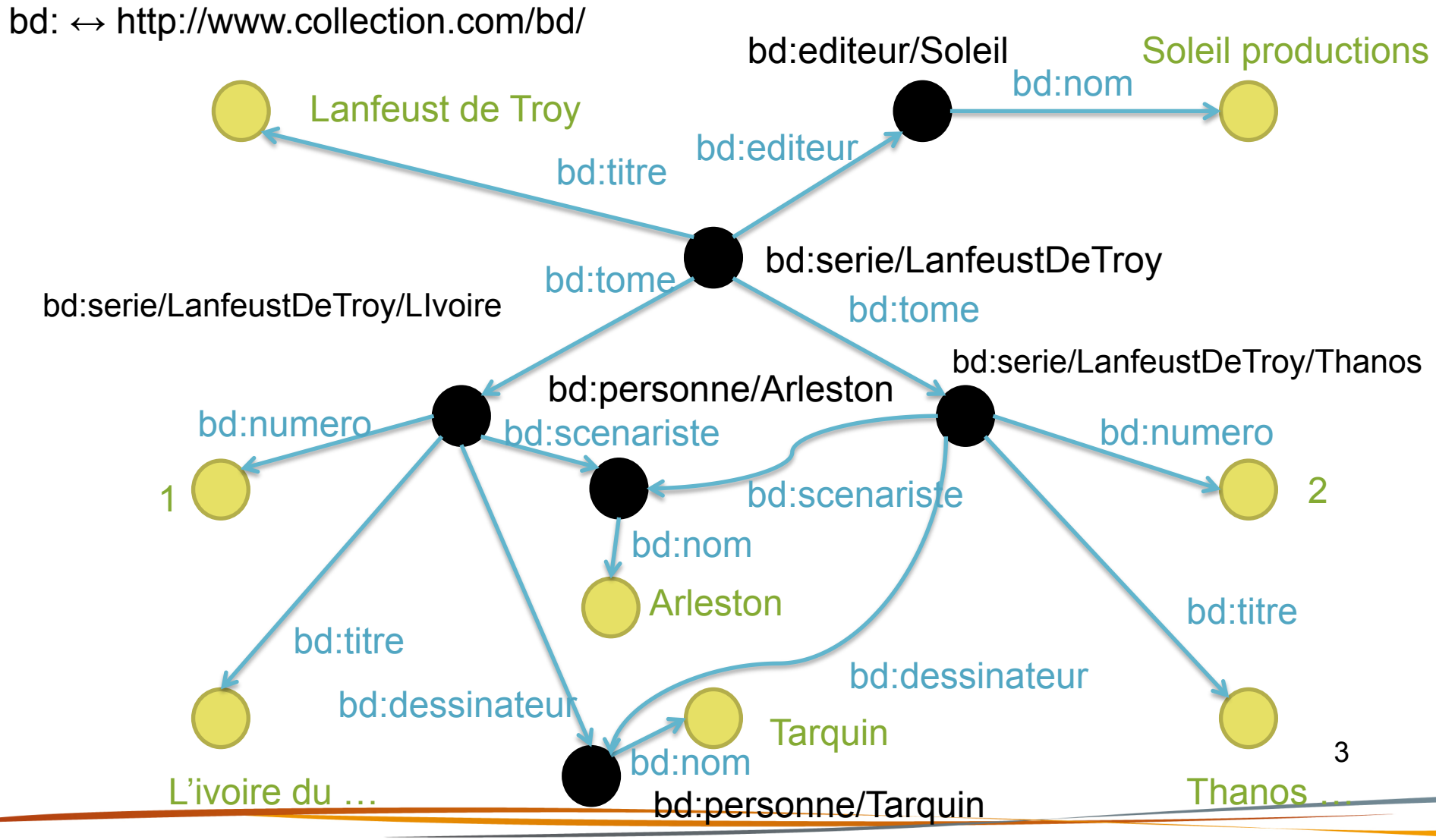
SPARQL

Requêtes de graphes RDF

Requête: matching de sous-graphe

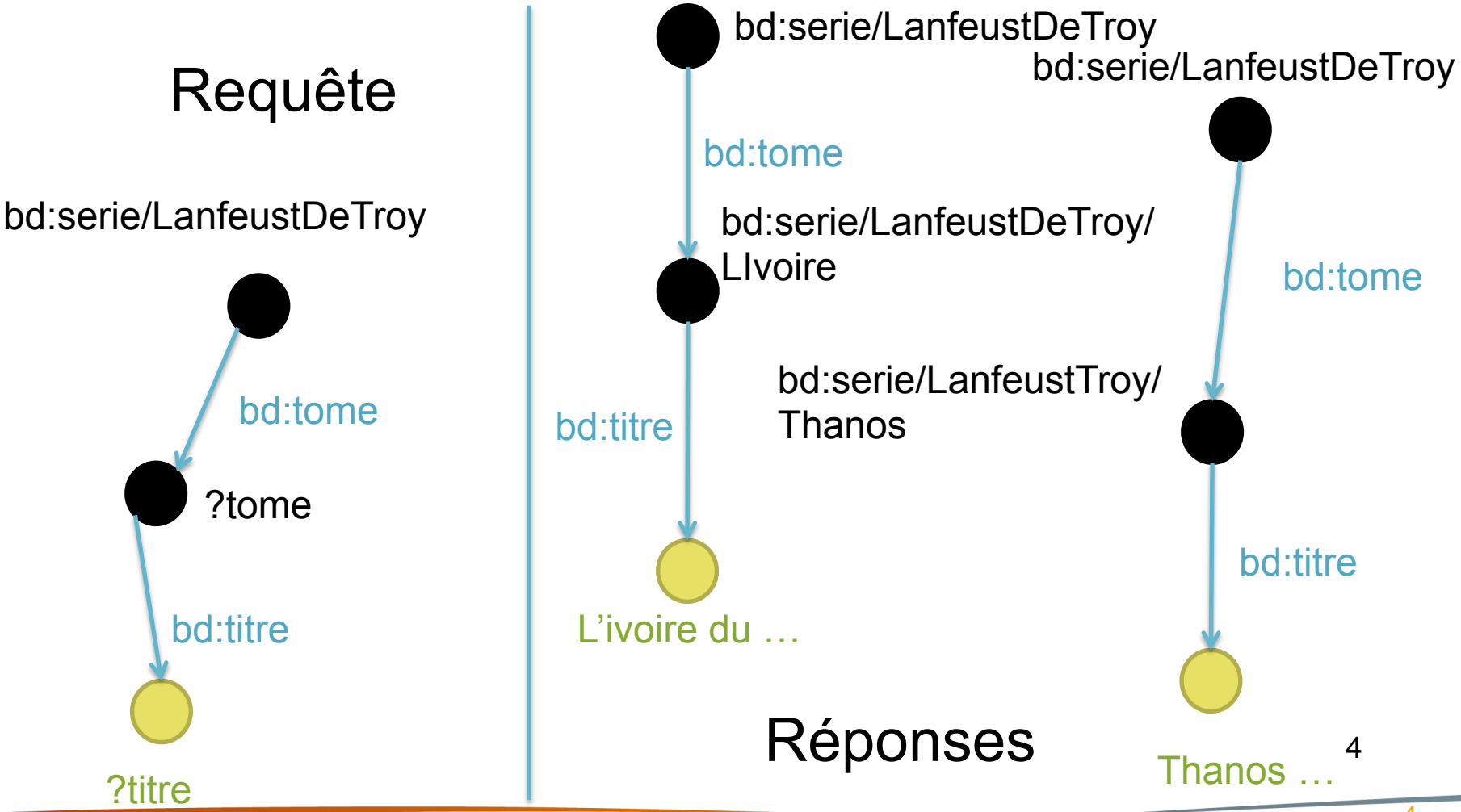
- Spécifier des contraintes sur les parties du graphe à récupérer
  - « pattern matching » pour les graphes
- Pattern =
  - Un graphe exemple
  - Avec des variables
- Réponses
  - Sous-graphes du graphe de départ
  - Correspondant au pattern
    - En instanciant les variables
- Une réponse explique comment 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/
bd:editeur/Soleil
Soleil productions
bd:nom
Lanfeust de Troy
bd:editeur
bd:titre
bd:serie/LanfeustDeTroy
bd:tome
bd:tome
bd:serie/LanfeustDeTroy/LIvoire
bd:serie/LanfeustDeTroy/Thanos
bd:personne/Arleston
bd:numero
bd:scenariste
bd:numero
1
bd:scenariste
2
bd:nom
Arleston
bd:titre
bd:titre
bd:dessinateur
bd:dessinateur
Tarquin
bd:nom
L'ivoire du …
bd:personne/Tarquin
Thanos …

Exemple

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



4

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/LaufeustDeTroy/Thanos</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/LaufeustDeTroy/LIvoire</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