https://www.transition-bibliographique.fr/2018-04-09-petit-laboratoire-semantique-experimentation-bm-fresnes/

Le petit laboratoire sémantique : expérimentation à la BM de Fresnes

Dans une démarche d'expérimentation, le petit laboratoire sémantique, développé en interne à la bibliothèque municipale de Fresnes, vise à explorer l'exploitation et l'interfaçage de données sémantiques construites selon la modélisation RDF dans un contexte FRBRisé. Les entités auteurs et œuvres sont représentées et contextualisées sous forme de graphe avec les relations qui les lient, en ayant l'auteur comme point d'entrée. Cette présentation fait suite à un premier article traitant de l'enrichissement du portail de la BM de Fresnes.

Le but de cet outil est d'explorer la logique entité-relation et l'utilisation exclusive des identifiants ARK comme point de départ. À court terme, il est envisagé d'exposer un lien depuis une autorité sur le catalogue en ligne de la bibliothèque, pour permettre à l'utilisateur d'accéder à une interface de découverte et d'exploration, présentant un auteur et une partie de ses œuvres contextualisées, en adéquation avec la 5e tâche utilisateur préconisée par le modèle FRBR. À plus long terme, l'idée est d'étendre l'applicatif au niveau des manifestations et de gérer les liens avec les notices du catalogue (disponibilité, cote/emplacement). Cette deuxième étape impliquera de connecter l'ensemble aux données locales du catalogue de la bibliothèque et donc d'interagir avec le SIGB (BGM de la société GMInvent).

© Bibliothèque municipale de Fresnes

Le point de départ a consisté à imaginer une interface pour laquelle un maximum d'information est disponible avec un minimum d'interaction de la part de l'utilisateur. Outre la taille et le code couleur qui permettent de facilement regrouper les entités par type ou par genre, un simple survol de la souris fournit une information textuelle ou une mise en évidence par l'intermédiaire d'une animation (dates d'édition / genres musicaux => œuvres reliées).

Récupération des métadonnées

À partir de l'identifiant ARK d'une entité type créateur, un flux de requêtes automatisé en SPARQL est lancé sur divers points de terminaisons pour récupérer les données et afficher les entités liées et alignées sous forme de graphe.

L'alignement principal exploité est celui avec DBpedia (skos:exactMatch), la contextualisation à l'aide des œuvres numérisées issues d'Europeana n'étant pas le résultat d'un alignement avec la BnF. Les données, issues des dumps « œuvres » et « auteurs » des jeux de données de data.bnf.fr au format RDF/XML (accessibles gratuitement sur api.bnf.fr), sont intégrées sur un triplestore instancié en local sur un serveur de la bibliothèque (graphdb qui s'appuie sur le framework RDF4J). L'interface web finale est développée pour le back-end en asp.net (c#) et pour le front-end en html/JavaScript (en s'appuyant sur la bibliothèque d3.js, idéale pour manipuler et interfacer des grands jeux de données). Un connecteur (back-end) va envoyer les requêtes en SPARQL sur le triplestore local et les points de terminaison de DBpedia et Europeana, et récupérer ces données sous forme de tables indexées. Après avoir été sérialisé en json, le flux de données est alors envoyé au front-end pour être exploité par la bibliothèque d3.js.

Une des fonctions importantes est la gestion des liens. Par exemple, concernant les œuvres agrégatives, le modèle de données de data.bnf.fr, qui est exhaustif en la matière, prévoit que le lien (dcterms:creator) demeure pour l'œuvre agrégée à partir du créateur. Nous avons dans ce cas pris le parti de ne pas représenter ce lien qui paraît redondant et alourdirait inutilement le graphe. Il demeure donc [créateur] -> a créé -> [œuvre] -> contient -> [œuvre agrégée]. Ainsi, on comprend que l'œuvre agrégée a été créée par l'entité de départ. Les liens de contribution (dcterms:contributor) sont gérés. Dans ce cas, une requête particulière va récupérer le créateur de l'œuvre concernée à partir duquel un nouveau graphe peut être généré.

Les libellés qui apparaissent sur les liens, en plus des couleurs différentes selon leur nature, ont leur sens de lecture en adéquation avec la sémantique. Ainsi une œuvre (a pour) créateur une entité auteur… Ces libellés disparaissent au bout d'un moment pour clarifier la lecture du graphe qui peut devenir surchargé. La bibliothèque d3.js va séparer les entités et leurs relations par un système d'indexation qu'il est impératif de respecter lors de la sérialisation. Toute représentation comporte sa propre modélisation.

Une contextualisation du discours

Nés la même année

L'interface fournit un échantillon de 10 auteurs maximum nés la même année que l'auteur exposé et ayant au moins une œuvre liée. Dans le cas où aucune date de naissance n'est renseignée pour l'auteur concerné, une liste de 10 auteurs pris aléatoirement est renvoyée par la requête, ceci afin de préserver une interface qui met l'accent sur la découverte et la sérendipité.

DBpedia

Par l'intermédiaire de DBpedia, et de son point de terminaison SPARQL, le résumé lié à un auteur (dbpedia-owl:abstract) est récupéré. Cela est facilement rendu possible car les données de Wikipédia sont alignées avec celles de data.bnf.fr (skos:exactMatch). À défaut, les notes bibliographiques de la BnF sont utilisées. D'autre part, pour certains auteurs des domaines artistiques ou philosophiques, sont récupérés 10 auteurs qui les ont influencés. Dans le cas où une réponse est obtenue, ils remplacent les auteurs nés la même année.

Europeana

À partir de la date de naissance d'un auteur (par défaut) ou de la date d'édition d'une partie de ses œuvres, il est intéressant de contextualiser la représentation en s'appuyant sur les ressources proposées par Europeana Collections par l'intermédiaire du point de terminaison SPARQL disponible. Cette requête va répondre en envoyant un ensemble d'œuvres numérisées diverses (de la poterie à la peinture en passant par l'orfèvrerie ou des partitions), créées ou éditées l'année concernée, et dont certaines ont pour source Gallica. Pour cette requête, les sources caduques et les ressources purement textuelles ont été délibérément exclues via un ensemble de filtres de type regex, dans la mesure où la contextualisation n'est alors pas aussi directe qu'un visuel.

Europeana a par ailleurs développé un modèle de structuration de données qui lui est propre, le « Europeana Data Model » ou EDM. Afin de retrouver un auteur lié à une œuvre exposée dans le contexte d'Europeana, une requête particulière *?auteur foaf:depiction <URI de l'image>* (explicitement : quelle entité a pour représentation cette image ?) est effectuée sur le triplestore local qui va tenter de retrouver celui-ci par sa représentation modélisée dans le contexte de data.bnf.fr.

Cela n'est valable bien entendu que pour les ressources visuelles qui pointent vers Gallica, Europeana utilisant d'autres sources de bibliothèques et musées européens qu'il est impossible d'aligner. De la sorte, il devient possible de partiellement « boucler le graphe » et de rebondir sur une entité à partir de cette contextualisation particulière.

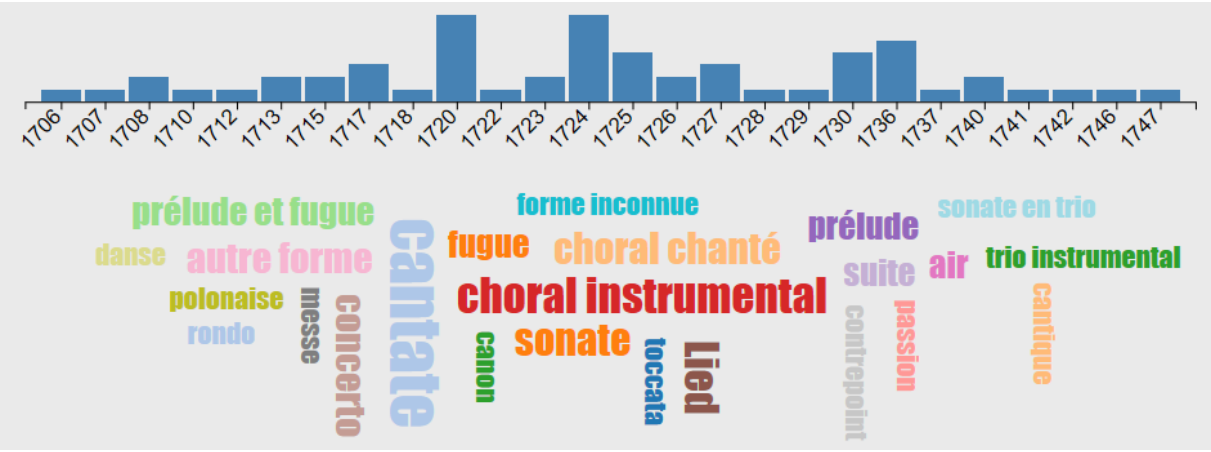
De la musique qui se manifeste et tente de s'exprimer sur une plage

Pour référence, selon le modèle FRBR, un enregistrement d'œuvre musicale se manifeste dans un disque et s'exprime dans une plage de ce support.

Les extraits musicaux, qui sont automatiquement joués lors du survol de la souris de certaines œuvres musicales, sont des flux audios, issus d'une transformation de l'URI qui pointe vers la

manifestation de l'œuvre sur Gallica, pour lesquelles un extrait sonore est disponible. Pour cela, et parce que l'information ne peut pas être déduite, la première plage (expression) est systématiquement pointée, ce qui est problématique dans la mesure où la plupart de ces manifestations sont agrégatives. Dans le meilleur des cas on a une écoute d'œuvre correspondant à l'ensemble (un mouvement d'une sonate particulière de Bach par exemple). Dans le pire des cas, pour certaines manifestations agrégeant des œuvres d'auteurs différents, on se retrouve avec une écoute qui ne correspond pas du tout à ce qui est décrit, tant en terme d'œuvre que d'auteur. Il s'agit là d'une preuve de concept qui ne sera malheureusement pas implémentée dans la version finale.

Prenons l'**exemple du choral instrumental de J.S. Bach** : *Ach bleib bei uns, Herr Jesu Christ, BWV 649* (transcription d'un extrait de la cantate BWV 6, édité en 1748-49) qui fait partie de *Sechs Choräle von verschiedener Art*, BWV 645 à 650 (« Chorals Schübler »).
Il s'agit d'une œuvre dont un enregistrement de 1955 existe (33 t ; Ed. de l'Oiseau Lyre) qui est exposé par Gallica. L'enregistrement est lié à la manifestation « Toccata, adagio et fugue en Ut majeur ; Prélude et fugue en Sol majeur ; … » : http://gallica.bnf.fr/ark:/12148/bpt6k8805036q
Par transformation de l'URI, nous pointons vers le flux de la première plage « à l'aveugle », durée=0:07:42 - *Prelude and Fugue in G Major, BWV 541* (Orgue : John Eggington) : http://gallica.bnf.fr/ark:/12148/bpt6k8805036q/f1.audio
Le flux du choral instrumental de départ, quant à lui, est sur la deuxième plage de la face B http://gallica.bnf.fr/ark:/12148/bpt6k8805036q/f4.audio
Une solution pourrait consister à analyser le flux sérialisé suivant dont l'URL peut être dérivée à partir de l'ARK http://gallica.bnf.fr/services/OAIRecord?ark=bpt6k8805036q duquel pourrait être extrait l'URI de l'expression correspondant à notre plage.



© Bibliothèque municipale de Fresnes

Mais un problème demeure : l'analyse serait alors basée sur une chaine de caractères issue du titre, ce qui n'est pas fiable et difficilement adaptable aux jeux de caractères autres que latins (c'est pourquoi les identifiants univoques et pérennes ont été inventés d'ailleurs).

Conclusion

L'une des grandes forces de la modélisation des données bibliographiques dans une logique ontologique - et dès lors que l'on envisage de manipuler les entités résultantes dans une interface - est la possibilité offerte d'avoir une approche globale et non fragmentaire des données et de leurs relations. Ceci est valable par la logique entité-relation inhérente au système, où chaque entité n'existe que par les interactions qu'elle entretient avec les autres entités de natures diverses. De ce fait, le graphe résultant, envisagé en tant que notion et de manière globale, peut alors potentiellement devenir le socle à partir duquel il est possible de contextualiser le discours, et ceci tout en ayant le choix de l'angle d'approche. En outre, la hiérarchisation des entités exposées par un graphe ne dépend que du sens qu'il expose, et non d'un modèle qui dépendrait d'une contrainte technique. Il s'agit alors vraiment d'une « représentation des connaissances », qui est un système utilisé dans la conception d'une ontologie.

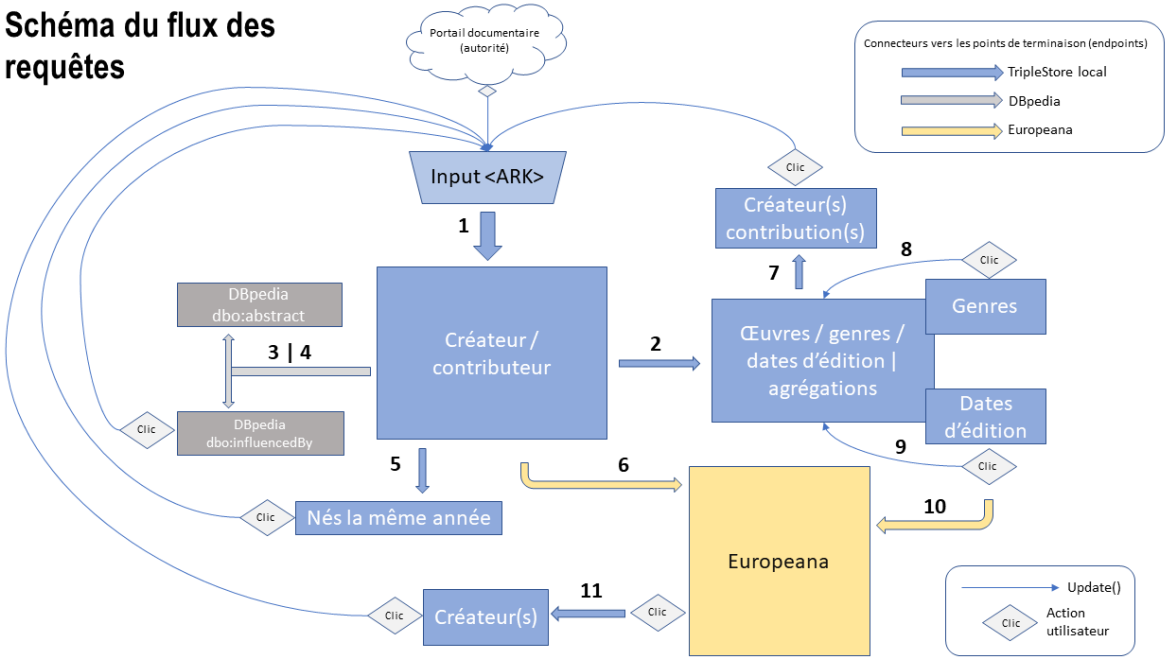
Le « petit laboratoire sémantique », en exploitant les données FRBRisées de data.bnf.fr pour les exposer sous forme de graphe au sens propre, est un point de départ pour expérimenter la concrétisation d'un ensemble de concepts abstraits transportant du sens de manière ludique. Les quelques démonstrations faites auprès d'un public restreint montrent un intérêt certain. Restera à déterminer, une fois l'outil mis en production et avec le recul nécessaire, s'il joue pleinement son rôle et ne demeure pas un simple gadget.

Se pose également la question de savoir si la représentation sous forme de graphe est le moyen privilégié pour transposer pleinement ces interactions et leur sens, et dans quelle mesure il ne serait pas plus intéressant d'imaginer une autre approche, peut-être plus subtile, pour intégrer cette sémantique dans une interface plus classique. Pour tout complément d'information auprès de Pierre BOURNERIE de la BM de Fresnes, merci d'utiliser la page contact du site Transition bibliographique.

Le processus des requêtes

Le schéma suivant décrit le flux des événements à partir du moment où l'ARK est récupéré, et les requêtes associées avec un bref descriptif. Il est impossible de prévoir à l'avance l'ordre dans lequel ces requêtes vont être envoyées et traitées, tout le processus étant asynchrone. Il faut imaginer que le code sous-jacent va devoir gérer un ensemble de « promesses » faites par le serveur de retourner une réponse, quelle qu'en soit la nature (il peut s'agir d'un code d'erreur). Certaines requêtes ont besoin du résultat d'autres requêtes, d'autres ne renverront pas de résultat, ou de manière partielle. Si dans les grandes lignes il est possible de dégager un processus « générique », il faut également envisager chaque requête comme étant une exception. Certaines entités auteurs n'auront pas d'œuvre référencée / liée, d'autres auront des alignements DBpedia <dbo:influencedBy> d'autres pas, certaines années n'auront aucune référence associée sur Europeana…

La BnF ne gère pas les alignements vers Europeana. Une requête à part est effectuée en passant une année en argument (l'année de naissance de l'entité par défaut) sur le point de terminaison Virtuoso. Un « alignement forcé » est effectué a posteriori en utilisant l'URI de la représentation de l'œuvre et en la reliant à une entité créateur (requête n°11). Le(s) créateur(s) dont la représentation d'une œuvre indexée dans Europeana sont reliés via le vocabulaire foaf (depiction) dans data.bnf.fr sont ainsi réinjectés dans le système par l'intermédiaire de l'ARK récupéré.



© Bibliothèque municipale de Fresnes

Les numéros renvoient au schéma du flux des requêtes ci-dessus.

Les « clauses » OPTIONAL sont là pour obtenir des résultats qui ne sont pas systématiques (certains auteurs par exemple n'ont pas de date de naissance renseignée). Si ces requêtes n'étaient pas incluses dans des clauses optionnelles, aucun résultat ne serait obtenu pour la requête globale.

ORDER BY RAND() LIMIT 100 : Les résultats d'une requête SPARQL sont ordonnés sous forme de tableau d'une manière prédéterminée par leur ordre d'arrivée. Lors d'une requête, le « pointeur » dans le tableau va partir de la position 0 (la première). Pour un nombre de résultats limité (dans le cas qui nous intéresse le graphe serait illisible si plus de 100 résultats étaient retournés concernant les œuvres), et pour éviter que l'on obtienne les même résultats à chaque requête, c'est la position du pointeur qui est déterminée de manière aléatoire grâce à la clause ORDER BY RAND().

Ci-dessous une représentation sous forme de tableau de la réponse à la première requête, avec l'ARK de Jules Verne passé en argument<http://data.bnf.fr/ark:/12148/cb11928016k >

La réponse reçue par le système à l'aide du connecteur triplestore, est un « *array* » (tableau contenant les données brutes sous-jacentes), qui va être sérialisé et analysé (*parsé*) au format *json* en *back-end*, puis exploité pour être interfacé sur une page web en *front-end* (d3.js).

	fimg	fonc	nom	pbirth	pdeath	dbirth	ddeath	wiki	field
1	http://gallica.bnf.fr/ark:/12148/btv1b531099559.thumbnail	Romancier et auteur dramatique	"Jules Verne (1828-1905)"@fr	Nantes	Amiens	1828-02-08	1905-03-24	http://fr.dbpedia.org/resource/Jules_Verne	Littératures

© Bibliothèque municipale de Fresnes

1/ La variable « wiki » contiendra l'URI de la ressource de l'auteur sur DBpedia, et sera exploitée lors de la requête n°3|4.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT distinct (SAMPLE(?uriImg) AS ?fimg) ?fonc ?nom ?pbirth ?pdeath ?dbirth ?ddeath ?wiki ?field
WHERE {
    <http://data.bnf.fr/ark:/12148/cb11928016k> <http://www.w3.org/2004/02/skos/core#prefLabel> ?nom .
    OPTIONAL{<http://data.bnf.fr/ark:/12148/cb11928016k#foaf:Person>
<http://rdvocab.info/ElementsGr2/fieldOfActivityOfThePerson> ?field .
        FILTER(isLiteral(?field)).}
    OPTIONAL{ <http://data.bnf.fr/ark:/12148/cb11928016k#foaf:Person> foaf:depiction ?uriImg.}
    OPTIONAL{ <http://data.bnf.fr/ark:/12148/cb11928016k#foaf:Person> <http://rdvocab.info/ElementsGr2/placeOfBirth> ?pbirth.}
    OPTIONAL{ <http://data.bnf.fr/ark:/12148/cb11928016k#foaf:Person> <http://vocab.org/bio/0.1/birth> ?dbirth.}
    OPTIONAL{ <http://data.bnf.fr/ark:/12148/cb11928016k#foaf:Person> <http://rdvocab.info/ElementsGr2/placeOfDeath> ?pdeath. }
    OPTIONAL{ <http://data.bnf.fr/ark:/12148/cb11928016k#foaf:Person> <http://vocab.org/bio/0.1/death> ?ddeath. }
    OPTIONAL { <http://data.bnf.fr/ark:/12148/cb11928016k#foaf:Person> <http://rdvocab.info/ElementsGr2/biographicalInformation>
?fonc.}
    OPTIONAL{<http://data.bnf.fr/ark:/12148/cb11928016k> <http://www.w3.org/2004/02/skos/core#exactMatch> ?wiki .
        FILTER (regex(str(?wiki),'dbpedia.org', 'i'))}
}
GROUP BY ?fonc ?nom ?pbirth ?pdeath ?dbirth ?ddeath ?wiki ?field
```

2/ Lors d'une requête de « récupération des œuvres », l'ensemble des genres musicaux exposés, s'il y en a, sont également récupérés, ainsi que les dates d'édition. La variable « ?pred » représente le prédicat (la relation qui existe entre la personne et l'œuvre considérée (créateur / contributeur…). Noter la variable ?aggreg qui va contenir une œuvre agrégative, l'œuvre considérée par la requête étant potentiellement agrégée par une autre.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT distinct ?nom ?work ?titre ?aggreg ?genre ?date ?desc ?pred (SAMPLE(?uriImg) AS ?fimg) (SAMPLE(?depic) as ?fdepic)
WHERE {
    <http://data.bnf.fr/ark:/12148/cb11928016k> <http://www.w3.org/2004/02/skos/core#prefLabel> ?nom .
    ?work ?pred <http://data.bnf.fr/ark:/12148/cb11928016k#about>;
          <http://www.w3.org/2000/01/rdf-schema#label> ?titre.
    OPTIONAL { ?work foaf:depiction ?depic. }
    OPTIONAL { ?work <http://www.openarchives.org/ore/terms/isAggregatedBy> ?aggreg .}
    OPTIONAL{?work <http://purl.org/ontology/mo/genre> ?uriG.
        ?uriG <http://www.w3.org/2004/02/skos/core#prefLabel> ?genre.}
    OPTIONAL{?work <http://rdvocab.info/Elements/dateOfWork> ?date.}
    OPTIONAL {?work <http://purl.org/dc/terms/description> ?desc}
    OPTIONAL { <http://data.bnf.fr/ark:/12148/cb11928016k#foaf:Person> foaf:depiction ?uriImg. }}
GROUP BY ?nom ?work ?titre ?aggreg ?genre ?date ?desc ?pred
ORDER BY RAND()
LIMIT 100
```

3|4 Récupération du résumé (dbpedia-owl:abstract) de l'entité, puis des éventuels « influenceurs », sur le point de terminaison de DBpedia.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?o ?page ?i ?nomInflu ?pageInflu
WHERE {
    <http://fr.dbpedia.org/resource/Jules_Verne> dbpedia-owl:abstract ?o.
    FILTER langMatches(lang(?o),'fr').
    <http://fr.dbpedia.org/resource/Jules_Verne> <http://www.w3.org/ns/prov#wasDerivedFrom> ?page.
    OPTIONAL{ <http://fr.dbpedia.org/resource/Jules_Verne> dbpedia-owl:influencedBy ?i.
        ?i foaf:name ?nomInflu;
        <http://www.w3.org/ns/prov#wasDerivedFrom> ?pageInflu;
        a dbpedia-owl:Person.
    }
}
ORDER BY RAND() LIMIT 20
```

5/ Un échantillon de 10 auteurs nés en 1828 (ici la date de naissance de Jules Verne), et qui ont au moins une œuvre référencée.

```
SELECT distinct (SAMPLE(?person) as ?pp) ?urip ?nom (COUNT(?work) AS ?countW)
WHERE {
    ?person <http://vocab.org/bio/0.1/birth> ?dbirth;
            <http://rdvocab.info/ElementsGr2/dateOfBirth> <http://data.bnf.fr/date/1828/>.
    ?urip <http://xmlns.com/foaf/0.1/focus> ?person.
    ?urip <http://www.w3.org/2004/02/skos/core#prefLabel> ?nom .
    ?work <http://purl.org/dc/terms/creator> ?person.
}
GROUP BY ?nom ?urip
HAVING (?countW > 1)
ORDER BY RAND() LIMIT 10
```

6/ La contextualisation Europeana (cette année-là…), par défaut la date de naissance de l'entité en cours d'exploration, divers filtres permettent d'éviter un certain nombre de ressources (principalement textuelles).

```
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
select ?title (SAMPLE(?obj) AS ?fimg) ?so ?creat ?idprov ?type
where {
    ?oe dc:date ?date; dc:title ?title;
                       dc:source ?so;
                       dc:type ?type;
                       dc:creator ?creat;
                       dc:identifier ?idprov;
                       ore:proxyIn ?proxy.
    ?proxy edm:object ?obj.
    FILTER (?date = "1828")
    FILTER (!regex(?type, "^Text", "i") && !regex(?type, "^Page record", "i"))
    FILTER (!regex(?obj, "^http://na.memorix.nl", "i"))
    FILTER (!regex(?idprov, "^oai:ebuw.uw.edu.pl", "i"))
    FILTER(regex(?idprov, "^http://", "i"))
}
ORDER BY RAND() LIMIT 100
```

7/ Accessoirement, si l'entité principale est contributeur d'une œuvre, obtenir son créateur.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT distinct (SAMPLE(?uriImg) AS ?fimg) ?fonc ?nom ?person
WHERE {
    <http://data.bnf.fr/ark:/12148/cb161496344#frbr:Work> <http://purl.org/dc/terms/creator> ?creator.
    ?person <http://xmlns.com/foaf/0.1/focus> ?creator;
    <http://www.w3.org/2004/02/skos/core#prefLabel> ?nom .
    OPTIONAL{ ?creator foaf:depiction ?uriImg.}
    OPTIONAL { ?creator <http://rdvocab.info/ElementsGr2/biographicalInformation> ?fonc.}
}
GROUP BY ?fonc ?nom ?person
LIMIT 1
```

8/ Exemple pris dans le cadre d'une requête « cantate » parmi les œuvres de Bach (l'utilisateur vient de cliquer sur le genre). Noter le filtrage pour le genre « cantate ».

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?work ?titre ?aggreg ?genre ?date ?desc ?pred (SAMPLE(?depic) as ?fdepic)
WHERE{
    ?work ?pred <http://data.bnf.fr/ark:/12148/cb118897907#about>;
          <http://www.w3.org/2000/01/rdf-schema#label> ?titre.
    OPTIONAL { ?work foaf:depiction ?depic. }
    OPTIONAL{?work <http://rdvocab.info/Elements/dateOfWork> ?date.}
    OPTIONAL {?work <http://purl.org/dc/terms/description> ?desc.}
    OPTIONAL { ?work <http://www.openarchives.org/ore/terms/isAggregatedBy> ?aggreg .}
    OPTIONAL{ ?work <http://purl.org/ontology/mo/genre> ?uriG. ?uriG <http://www.w3.org/2004/02/skos/core#prefLabel> ?genre.
    }
    FILTER regex(?genre, "cantate", "i")
}
GROUP BY ?work ?titre ?aggreg ?genre ?date ?desc ?pred
ORDER BY RAND() LIMIT 100
```

9/ Les œuvres de Jules Verne en 1889 (l'utilisateur vient de cliquer sur cette date spécifique).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?work ?titre ?aggreg ?genre ?date ?desc ?pred (SAMPLE(?depic) as ?fdepic)
WHERE{
    ?work ?pred <http://data.bnf.fr/ark:/12148/cb11928016k#about>;
          <http://www.w3.org/2000/01/rdf-schema#label> ?titre.
    OPTIONAL { ?work foaf:depiction ?depic. }
    OPTIONAL{?work <http://rdvocab.info/Elements/dateOfWork> ?date.}
    OPTIONAL {?work <http://purl.org/dc/terms/description> ?desc.}
    OPTIONAL { ?work <http://www.openarchives.org/ore/terms/isAggregatedBy> ?aggreg .}
    OPTIONAL{ ?work <http://purl.org/ontology/mo/genre> ?uriG.
        ?uriG <http://www.w3.org/2004/02/skos/core#prefLabel> ?genre. }
    FILTER regex(str(?date), 'http://data.bnf.fr/date/1889/')
}
GROUP BY ?work ?titre ?aggreg ?genre ?date ?desc ?pred
ORDER BY RAND() LIMIT 100
```

10/ Cf. requête 6/ avec la date cliquée en référence.

11/ Le(s) créateur(s) dont la représentation d'une œuvre indexée dans Europeana est reliée via le vocabulaire foaf (foaf:depiction) dans data.bnf.fr. Dans ce cas <http://gallica.bnf.fr/ark:/12148/btv1b77413194.thumbnail>

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT distinct *
WHERE {
    ?creat foaf:depiction <http://gallica.bnf.fr/ark:/12148/btv1b77413194.thumbnail>;
    foaf:name ?nom
}
```

Lexique des termes et concepts employés

Triplestore

Un triplestore est une base de données dévolue au stockage et au traitement de données modélisées en RDF. Un *triple* (en français triplet) est une déclaration sous la forme sujet / prédicat / objet exposé par RDF. Le langage de requête sur un triplestore est le SPARQL.

Back-end / Front-end

Dès lors qu'il s'agit de développer des pages web devant afficher des résultats dynamiquement (la majorité des pages web aujourd'hui), il est essentiel de bien comprendre l'interaction client-serveur. Le serveur héberge le site, sa structure et tout le code devant être exécuté lors d'une requête de page (c'est le code back-end). Le client, représenté par l'ordinateur qui accède au site, a quant à lui également la possibilité d'exécuter du code (c'est le code front-end). Tout l'art du développement web est en grande partie dévolu à gérer l'interaction client/serveur.

Asynchronisme

Dans le contexte du web, lorsqu'un serveur et un client interagissent, ils ont deux manières d'aborder cette interaction. Imaginons un restaurant dans lequel un serveur prendrait une commande et attendrait qu'elle soit honorée avant de passer à la suivante. Le client connait la règle et patiente donc : c'est un processus synchrone, lent et couteux. Si maintenant le serveur peut prendre plusieurs commandes à la fois et les servir au fur et à mesure de leur disponibilité, il s'agit d'un processus asynchrone, bien plus rapide. Notez que dans ce cas le client qui fait sa demande en premier n'est pas forcément le premier servi… Pour poursuivre l'analogie avec le restaurant, le cuisinier est représenté par le couple microprocesseur/mémoire qui donne l'illusion, en passant d'une tâche à l'autre très rapidement, d'être multitâches.

Dump

Un dump est une copie (partielle ou complète) d'une base de données, qu'il est possible d'exploiter en l'état.

Framework

Un framework est un cadre applicatif homogène et générique qui expose une architecture structurelle. Une application peut décider de s'appuyer sur un framework particulier. Sa structure va alors en dépendre.

Regex

Une regex ou expression régulière, va s'appuyer sur un motif (*pattern*) et une syntaxe particulière afin d'analyser une chaîne de caractères ou un ensemble de chaîne de caractères. Soit en vue d'analyser une chaîne « régulière » (dont la structure est toujours la même), soit en vue de la filtrer en fonction du pattern. Par exemple le pattern « ^Jules » va permettre de filtrer la recherche aux chaînes commençant par (caractère ^) Jules.

Liens utiles

http://data.bnf.fr/opendata
http://wiki.dbpedia.org/about
https://pro.europeana.eu/resources/apis/sparql
http://metadataregistry.org

Points de terminaison

Europeana : http://sparql.europeana.eu/
data.bnf.fr : http://data.bnf.fr/sparql/
DBpedia : https://dbpedia.org/sparql

Pour le groupe Systèmes & Données : Pierre BOURNERIE, BM de Fresnes