spécial: RDF

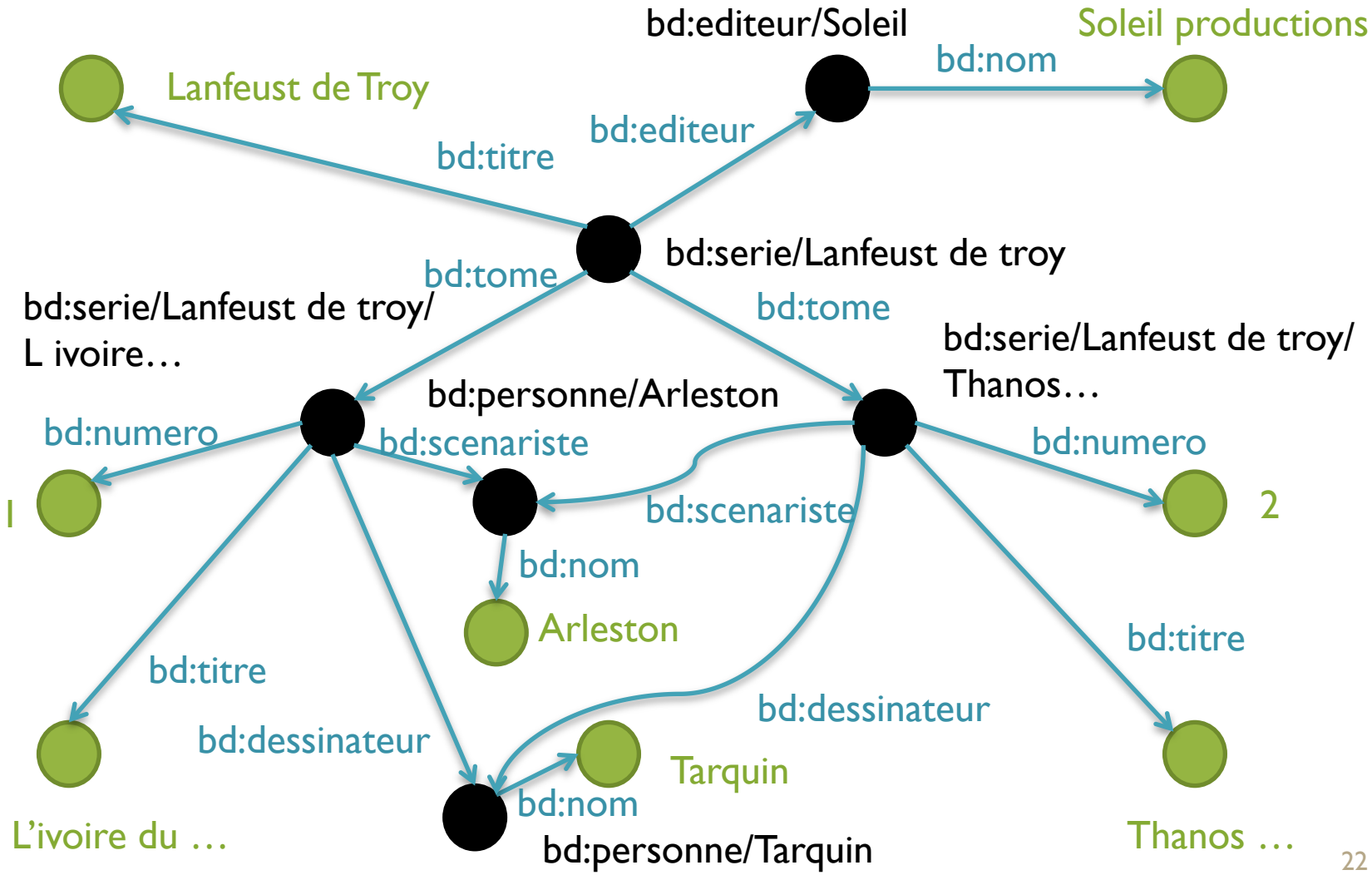
- `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- Typage:
  - rdf:type rdf:langString rdf:Property
- Réification:
  - rdf:subject rdf:predicate rdf:object
- Listes:
  - rdf:first rdf:rest rdf:value rdf:nil rdf:List
- Conteneurs:
  - rdf:_1 rdf:_2

Requête: matching de sous-graphe

- Spécifier des contraintes sur les parties du graphe à récupérer
  - « pattern matching » pour les graphes
- Pattern =
  - Un graphe exemple
  - Avec des variables
- Réponses
  - Sous-graphes du graphe de départ
  - Correspondant au pattern
    - En instanciant les variables
- Une réponse explique comment le graphe requêté implique le graphe de départ en supposant que les trous sont des nœuds anonymes

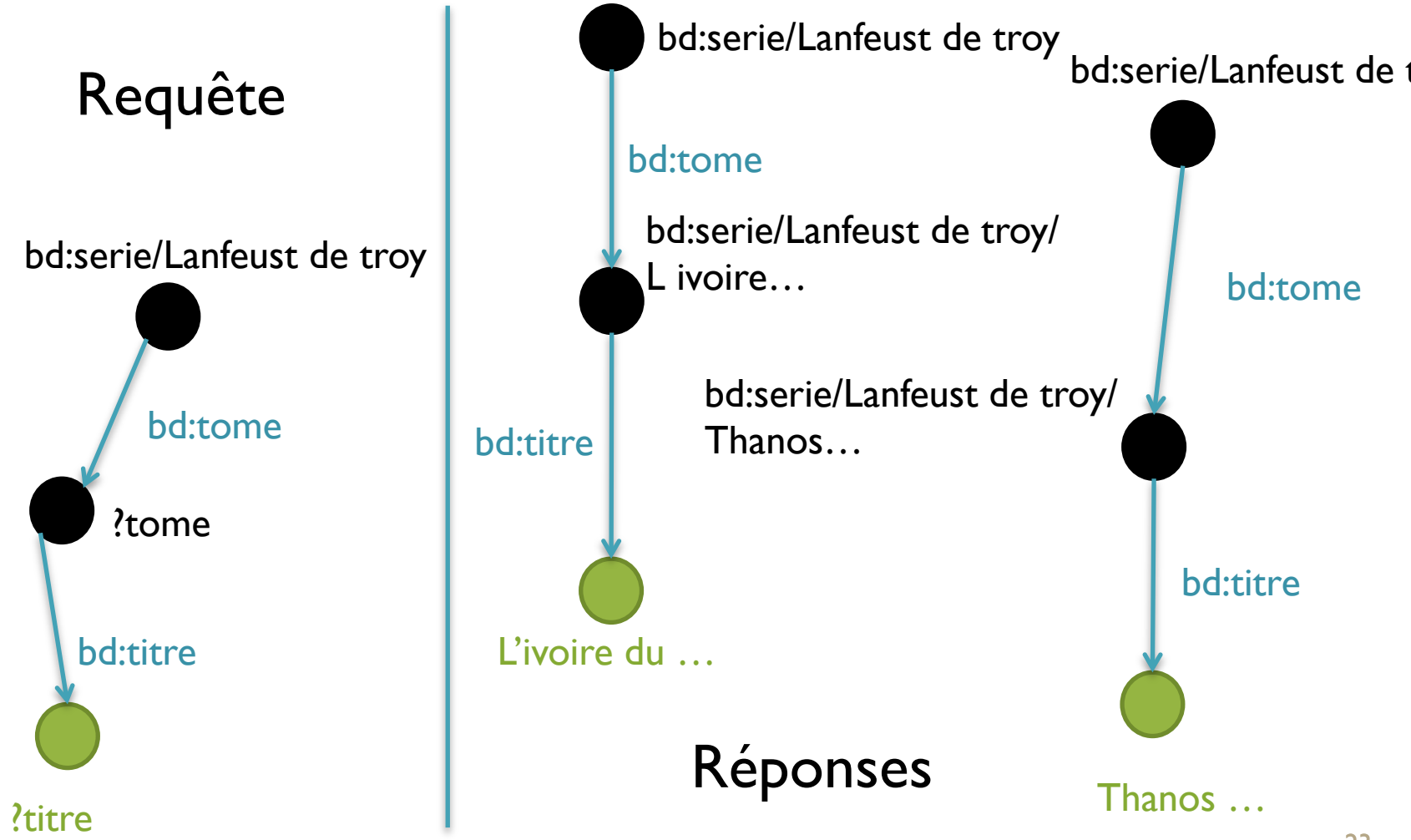
RDF: Exemple

bd: ↔ http://www.collection.com/bd/



Exemple

Quels sont les tomes de la série bd:serie/Lanfeust de troy, avec leur titre ?



SPARQL

- Langage de requête pour les graphes RDF
  - Basé sur le matching de sous-graphe
  - Standard W3C
- Syntaxe des triplets basée sur TURTLE
  - Variables: ?nomVar
  - Toute IRI/Valeur peut être remplacée par une variable
    - y compris les prédicats
- Réponse sous forme
  - D'affectation des variables du pattern (SELECT)
  - De graphe (CONSTRUCT)

Exemple

```
PREFIX bd: <http://www.collection.com/bd#>

SELECT * WHERE {
{ ?t bd:dessinateur ?p }
}
```

Donner les ?t et ?p tels que ?p est le dessinateur de ?t.

Résulat (sérialisé en XML)

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="t"/>
    <variable name="p"/>
  </head>
  <results>
    <result>
      <binding name="t"><uri>http://www.collection.com/bd/serie/Laufeust
de Troy/Thanos l incongru</uri></binding>
      <binding name="p"><uri>http://www.collection.com/personne/Tarquin</
uri></binding>
    </result>
    <result>
      <binding name="t"><uri>http://www.collection.com/bd/Laufeust de
Troy/L ivoire du Magohamoth</uri></binding>
      <binding name="p"><uri>http://www.collection.com/personne/Tarquin</
uri></binding>
    </result>
  </results>
</sparql>
```

Exemple avec une variable sur un prédicat

```
PREFIX bd: <http://www.collection.com/bd#>

SELECT * WHERE {
   <http://www.collection.com/bd/Lanfeust de
Troy> ?p ?v .
}
```

Nœuds anonymes

- Nœud non étiqueté
  - Dans le graphe requêté
- Dont l'étiquette n'est pas importante
  - Dans le patterns
- En TURTLE: []
- Les ?t ayant un dessinateur:

```
PREFIX bd: <http://www.collection.com/bd#>

SELECT * WHERE {
  ?t bd:dessinateur []
}
```

Patterns plus complexes: et / ou

- Opérateur ET implicite

```
PREFIX bd: <http://www.collection.com/bd#>

SELECT * WHERE {
  ?t bd:titre ?ti .
  ?t bd:dessinateur [] .
}
```

- Opérateur OU: UNION

```
PREFIX bd: <http://www.collection.com/bd#>

SELECT * WHERE {
  { ?t bd:scenariste ?p .}
UNION
  { ?t bd:dessinateur ?p .}
}
```

Filtres

- Complémentaire au WHERE
  - pas directement du matching
  - !, &&, ||, =, !=, >, +, -, etc

```
PREFIX bd: <http://www.collection.com/bd#>
PREFIX xsd: <http://www.w3.org/2001/
XMLSchema#>

SELECT * WHERE {
    ?t bd:numero ?n .
    FILTER(?n > "1"^^xsd:integer).
}
```

Requêter plusieurs graphes

- Combiner les information provenant de plusieurs graphes
- FROM permet de spécifier le graphe requêté
  - Simple IRI
  - NAMED: permet de faire référence dans le WHERE au graphe indiqué
  - GRAPH permet de spécifier un pattern à chercher dans un autre graphe
    - Le graphe peut être identifié par une variable

Exemple

```
PREFIX bd: <http://www.collection.com/bd#>

SELECT *
FROM <http://www.collection.com/bd#g1>
FROM NAMED bd:g2
WHERE {
  ?t bd:titre ?ti .
  GRAPH bd:g2 {?t bd:titre ?ti2 .}
}
```

Projections et assimilés

- Fonctionnement du SELECT similaire à SQL
  - Possibilités de renommage
  - Calculs

```
PREFIX bd: <http://www.collection.com/bd#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>

SELECT (fn:concat(?ti," par ",?scn," et ",?den) as ?album) WHERE {
    ?t bd:titre ?ti .
    ?t bd:scenariste ?sc .
    ?sc bd:nom ?scn .
    ?t bd:dessinateur ?de .
    ?de bd:nom ?den .
}
```

Aggrégation

- GROUP BY + fonctions d'aggrégation
  - COUNT, SUM, MAX, …

```
PREFIX bd: <http://www.collection.com/bd#>

SELECT ?serie (COUNT(?to) as ?nbTomes) WHERE {
    ?serie bd:tome ?to .
}
GROUP BY ?serie
```

Négation

- Via NOT EXIST dans FILTER
  - Autres méthodes
    - MINUS (peu/pas implémenté)
    - !bound dans FILTER(SPARQL 1.0)

```
PREFIX bd: <http://www.collection.com/bd#>

SELECT * WHERE {
     ?serie bd:tome ?to .
     OPTIONAL {?serie bd:editor ?ed}
     FILTER(!bound(?ed)).
     FILTER(NOT EXISTS {?serie bd:editor []}).
}
```

Créer des graphes résultats

- CONSTRUCT à la place de SELECT

```
PREFIX bd: <http://www.collection.com/bd#>

CONSTRUCT {
?serie bd:album ?ti .
?serie bd:numero ?n . }
WHERE {
    ?serie bd:tome ?to .
    ?to bd:numero ?n .
    ?to bd:titre ?ti .
}
```

Autres mots clés

- DISTINCT
  - Comme en SQL
- LIMIT, OFFSET
  - Nb réponses, quelles réponses
- OPTIONAL
  - Le matching est optionnel pour le pattern spécifié
- ASK
  - true/false

Mise à jour

- INSERT DATA { triples }
- DELETE DATA { triples }
- [DELETE { template }] [INSERT { template }] WHERE { pattern }
- LOAD uri [INTO GRAPH uri]
- CLEAR GRAPH uri
- CREATE GRAPH uri
- DROP GRAPH uri

RDFS, OWL etc

- Règles/axiomes logiques permettant:
  - De déduire des triplets additionnels
  - D'ajouter des contraintes d'intégrité
    - Seulement sur les types de données en RDFS

- Exemple: tous les tomes de série sont des livres

Schémas

- RDF Schema
  - Classification des ressources
  - Contraintes d'intégrité simples
    - Sur les types primitifs
- OWL
  - Logique plus riche
  - Contraintes d'intégrité plus complexes

RDF Schema (RDFS)

- Système de classes de ressources
  - Avec système de sous-classes
- Description des prédicats
  - Quel sujet, quel objet ?
- Interprétation spécifiques → système d'inférence
  - Déduction de nouveaux faits
    - Comment les prendre en compte ?

rdfs: http://www.w3.org/2000/01/rdf-schema#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

Vocabulaire RDFS

- Typage des propriétés:
  - rdfs:domain rdfs:range
- Types de base
  - rdfs:Resource rdfs:Literal rdfs:Datatype rdfs:Class
- Relations entre types et propriétés
  - rdfs:subClassOf rdfs:subPropertyOf
- Conteneurs
  - rdfs:member rdfs:Container rdfs:ContainerMembershipProperty
- Divers
  - rdfs:comment rdfs:seeAlso rdfs:isDefinedBy rdfs:label

Interprétations RDFS

- Interprétations *conformes* à RDFS
- Fixe la manière d'interpréter les éléments du vocabulaire RDFS
- Implication RDFS:
  - E1 implique E2 si toute interprétation conforme à RDFS qui satisfait E1 satisfait aussi E2

Inférence

- Certains triplets peuvent être considérés comme existants *implicitement:* les ajouter ne change pas le fait d'être impliqué

  - ***Inférence***: les rendre explicites

  - Graphe ***saturé***: on a ajouté toutes les inférences possibles

Classes et instances

- Classe ↔ ensemble de ressources
- Ressource R ∈ classe C:
  - R rdf:type C
- Une ressource peut appartenir à plusieurs classes

Interprétation des types

- Classes: de type rdfs:Class
- Ensemble IC des (interprétations de) classes
- Sémantique ICEXT d'une classe C:
  - $ICEXT(C)=\{ x \mid (x,C) \in IEXT(I(rdf:type)) \}$

- Inférence:
  - sss rdf:type ooo $\Rightarrow$ ooo rdf:type rdfs:Class

Datatypes

- Types de données primitifs
- Sous classe de rdf:Literal
- Types standard de XMLSchema

Type de données des littéraux

- Les types de littéraux aaa sont de type rdfs:Datatype
  - I(aaa) ∈ ICEXT(I(rdfs:Datatype))
- Les types de littéraux aaa sont des sous-classes de rdfs:Literal
- Idéalement, les types des littéraux doivent être respectés par l'interprétation
  - Seule vérification de type de RDFS
    - Une date ne peut pas être un entier
  - Norme un peu plus subtile

rdfs:domain

- Domaine au sens domaine d'une fonction
- Fixe le type CCC des sujets d'un prédicat ppp
  - ppp rdfs:domain CCC
- Exercice: l'écrire sous forme d'interprétation RDF
- Inférence:

ppp rdfs:domain CCC $\wedge$ sss ppp ooo

$\Rightarrow$sss rdf:type CCC

rdfs:range

- Dual de rdfs:domain
- Fixe le type CCC des objets d'une propriété ppp:
  - ppp rdfs:range CCC

- Inférence:

$$\text{ppp rdfs:range CCC} \wedge \text{sss ppp ooo} \Rightarrow \text{ooo rdf:type CCC}$$

Transitivité

- RDFS impose la transitivité et la réflexivité pour certaines relations:
  - rdfs:subPropertyOf, rdfs:subClassOf
- Exemple d'inférence

ccc rdfs:subClassOf ddd

$\wedge$

ddd rdfs:subClassOf eee

$\Rightarrow$

ccc rdfs:subClassOf eee

Sous-classes

- Classe C $\subseteq$ classe D
  - C rdfs:subClassOf D

- Inférence:
  - CCC rdfs:subClassOf ddd $\wedge$ rrr rdf:type CCC $\Rightarrow$ rrr rdf:type DDD

Sous-prédicats

- Correspond à un raffinement d'un prédicat
  - ppp rdfs:subPropertyOf qqq

- Inférence

ppp rdf:subPropertyOf qqq

$\wedge$ sss ppp ooo

$\Rightarrow$ sss qqq ooo

Axiomes et règles d ’inférence

http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/#rdfs-interpretations

Inférence et SPARQL

- Evaluation de requêtes en présence d'inférence:
  - Compléter le graphe RDF avant de répondre
    - Souvent trop coûteux en termes de stockage
  - ou réécrire la requête

Réécriture de requête SPARQL

Si bd:tome rdfs:subPropertyOf bd:album

```
SELECT ?l WHERE {
?l bd:album bd:serie/Lanfeust de troy. }
```

devient

```
SELECT ?l WHERE {
?l bd:album bd:serie/Lanfeust de troy.
UNION
?l bd:tome bd:serie/Lanfeust de troy. }
```

Références

- http://www.w3schools.com/rdf/rdf_intro.asp
- http://www.cambridgesemantics.com/2008/09/sparql-by-example/
- http://www.w3.org/TR/rdf-schema/
- http://www.w3.org/TR/rdf-mt
- http://www.foaf-project.org/