

WEB DES DONNÉES

1. *Web des données et vers un Web de données liées*
2. *Modèle de données RDF*
3. *Schémas RDFS*
4. *Le langage de requête SPARQL*

Interrogation de graphes

- Plusieurs langages d'interrogation de données RDF:
 - *SPARQL*, *SquishQL*, *RDQL*, et *TriQL*(requêtage sans schéma/ontologie)
 - *RQL*, *SeREQ*, et *eRQL* (requêtage du schéma ou des données, plus expressifs)
 - Des langages basés sur *Xquery* (après sérialisation des données RDF en *RDF/XML*)
 - ...

Objectif

- Vous donner des bases pour écrire des requêtes SPARQL.
- Lire du Turtle (très proche de SPARQL).
- Ce n'est qu'une introduction ; pour en savoir plus : <http://www.w3.org/TR/sparql11-overview/>

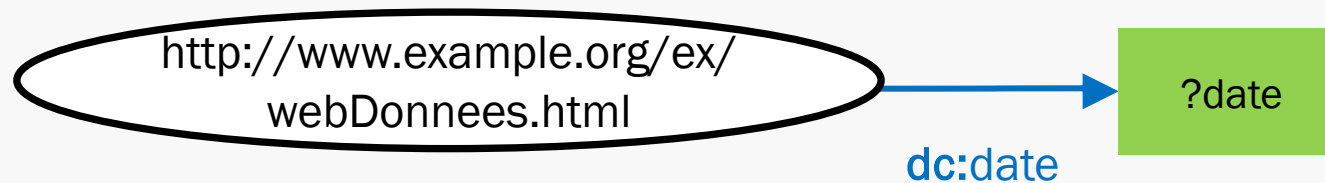
SPARQL

SPARQL (Simple Protocol and RDF Query Language):

- SPARQL fournit le langage d'interrogation du web sémantique
- SPARQL est à RDF ce que SQL est aux bases d données relationnelles.
- Recommandation W3C (version 1.0 en 2008, version 1.1 mars 2013)
- Syntaxe basée sur Turtle
- → SPARQL est un langage d'interrogation de graphes RDF dont l'énoncé de base est lui aussi le triplet (ressource, propriété, valeur)
- Syntaxe inspirée de SQL (SELECT, WHERE, GROUP BY)

Exemple de RDF et d'un motif

- **Requête:** la date de création du cours Web des données



```
PREFIX dc:<http://purl.org/dc/elements/1.1/>
SELECT ?date
WHERE {<http://www.example.org/ex/webDonnees.html> dc:date ?date.
}
```

- **Résultat de la requête:**

Date
2023

SPARQL: Syntaxe

- Syntaxe Turtle avec des points d'interrogation pour les variables:

`?x rdf:type ex:Person`

- Décrire des patrons de graphes à trouver:

`SELECT ?subject ?property ?value`

`WHERE { ?subject ?property ?value }`

- Un patron est par défaut une conjonction de triplets:

`SELECT ?x WHERE`

`{ ?x rdf:type ex:Person .`

`?x ex:name ?name . }`

SPARQL: Syntaxe

```
PREFIX p1:<uri1> } Définition des préfixes (IRI relatifs)
PREFIX p2:<uri2> }
SELECT ?var1 ... ?varn Définition des résultats en sortie
FROM <...> } Définition des jeux de données sur lesquelles porte la
FROM NAMED <...> } requête (SPARQL 1.1)
WHERE {triplet1 . } Définition des conditions
      ...
      tripletk . }

#Modificateurs de résultats (facultatif)
GROUP BY ... #SPARQL 1.1 }
HAVING ... #SPARQL 1.1 }
ORDER BY ... }
LIMIT ... }
OFFSET ... }
}
```

SPARQL: Syntaxe

- **Requête:** la date de création du cours Web des données

```
PREFIX dc:<http://purl.org/dc/elements/1.1/>  
SELECT ?date  
WHERE {<http://www.example.org/ex/webDonnees.html> dc:date ?date.  
}
```

date

2023

- **Requête:** le type de Web des données

```
PREFIX dc:<http://purl.org/dc/elements/1.1/>  
SELECT ?type  
WHERE {<http://www.example.org/ex/webDonnees.html> rdf:type  
?type  
}
```

http://www.example.org/
ex/webDonnees.html

rdf:type

Text

SELECT

- La clause SELECT permet de définir la forme de résultat. Ce résultat est un ensemble de n-uplets.

```
PREFIX p1:<uri1>  
PREFIX p2:<uri2>  
SELECT ?var1 ... ?varn
```

- Identifie les variables (var1,...,varn) à faire apparaître dans la réponse de la requête.
- *toutes variables commencent par le caractère ‘?’ ou \$ et ce caractère ne fait pas partie du nom de la variable*

Les variables

- *Une variable dans une requête commence par le caractère "?" ou \$ et ce caractère ne fait pas partie du nom de la variable. Le nom d'une variable :*
 - *ne doit pas commencer par un chiffre*
 - *est sensible à la casse*
 - *ne doit pas contenir d'espace*
 - *doit être signifiant, car il sert de nom de colonne dans le résultat*
- SPARQL 1.1 introduit le mot clé AS pour changer le nom des colonnes comme on peut le faire en SQL.

Exemples

- **Select ?type ?creator**

permettra de conserver les couples (type, creator) des valuations des variables ?type et ?creator calculées par les requêtes

- **Select ***

permet de conserver toutes les variables qui apparaissent dans la requête

- **Select distinct ?type ?creator**

enlèvera les doublons parmi les couples (type, creator)

Clause WHERE

- La clause WHERE définit des conditions qui devront être respectées (ce qu'on cherche dans le graphe cible)

```
PREFIX p1:<uri1>  
PREFIX p2:<uri2>  
SELECT ?var1 ... ?varn  
WHERE {triplet1 .  
      ...  
      tripletk  
      }
```

- La plupart des requêtes SPARQL contiennent un ensemble de triplets : patron de graphe.
- Chaque sujet, prédicat et objet peut être une variable
- Le “.” à la fin permet la conjonction de conditions, c’est le “AND” dans la requête.

WHERE: Motifs de triplets

- Utilisation des triplets pour définir un motif pour la clause Where (comme en RDF)
 - *On parle de motif de triplet car le triplet peut contenir des variables.*
- Les termes utilisés dans les motifs de triplets sont:
 - *Des IRIs (Internationalized Ressource Identifiers), généralisation des URIs*
 - *Des littéraux. En général, ce sont des chaînes de caractères entre simples quotes ou guillemets.*
 - *Des variables, préfixées par ? ou par \$*
 - *Des nœuds blancs qui jouent le rôle de variables non distinguées.*

Clause WHERE

- Expliquer la requête

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

```
SELECT ?film
```

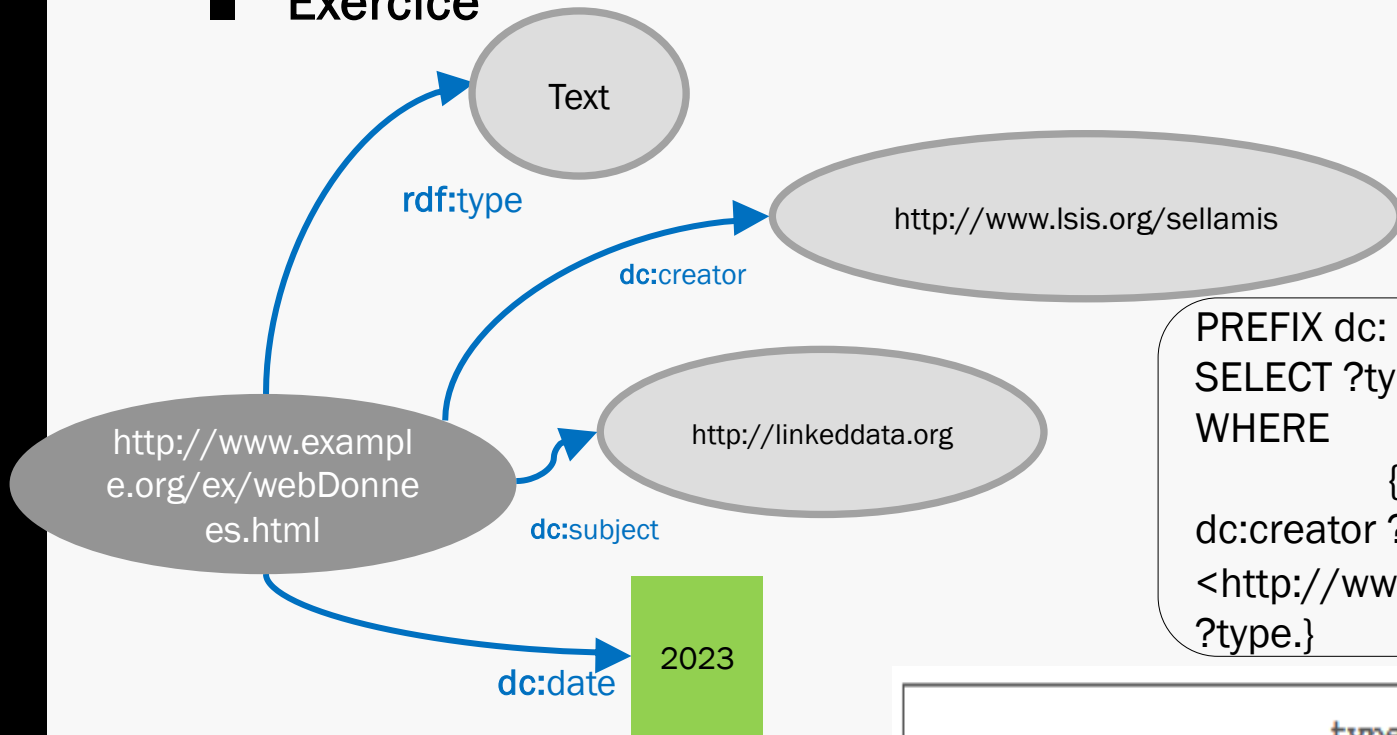
```
WHERE {
```

```
  ?film a <http://dbpedia.org/ontology/Film> }
```

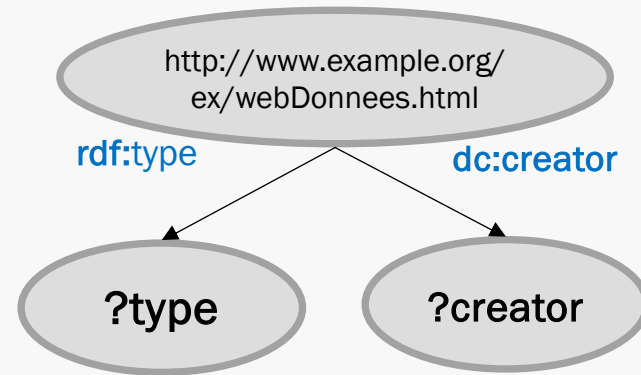
- Rechercher toutes les ressources ?film qui ont pour type (a) `<http://dbpedia.org/ontology/Film>`
- Le mot clé « a » représente l'abréviation du prédicat <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
- « a » signifie a pour type
- La requête (SELECT) retourne les valeurs trouvées pour chaque appariement de graphe.

SPARQL

■ Exercice



■ Donner la requête SPARQL



```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?type ?creator
WHERE
    {<http://www.example.org/ex/webDonnees.html>
    dc:creator ?creator.
    <http://www.example.org/ex/webDonnees.html> rdf:type
    ?type.}
```

type	creator
http://www.example.org/ex/webDonnees.htm/Text	http://www.lsis.org/sellamis

Clause WHERE

- La clause WHERE est sous la forme d'un groupe de motifs de graphe:
- Un graphe peut être
 - *Un motif de graphe basique = ensemble de motifs de triplets entre { }*

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?type ?creator
WHERE
    { ?x dc:creator ?creator.
      ?x rdf:type ?type }
```

- *Un groupe de groupes de motifs entre { }*

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?type ?creator
WHERE {
    { ?x dc:creator ?creator. }
    { ?x rdf:type ?type. }
}
```

Décomposition du groupe en 2 sous-groupes
motifs contenant chacun 1 motif triplet

Motifs de triplets

Exemple:

- Triplets ayant un sujet commun :

```
SELECT ?name ?fname
WHERE{?x rdf:type ex:Person.
      ?x ex:name ?name .
      ?x ex:firstname ?fname .
      ?x ex:author ?y . }
```



```
SELECT ?name ?fname
WHERE {?x a ex:Person;
      ex:name ?name ;
      ex:firstname ?fname ;
      ex:author ?y . }
```

- plusieurs valeurs

```
?x ex:firstname "Fabien".
?x ex:firstname "Lucien".
```

```
?x ex:firstname "Fabien", "Lucien".
```

SPARQL

Exercice

- Requête à écrire en SPARQL: la ressource qui a pour sujet <http://linkeddata.org/> et créée en 2023

```
PREFIX dc: http://purl.org/dc/elements/1.1/
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?x
WHERE
    { ?x dc:subject <http://linkeddata.org/>;
      dc:date "2023"^^xsd:year. }
```

IRIs et espaces de noms

- Comme pour les données RDF, la requête peut contenir une (ou plusieurs) définition de préfixe associé à un espace de noms.
- Les requêtes suivantes sont équivalentes:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>  
SELECT ?title  
WHERE { <http://example.org/book/book1> dc:title ?title }
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>  
PREFIX : <http://example.org/book/>  
SELECT $title  
WHERE { :book1 dc:title $title }
```

Filtres

```
PREFIX p1:<uri1>
PREFIX p2: <uri2>
SELECT ?var1 ... ?varn
WHERE {triplet1 .
      ...
      tripletk .
      [FILTER]
}
```

- déclarer des contraintes supplémentaires notamment sur les variables
- SELECT= clause sélectionnant les valeurs à retourner
- WHERE= patron de graphe à apparier
- **FILTER** = contraintes ajoutées dans la clause WHERE exprimées avec des fonctions de tests XPath 2.0 ou externes

Filtres

- Définition de contraintes sur les valeurs obtenues: Clause FILTER

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE
    { ?x foaf:knows ?y.
      ?x foaf:age ?age.
      filter (?age <=65 && ?age>=18)
    }
```

- Filter est insérée **dans la clause Where**
- Les filtres permettent de faire des calculs arithmétiques, des comparaisons, de combiner des expressions avec des connecteurs booléens (and, or, not), d'appeler des fonctions prédéfinies et des fonctions d'extension définies par l'application.
- On peut définir plusieurs FILTER
- Lié à FILTER :
 - **BOUND()** teste si une variable est liée à une valeur
 - **REGEX()** pour test d'expression régulière

Opérateurs et fonctions

- Un filtre s'applique à un motif de graphe
- **Exemple:** les titres qui commencent par "S"

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { ?x dc:title ?title.
       FILTER regex(?title, "^S")
}
```

Query Result:

title
"SPARQL Tutorial"

<http://www.example.org/ex/webDonnees.html>

dc:title

SPARQL Tutorial

Opérateurs et fonctions

■ Exemple: Les ressources créées en 2021

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?x ?y
WHERE { ?x dc:date ?y.

    filter (?y="2021"^^xsd:year)

}
```

x	y
<http://www.example.org/ex/webDonnees.html>	2021

Attention, les littéraux doivent être typés

Pour les différents opérateurs et fonctions consulter la documentation SPARQL:

<https://www.w3.org/TR/rdf-sparql-query/>

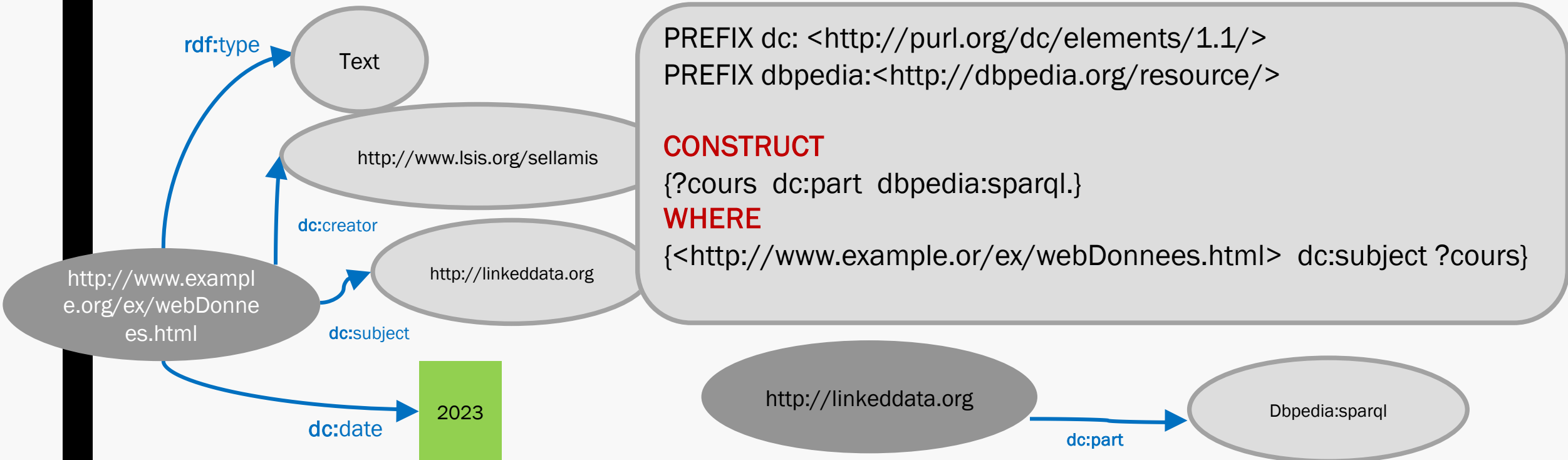
Opérateurs et fonctions

- Une condition est une combinaison (&&, ||, !) d'expressions (=, !=, <, >, etc.)
- Expressions régulières : `regex(string, pattern)`
- Tests sur les valeurs des variables : `isURI(?x)`, `isBlank(?x)`, `isLiteral(?x)`, etc.
- Et bien d'autres fonctions sur les chaînes et les littéraux, sur les dates, etc...

Autres formes d'interrogation

CONSTRUCT

Retourne un graphe RDF au lieu d'un ensemble de valeurs de variables.



subject	predicate	object
http://linkeddata.org	dc:part	http://dbpedia.org/resource/sparql

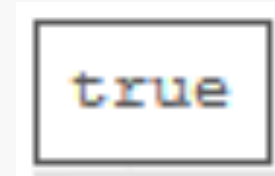
Autres formes d'interrogation

ASK

Retourne un booléen indiquant si un motif d'interrogation correspond ou non.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
```

```
ASK { ?x dc:title "SPARQL Tutorial" }
```



DESCRIBE

Retourne un graphe RDF décrivant les ressources trouvées.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
```

```
DESCRIBE ?UE
```

```
WHERE
```

```
{?UE dc:subject <http://linkeddata.org>}
```

Trier, filtrer et limiter les réponses

- Exemple:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?familyName  
WHERE { ?x foaf:familyName ?familyName . }
```

- Trier par nom les réponses

```
SELECT ?familyName  
WHERE { ?x foaf:familyName ?familyName . }  
ORDER BY ?familyName
```

ORDER BY

- ORDER BY permet d'ordonner le résultat de la requête. Il suffit de lui donner en argument la colonne selon laquelle il doit ordonner les résultats.

```
SELECT ?familyName  
WHERE { ?x foaf:familyName ?familyName . }  
ORDER BY ?familyName
```

- Possibilité de spécifier plusieurs colonnes pour l'ordonnancement du résultat

```
SELECT ?familyName ?givenName  
WHERE  
    { ?x foaf:familyName ?familyName ;  
      foaf:givenName ?givenName . }  
ORDER BY ?familyName ?givenName
```

```
SELECT ?familyName ?givenName  
WHERE  
    { ?x foaf:familyName ?familyName ;  
      foaf:givenName ?givenName . }  
ORDER BY DESC(?familyName) ?givenName
```

- Trier une colonne par ordre inverse: ajouter le mot-clef DESC devant le nom de la colonne concernée entre parenthèses

ORDER BY

Que pensez vous de la requête suivante?

```
SELECT ?familyName  
WHERE {  
  ?x foaf:familyName ?familyName ;  
      foaf:givenName ?givenName . }  
ORDER BY ?familyName ?givenName
```

La requête suivante n'est pas conforme, étant donné que la colonne "givenName" ne fait pas partie des colonnes rapatriées (dans la clause SELECT).

Trier, filtrer et limiter les réponses

- Trier par nom les réponses de la n°21 à la n°40

Quelle est la clause qui limite le nombre de réponses?

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?familyName  
WHERE { ?x foaf:familyName ?familyName . }  
ORDER BY ?familyName  
ORDER BY ?familyName  
LIMIT 20  
OFFSET 20
```

LIMIT permet de limiter le nombre de lignes retournées.

OFFSET permet de n'afficher le résultat qu'à partir de la ligne indiquée.

Agrégation des résultats

■ Clause GROUP BY et HAVING :

```
PREFIX  
PREFIX  
....  
SELECT ?x  
WHERE {?x ?y ?z.}  
GROUP BY ?x  
HAVING condition
```

- *GROUP BY: Une ou plusieurs variables de regroupement*
- *HAVING: Condition sur les groupes*
- *Agréger les valeurs: Fonctions d'agrégation (count, max, avg, etc.)*

Agrégation des résultats

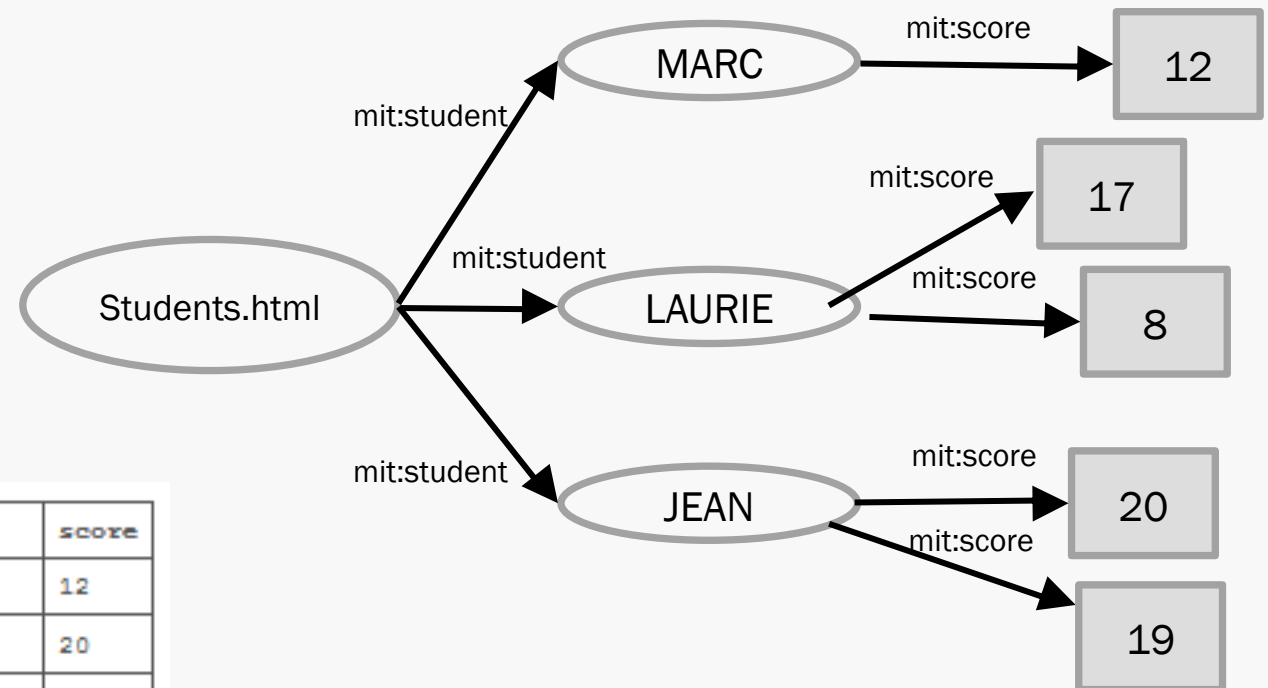
■ Exemple

Déterminer les étudiants qui ont un score maximal supérieur à 10

```
PREFIX mit: <http://www.mit.edu/>  
PREFIX xsd:  
<http://www.w3.org/2001/XMLSchema#>
```

```
SELECT ?student (MAX(?score) as ?score)  
WHERE {?student mit:score ?score.}  
GROUP BY ?student  
HAVING (MAX(?score)>=10)
```

student	score
http://www.example.org/ex/webDonnees.html/student.html/MARC	12
http://www.example.org/ex/webDonnees.html/student.html/JEAN	20
http://www.example.org/ex/webDonnees.html/student.html/LAURIE	17



Agrégation des résultats

- Exemple: Déterminer le nombre d'étudiants

```
PREFIX mit: <http://www.mit.edu/>
```

```
SELECT (count(?n) as ?nbreEtudiants)  
WHERE {?student mit:student ?n.}
```

Outils utilisant SPARQL

- **Endpoints** SPARQL (jeux de données fournis) : éditeurs
- DBpedia et Wikidata, YASQUI, SPARQLer, Twinkle, etc.
- **Librairies** ou **frameworks** : RDFlib (Python), Apache Jena, Apache Marmotta, Apache Jena Fuseki, etc.
- **SGBD** : AllegroGraph, BlazeGraph, BrightstarDB, Dydra, Stardog, etc.

<http://dbpedia.org/sparql>
<http://query.wikidata.org/>
<http://legacy.yasgui.org/>
<http://rdflib.readthedocs.io/en/latest/>
<http://jena.apache.org/>
<http://marmotta.apache.org/>
http://en.wikipedia.org/wiki/List_of_SPARQL_implementations

Ce que vous n'avez pas vu

- Rendre certains patterns (triplets) optionnels (OPTIONAL)
 - Pas de correspondance pour un pattern (NOT EXISTS)
 - Sous-requêtes imbriquées (avec des opérateurs ensemblistes)
 - Requêtes de création, suppression, mise à jour (create graph, insert data, etc.)
 - Interrogation de plusieurs graphes
-
- → Visiter le site W3C (même si c'est en anglais): <http://www.w3.org/TR/sparql11-query/> et <http://www.w3.org/TR/sparql11-update/>

Synthèse

- Des données sous forme de triplets qui forment des graphes orientés et étiquetés
- Plusieurs sérialisations de RDF
- Un schéma RDFS pour classifier les ressources et spécifier des contraintes d'intégrité simples
- Langage de requêtes SPARQL (syntaxe similaire à SQL)

Références

■ Sites Web:

- *Recommandation SPARQL*: <https://www.w3.org/TR/sparql11-query/Supports de cours et la version française:>
<http://www.yoyodesign.org/doc/w3c/rdf-sparql-query/>
- *Support de cours de Anne Cécile Caron Université de Lille*
- <https://www.w3.org/2009/Talks/0615-qbe/>
- *Support de cours de Michel Gagnon*: <https://moodle.polymtl.ca/course/view.php?name=INF8410>
- <https://www.w3.org/2001/12/semweb-fin/w3csw>

■ Livres

- *Fabien Gandon, Catherine Faron-Zucker, Olivier Corby. Le Web sémantique. Comment lier les données et les schémas sur le Web?*
- *Bob DuCharme. Learning SPARQL : Querying and Updating with SPARQL 1.1*

■ MOOC

- *MOOC Web sémantique et Web de données*: <https://www.fun-mooc.fr/courses/course-v1:inria+41002+session04/about>