Graphes de données

RDF & SPARQL

Emmanuel Coquery

RDF, SPARQL et le Web sémantique

- RDF
  - format de graphe annoté par des URI
- 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

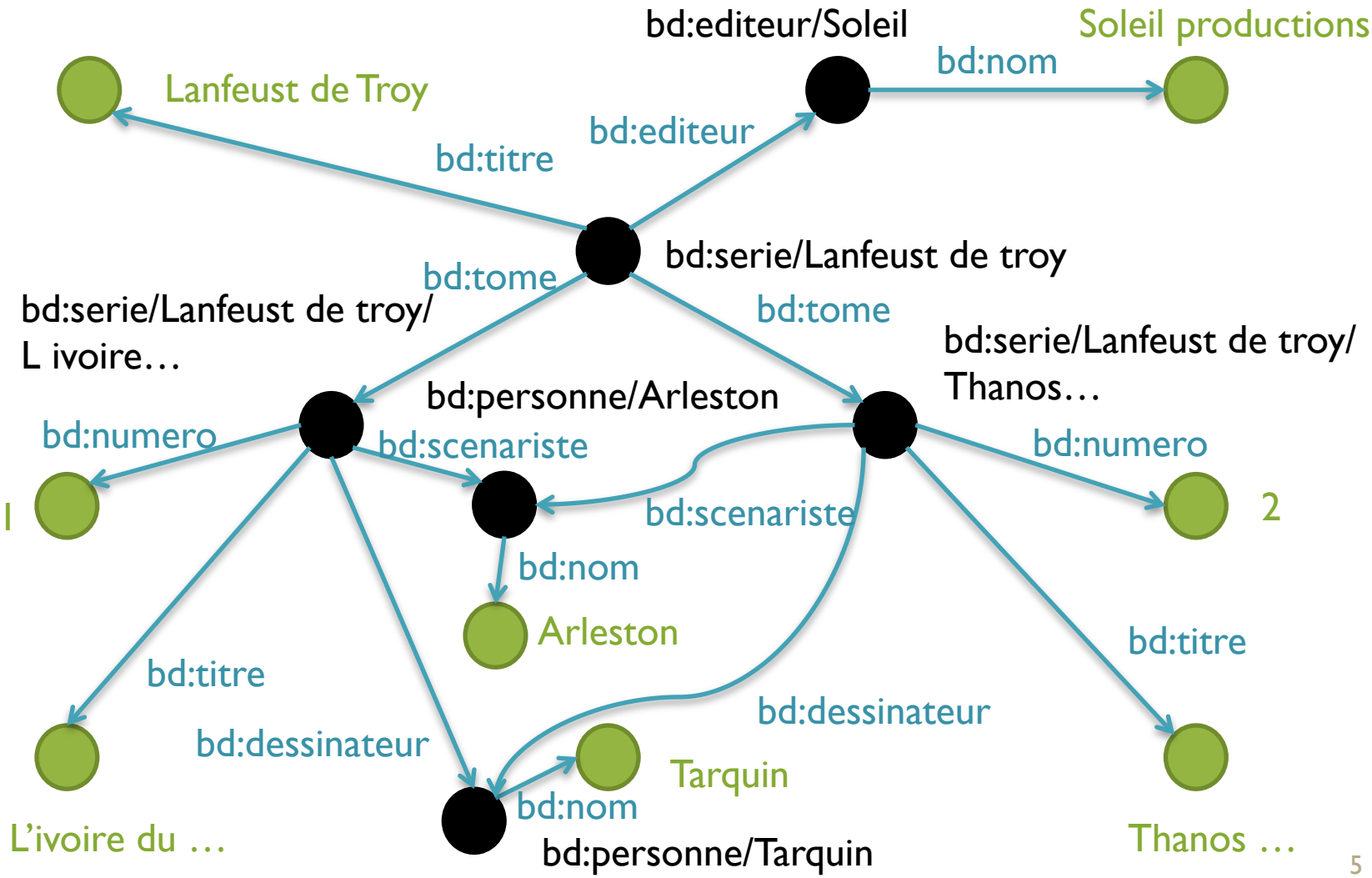
- Graphes orientés
  - Les sommets et les arêtes sont étiquetés
- Constitue un modèle de données alternatifs aux modèles:
  - Relationnel, semi-structuré, objet
- Permet de représenter aisément des liens entre des *choses* référencées par un identifiant:
  - Sommet : *chose*
  - Arête : relation entre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

RDF: graphes pour le Web sémantique

- Standard W3C
- Graphes RDF:
  - Étiquetés (arêtes et sommets) par des URIs
    - 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'URI est symbolique
    - Dans le cadre du « Linked data », on attend qu'un certain nombre d'URI 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
    - 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'URI indiquée par l'attribut rdf:about
  - Attributs/élements:
    - Espaces de nommage + nom local = URI du prédicat
  - Valeur/attribut rdf:resource
    - Objet
    - Littéral/URI
  - 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
- URI:
  - <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

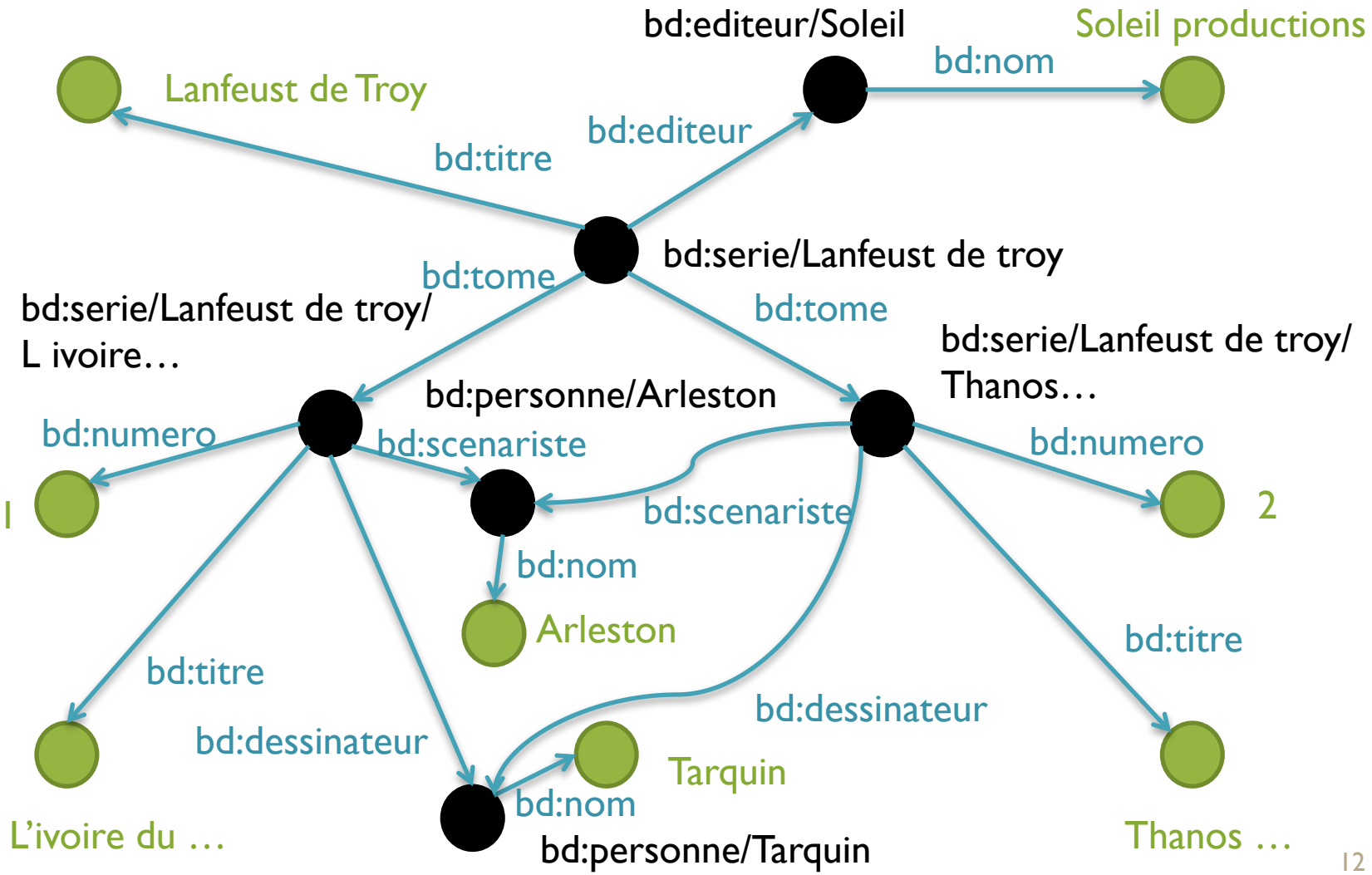
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bd: <http://www.collection.com/bd#>

<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"> .
```

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
- NP-Complet dans le cas général

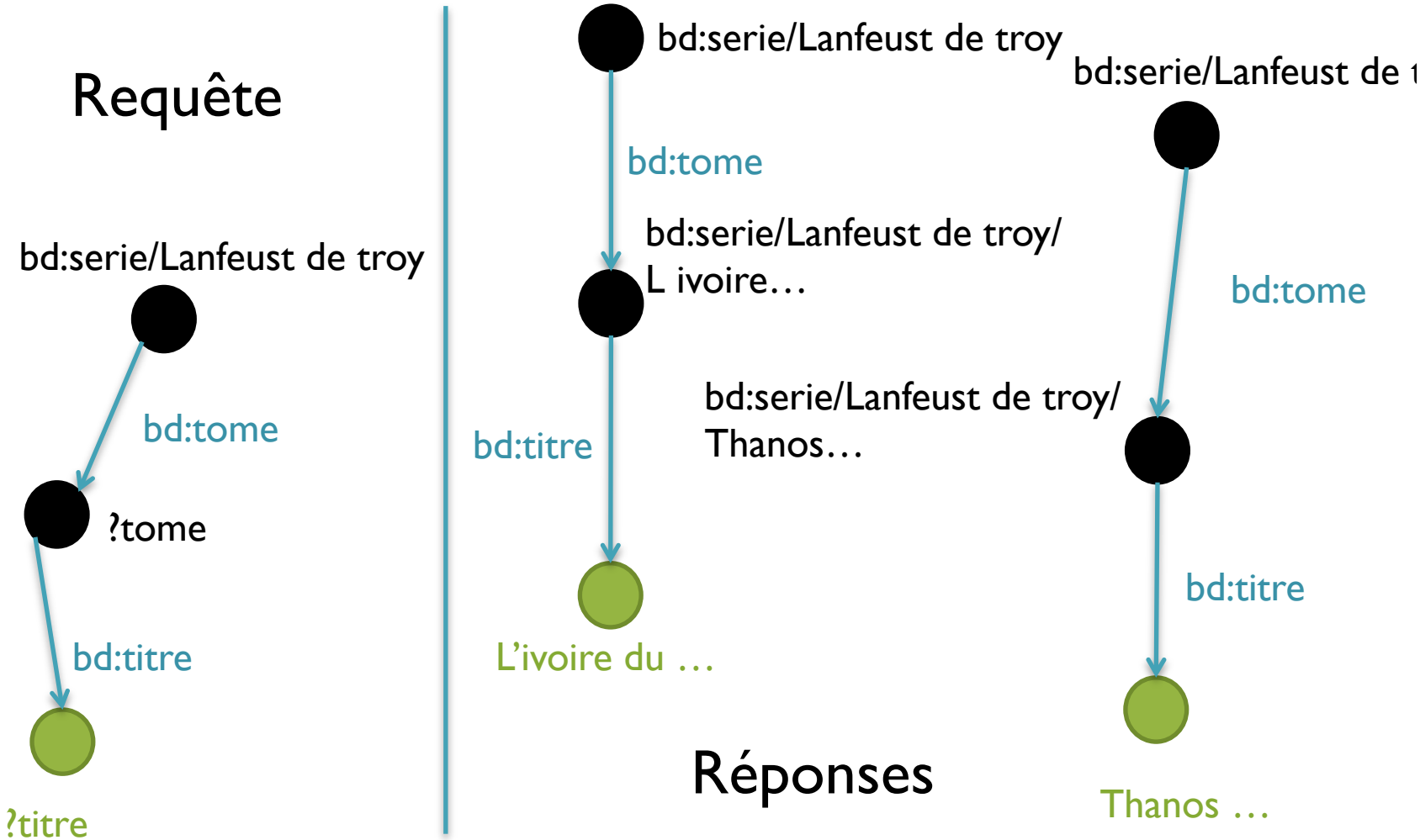
RDF: Exemple

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 URI/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 URI
  - 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

Références

- http://www.w3schools.com/rdf/rdf_intro.asp
- http://www.cambridgesemantics.com/2008/09/sparql-by-example/