Cassiane CLOCHARD

Licence 3 Humanités Numériques

UNIVERSITÉ CAEN NORMANDIE

Le ghetto de Brest-Litovsk

A la mémoire d'un peuple

MRSH PDN Pôle Document Numérique

Table des matières

Introduction

https://ghettobrest.hypotheses.org/

Le ghetto de Brest-Litovsk à la croisée des Humanités Numériques

Ouvert en 1941, le ghetto de Brest-Litovsk se trouvait entre la Pologne et la Biélorussie. Il a été créé afin de concentrer la population juive pendant la seconde guerre mondiale. Plus de 17000 juifs y ont été exécuté, avant sa fermeture en 1942.

Le ghetto de Brest-Litovsk n'a jamais été étudié, mais des documents ont été retrouvés et ont permis le début de recherches afin de le reconstituer, en faire une édition numérique afin de rendre leurs identités aux personnes qui y ont vécu. Les archives ont été transcrites, ce qui a permis d'obtenir beaucoup d'informations, comme le nom, prénom, le sexe et la profession d'une personne. Ainsi, il serait possible d'effectuer des recherches par années, par profession, de dégager des règles de regroupement afin de pouvoir trouver les identités des personnes en question.

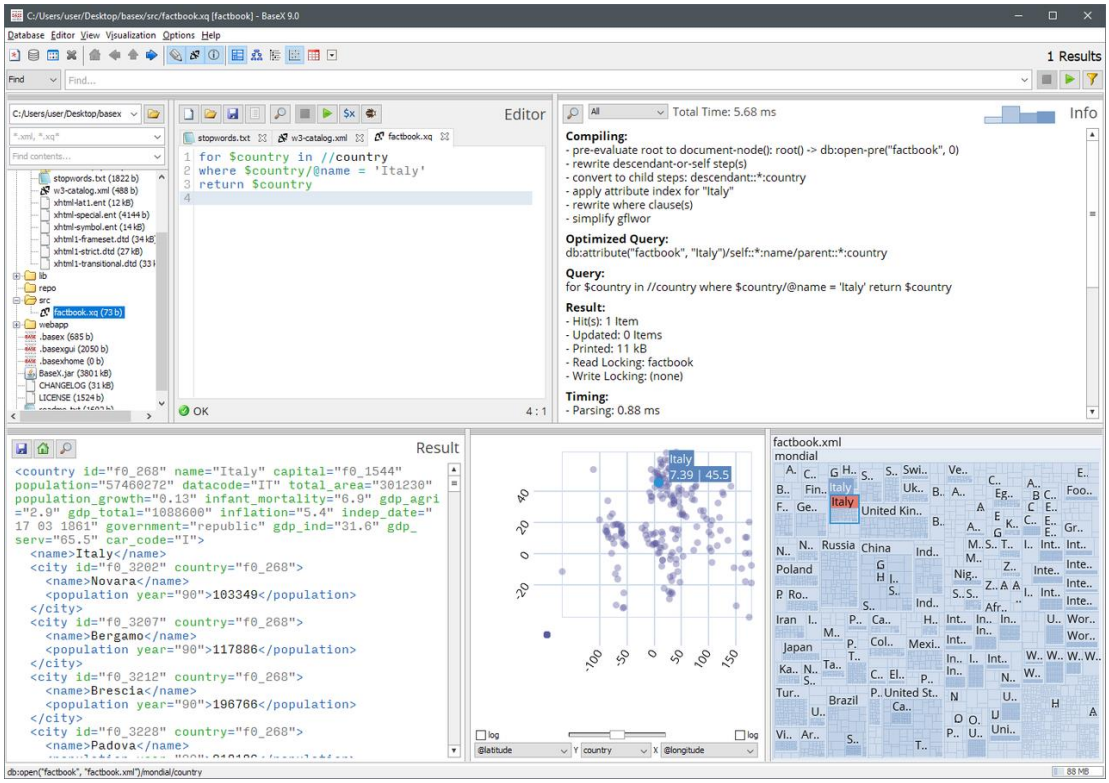
Pour cela, nous utilisons le langage XQuery avec le logiciel BaseX, destiné à interroger une vaste base de données XML, rendant ainsi les détails historiques, géographiques, et

personnels des victimes de la Shoah plus accessibles (annexe 1). Cette approche permet à tout un chacun de formuler des recherches complexes de manière intuitive, qu'il s'agisse de retrouver l'histoire d'un individu spécifique, de comprendre la structure sociale du ghetto, ou de visualiser les connexions géographiques entre différents lieux de vie et de travail forcé. L'utilisation de XQuery sert de pont entre les richesses de notre base de données et les utilisateurs désireux d'explorer cette période sombre de l'histoire avec précision et empathie.

En résumé, ce projet consiste à faciliter la navigation et la recherche dans une base de données mémorielle grâce à des requêtes XQuery. Cette démarche technique permet non seulement de personnaliser l'expérience de recherche en fonction des intérêts et des besoins spécifiques de chaque utilisateur mais aussi de garantir que chaque victime du ghetto de Brest-Litovsk soit rappelée et honorée, contribuant ainsi à la conservation de leur mémoire et à l'éducation des générations futures.

Le logiciel BaseX

BaseX est un système de gestion de base de données XML créé en 2005 par Christian Grün. Il a été spécialement conçu pour stocker, gérer et interroger des données XML de manière efficace. Il offre un ensemble complet de fonctionnalités pour travailler avec des documents XML, facilitant ainsi le traitement et l'analyse de données structurées. Il prend en charge des langages tels que **XPath** et **XQuery**, offrant ainsi la possibilité d'interroger et de manipuler les données de manière flexible. Ce logiciel devient open source en 2007.



BaseX montrant un document XML dans plusieurs visualisations, Christian Grün, Wikipédia.

XPath, XQuery et FLOWR

XPath et XQuery sont deux technologies liées à la manipulation de documents XML, mais ont des distinctions.

XPath, qui signifie XML Path Language, est essentiellement conçu pour permettre la navigation et la sélection de parties spécifiques d'un document XML. Son utilisation principale est de localiser rapidement des informations dans un document XML en définissant un chemin spécifique qui parcourt la structure du document. Cette technologie est particulièrement utile pour extraire des nœuds ou des ensembles de nœuds spécifiques, comme des éléments, des attributs ou des sections de texte.

XQuery, ou XML Query Language, est un langage de requête XML standardisé, largement utilisé pour interroger, manipuler et extraire des données stockées sous

forme XML. Il a été conçu par le World Wide Web Consortium (W3C). XQuery étend les capacités de XPath en permettant non seulement de sélectionner des données, mais également de construire de nouvelles structures XML.

XQuery supporte des opérations complexes telles que les jointures entre documents, le tri, l'agrégation de données, et peut même effectuer des mises à jour conditionnelles et créer de nouveaux documents XML. Cela en fait un choix idéal pour les applications nécessitant une manipulation intensive de données XML, comparable en plusieurs aspects aux fonctionnalités offertes par SQL dans le contexte des bases de données relationnelles.

Une caractéristique clé de XQuery est l'expression FLWOR : For, Let, Where, Order by, Return. Cela permet de formuler des requêtes complexes en combinant différentes opérations. Avec FLWOR, les utilisateurs peuvent par exemple filtrer les données, trier les résultats et spécifier ce qui doit être retourné dans le résultat final.

For : Cette étape permet de spécifier une ou plusieurs expressions qui génèrent des séquences de données à partir d'éléments sources. Elle est similaire à la boucle *for* que l'on trouve dans de nombreux langages de programmation, mais elle fonctionne aussi sur des séquences de données XML.

Let : L'étape *let* permet de définir des variables pour stocker des valeurs intermédiaires ou calculées à utiliser dans la suite de l'expression FLWOR. Elle offre une manière pratique de rendre le code plus lisible et plus modulaire.

Where : Cette étape permet de filtrer les données en spécifiant des conditions logiques. Elle est utilisée pour sélectionner uniquement les éléments qui satisfont certaines conditions définies dans l'expression XQuery.

Order By : Order by est utilisé pour ordonner les résultats selon un critère spécifié, tel que le contenu d'un élément ou d'un attribut. Elle permet de contrôler l'ordre dans lequel les résultats sont renvoyés par la requête XQuery (ordre croissant, décroissant...).

Return : L'étape "return" est utilisée pour spécifier les résultats finaux de la requête XQuery. Elle définit ce que la requête doit retourner une fois que toutes les opérations précédentes ont été effectuées.

Etat de l'art

Lodz-names

D'autres projets similaires à celui de la reconstitution du ghetto de Brest-Litovsk ont été fait. On peut notamment citer un projet appelé Lodz-Names.

https://www.jewishgen.org/databases/poland/lodzghetto.html#P2.3

En février 1940, un ghetto fut créé par les allemands dans la ville de Lodz, en Pologne. Plus de 240000 personnes y ont habité. Une base de données est disponible, comprenant des informations sur chaque personne.

Cette base de données permet de pouvoir faire des recherches sur les habitants, comme pour le ghetto de Brest-Litovsk. Pour Lodz, la base de données a notamment servi à pouvoir reconstituer des unités familiales (annexe 2).

La base de données centrale des noms des victimes de la Shoah

La Base de données centrale des noms des victimes de la Shoah est un projet dirigé par Yad Vashem. Il vise à retrouver le nom de chaque victime de la Shoah et à reconstituer son histoire. Lancée en 2004 avec près de trois millions de noms, elle contient désormais près de quatre millions et demi de noms de Juifs assassinés pendant la Shoah. Cependant, de nombreux noms n'ont pas encore été identifiés et de nouveaux sont constamment ajoutés à la base de données. Les efforts pour obtenir des informations fiables sur le sort des victimes se poursuivent.

En 2014, la base de données a été élargie pour inclure des informations sur les Juifs persécutés pendant la Shoah, y compris ceux dont le sort reste à éclaircir. Elle a également été enrichie en janvier 2021 pour inclure les noms des survivants de la Shoah, tout en respectant les restrictions de confidentialité. Les dépositions peuvent prendre jusqu'à

six mois pour être intégrées, car elles sont vérifiées par le personnel de Yad Vashem pour garantir leur exactitude historique.

Il s’agit donc d’une base de données qui peut elle aussi être traitée en XQuery, afin d’obtenir des recherches précises.

Tout ces projets rendent leurs identités aux victimes juives.

Mise en œuvre

Les requêtes demandées

Démographie : Par sexe ; sexe et âge

Épigraphie : Langues de signature

Géographie : Lieux de naissance ; lieux de vie dans le ghetto

Espace social : Professions et adresse dans le ghetto

Date d'enregistrement + adresse pour répondre à la question : selon quel(s) critère(s) les individus ont-ils été enregistrés ?

Les bases de données à notre disposition

Nous avons trois bases de données XML à notre disposition : A.xml, B.xml, D.xml. Chaque lettre correspond à la première lettre du nom de famille d'une personne. Ces bases de données sont complexes, elles demandent donc des requêtes précises.

Les requêtes par sexe des personnes

Afin de permettre une familiarisation avec les bases de données, j'ai commencé à chercher une requête permettant de donner tous les noms des femmes, modifiable afin de pouvoir également donner celui des hommes.

Le sexe d'une personne est défini par « f » (sexe féminin) et « m » (sexe masculin), définit dans la variable *person* :

```
<person xml:id="a_p0" sex="f">
```

Cependant, même avec une requête valide, aucun résultat n'était donné. Je suis donc allée chercher du côté de la base de donnée afin de comprendre ce qui ne convenait pas.

C'est le namespace TEI qui pose problème :

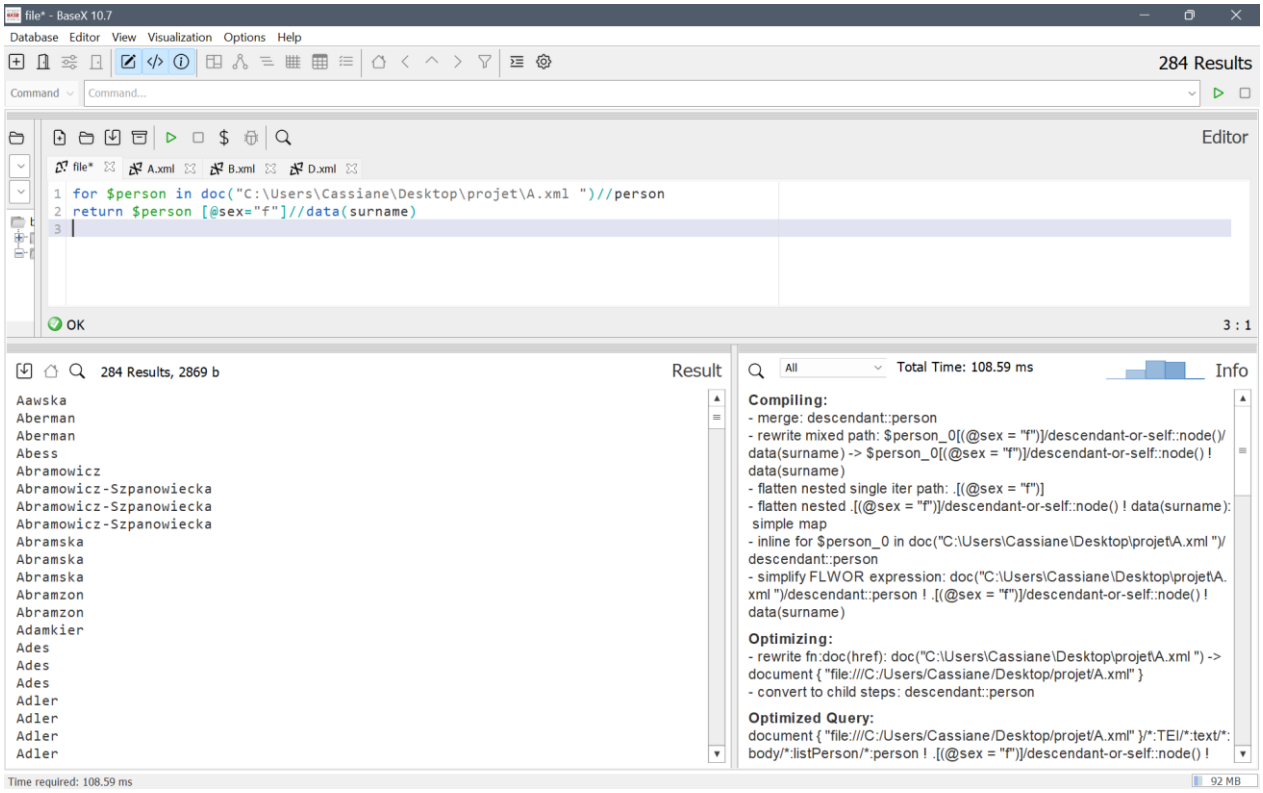
```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
```

En faisant une requête ainsi, aucun résultats n'est trouvé car les namespace sont pris en compte : c'est une recherche dans les données sans namespace, la requête renvoie donc 0 résultat car il n'y en a pas.

Deux solutions sont possibles face à ce problème.

Nous pouvons d'abord enlever ce namespace en modifiant les bases de données A.xml, B.xml et D.xml, en ne laissant que la balise <TEI>, au lieu de <TEI xmlns="http://www.tei-c.org/ns/1.0">.

Voici le résultat obtenu pour la base de données A :

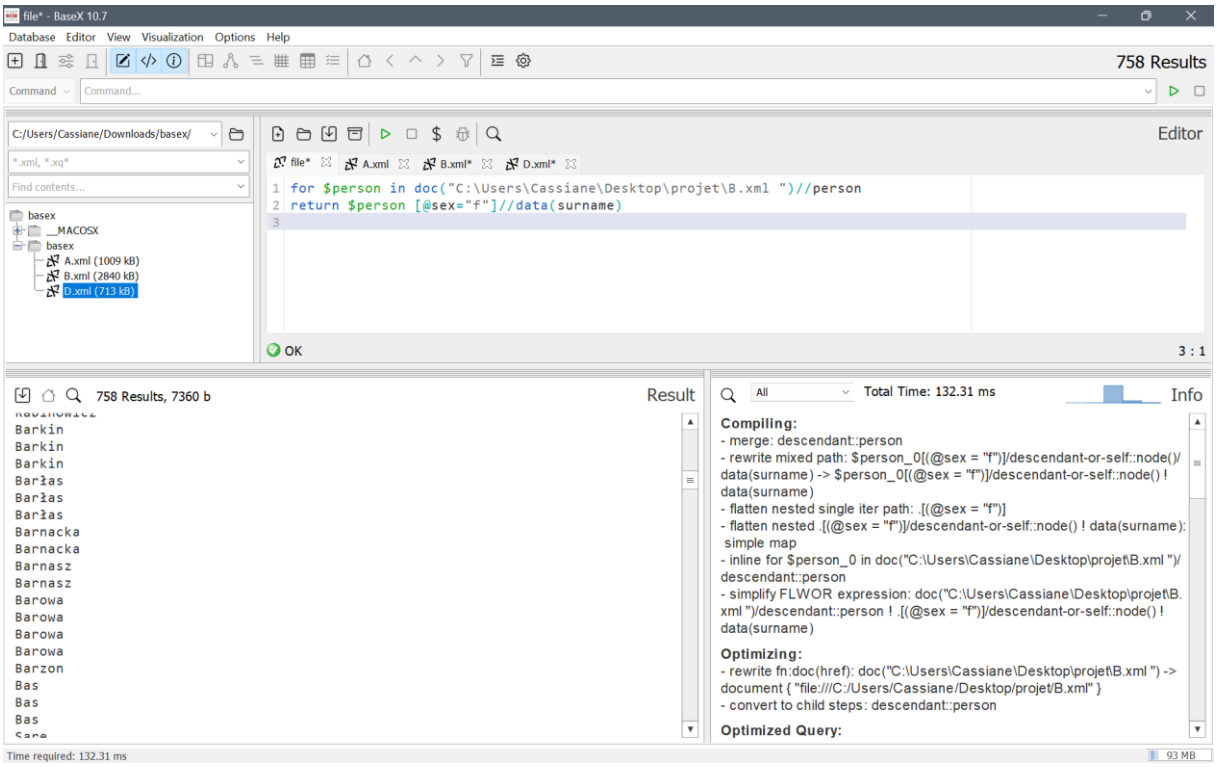


Il est également possible, et ce sans modifier la base de données, de sélectionner les données de n'importe quel namespace, avec « : » placé devant tous les éléments. Afin de faciliter les expressions régulières, j'ai pris le parti d'utiliser la première option, en supprimant le namespace. Il est néanmoins parfaitement possible de faire avec les deux méthodes.

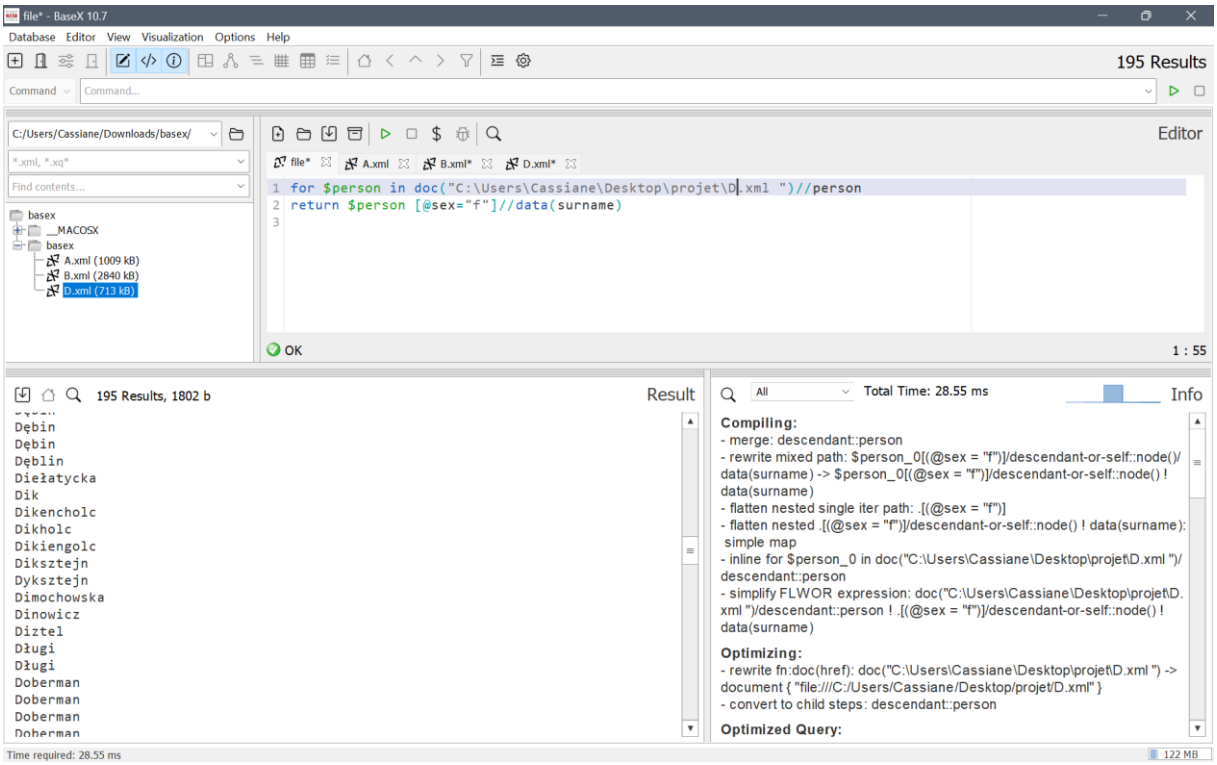
Voici donc la requête qui donne les noms de toutes les femmes :

```
for $person in doc("path/to/xml ")//person
return $person [@sex="f"]//data(surname)
```

Voici les résultats pour la base de données B :



Voici les résultats pour la base de données D :



Pour trouver les noms d'hommes, il suffit juste de remplacer le « f » par un « m » :

```
for $person in doc("path/to/xml")//person
return $person [@sex="m"]//data(surname)
```

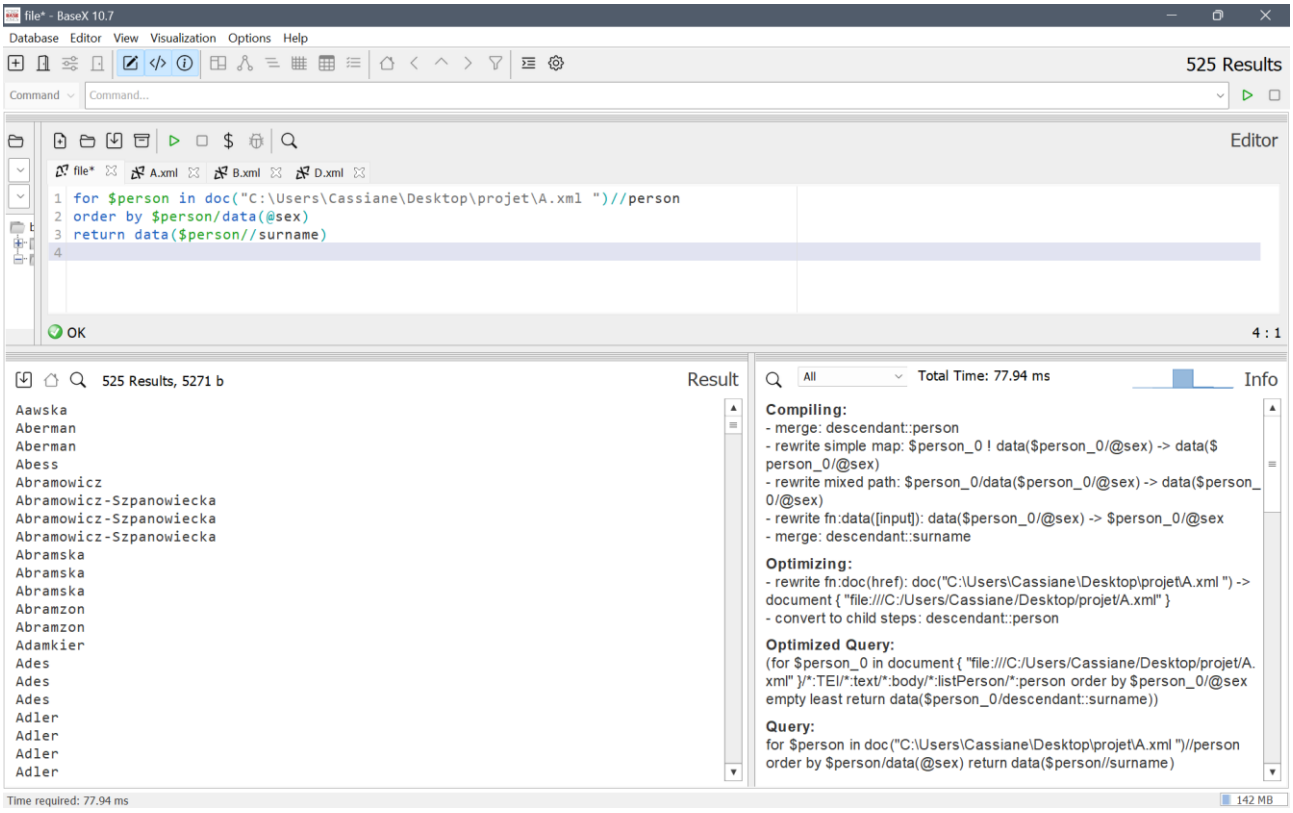
Dans les bases de données A, B et D, les personnes sont déjà triées par ordre alphabétique. Il n'est donc pas nécessaire d'utiliser *order by*. Le résultat suivra les données : le résultat sera dans l'ordre alphabétique des noms de famille.

Après avoir trouvé la requête pour trouver les femmes, puis les hommes, nous pouvons chercher comment trier les personnes par sexe. Pour cela, j'ai utilisé *order by*, afin d'avoir tous les noms de femmes, puis d'hommes.

```
for $person in doc("path/to/xml ")//person
order by $person/data(@sex)
return $person//surname
```

Pour que seuls les noms soient visibles, sans les éléments, nous pouvons ajouter *data* au niveau du *return*.

Voici le résultat :



Les requêtes par âge / date de naissance

Dans les bases de données, l'âge n'est pas donné, mais la date de naissance l'est. La date de naissance est regroupée avec le lieu de naissance dans un *notetype* commun :

Afin d'extraire seulement la date, il faut faire usage d'expressions régulières.

Les expressions régulières dans XQuery permettent de rechercher des motifs ou des modèles de texte dans les données XML. Elles sont utilisées pour effectuer des

opérations de recherche et de remplacement, ainsi que pour valider et extraire des données basées sur des modèles spécifiques.

Voici la requête pour trier selon la date de naissance et le sexe :

for $person in doc("path/to/xml ")/TEI/text/body/listPerson/person → *le chemin est écrit en entier pour rechercher directement par personne*.

let $surname := data($person/persName/surname) → *crée une variable, avec le chemin complet également, au cas où d'autres noms seraient détectés (par exemple, les prénoms des parents)*.

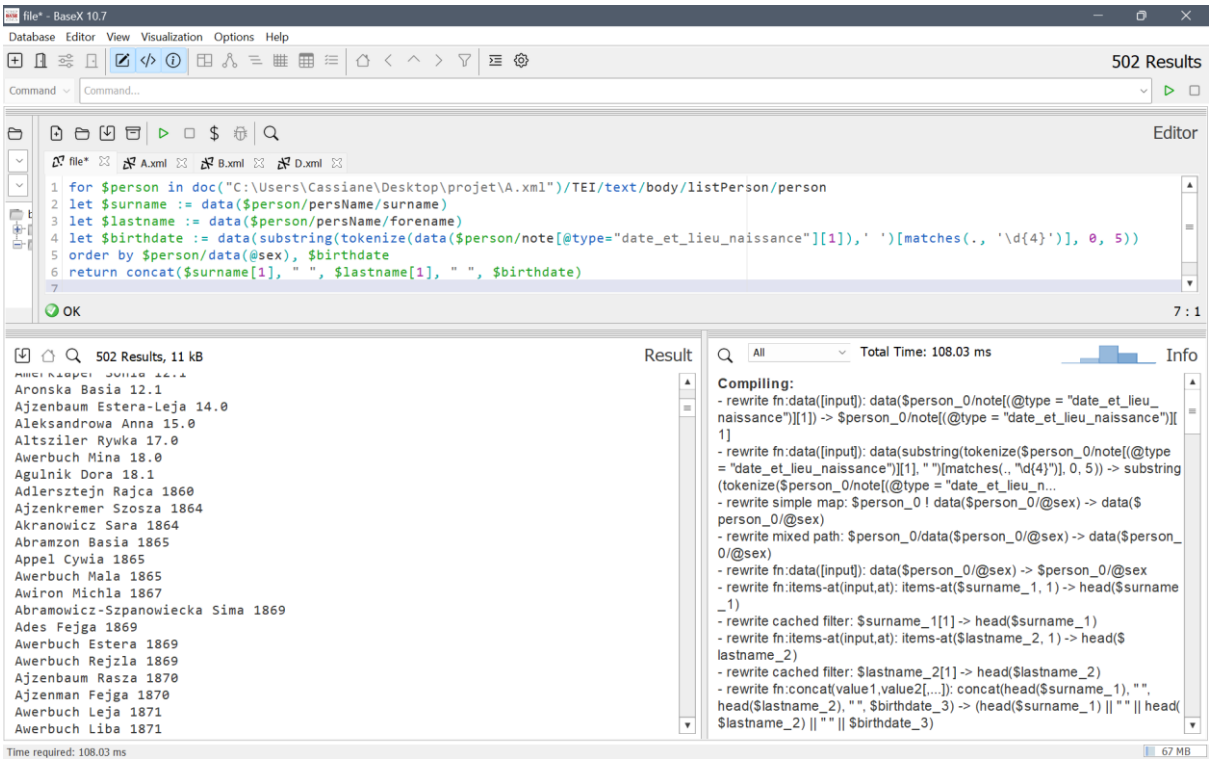
let $lastname := data($person/persName/forename) → *crée une variable, avec le chemin complet également, au cas où d'autres noms seraient détectés (par exemple, les noms des parents)*.

let $birthdate := data(substring(tokenize(data($person/note[@type="date_et_lieu_naissance"][1]),' ')[matches(., '\d{4}')], 0, 5)) → *substring (0, 5) sert ici à ne garder que les 4 premiers caractères, qui sont l'année de naissance. Tokenize sert à découper la chaîne de caractère afin de garder qu'une certaine partie (en l'occurrence la date et le lieu de naissance, sans par exemple « est né à »). Matches repère les 4 chiffres, ceux de l'année*.

order by $person/data(@sex), $birthdate → *organise les résultats par dates de naissance croissantes*.

return concat($surname[1], " ", $lastname[1], " ", $birthdate) → *le résultat sera constitué du nom, du prénom et de l'année de naissance*.

Voici le résultat obtenu sur la base de donnée des noms en A :



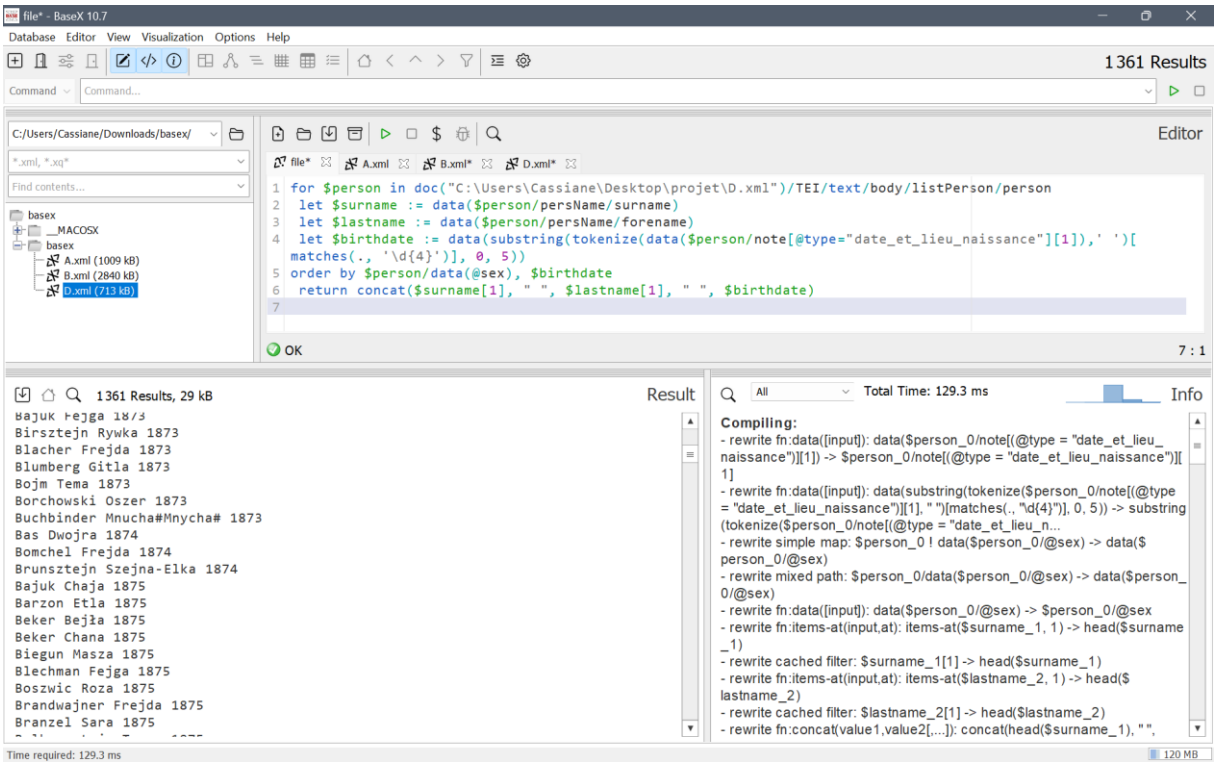
Les résultats obtenus avec cette requête ne sont pas optimaux. Effectivement, certains noms ne ressortent avec aucune date de naissance, et d'autres avec des données telles que 14.0 ou 15.0.

Ce résultat est dû au fait que l'expression régulière détecte les 4 premiers caractères, car dans la majorité du temps cela correspond à la date de naissance. Cependant, pour certaines personnes, aucune date de naissance n'est donnée. Pour d'autres, le jour et le mois sont précisés. Voici ce qui est spécifié dans le notetype date et lieu de naissance pour Basia Aronska :

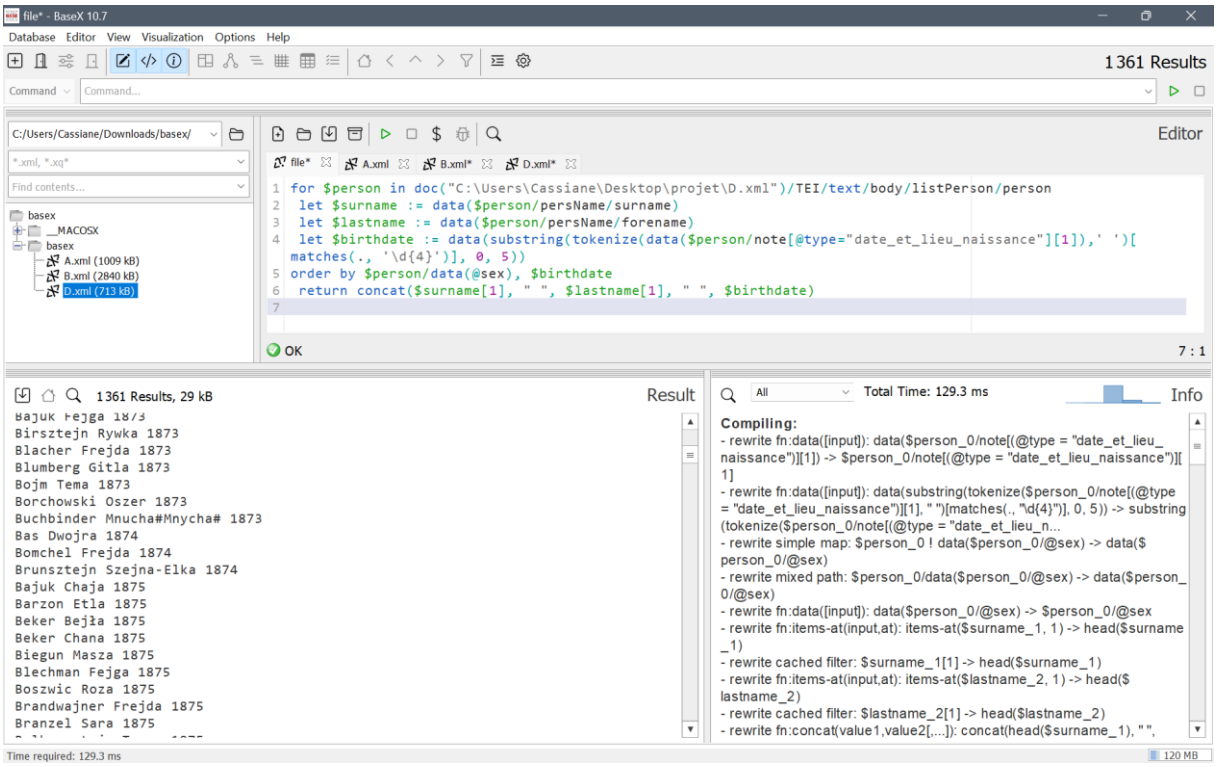
```
<note type="date_et_lieu_naissance">12.11.1908 à Podol.</note>
```

Les 4 premiers caractères détectés ne sont pas l'année de naissance, mais 12.1, qui correspond au jour et au premier numéro du mois de naissance. Les résultats sont donc faussés. Je n'ai pas encore trouvé le moyen d'extraire seulement l'année de naissance.

Voici les résultats pour la base de données B :



Voici les résultats pour la base de données D :



La requête par lieu de naissance et celle par langue de signature demandent également l'utilisation d'expressions régulières. Cependant, nous ne pouvons pas extraire ces données de la même manière que pour la date de naissance, car les lieux de naissances et les langues (polonais, russe…) ont un nombre de lettres différent. Il n'est donc pas possible de chercher dans le notetype une suite précise. Je n'ai pas encore réussi à imaginer des requêtes pour ces deux critères de recherche.

Requête globale

Ces expressions servent à chercher des gens. Il m'a donc semblé juste de chercher une requête qui rassemble toutes les informations principales d'une personne.

for $person in doc("path/to/xml ")/TEI/text/body/listPerson/person → *le chemin est écrit en entier pour rechercher directement par personne*.

let $surname := data($person/persName/surname) → *crée une variable, avec le chemin complet également, au cas où d'autres noms seraient détectés (par exemple, les prénoms des parents)*.

let $lastname := data($person/persName/forename) → *crée une variable, avec le chemin complet également, au cas où d'autres noms seraient détectés (par exemple, les noms des parents)*.

let $birthdate := data(substring(tokenize(data($person/note[@type="date_et_lieu_naissance"][1]),' ')[matches(., '\d{4}')], 0, 5)) → *crée une variable qui donne l'année de naissance dans la majorité des cas*.

let $sexe := $person/data(@sex) → *crée une variable qui donne le sexe de la personne*.

let $date_enregistrement := $person/note[@type="date_creation_document"][1] → *crée une variable avec la date d'enregistrement de la personne dans le ghetto*.

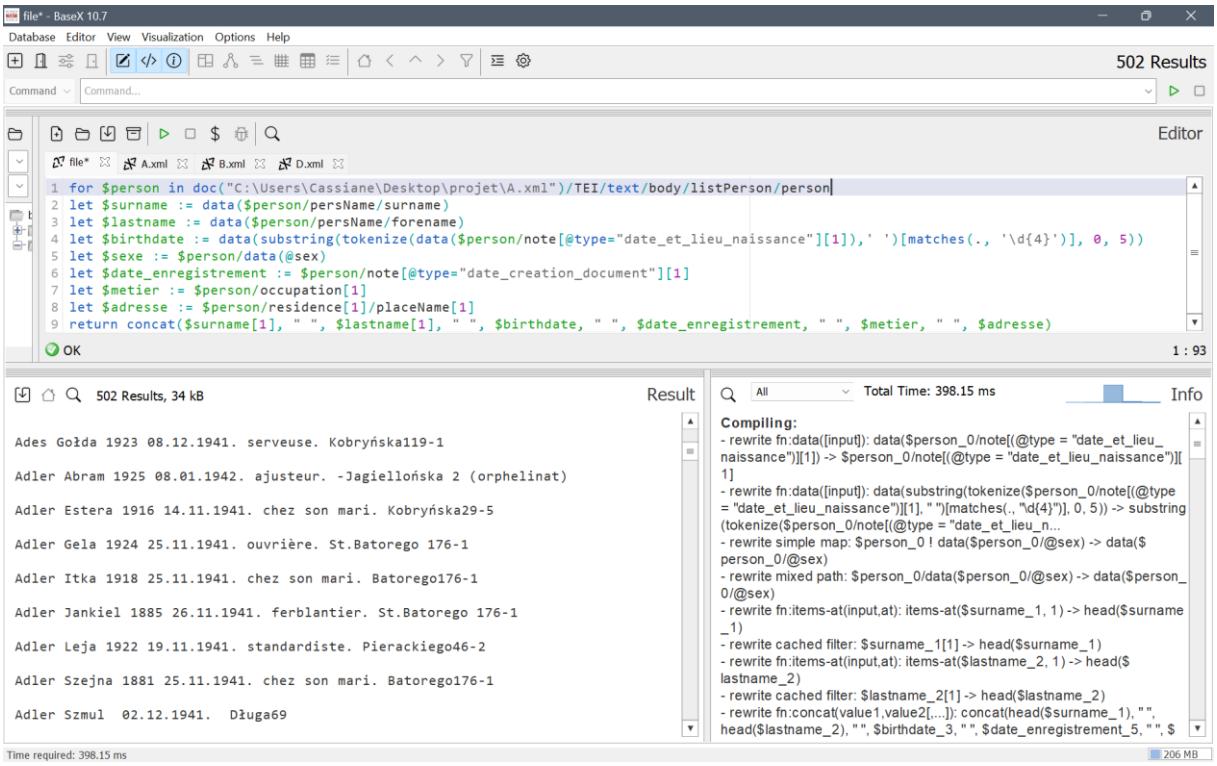
let $metier := $person/occupation[1] → *crée une variable avec le métier de la personne*.

let $adresse := $person/residence[1]/placeName[1] → *crée une variable qui donne le lieu de résidence*.

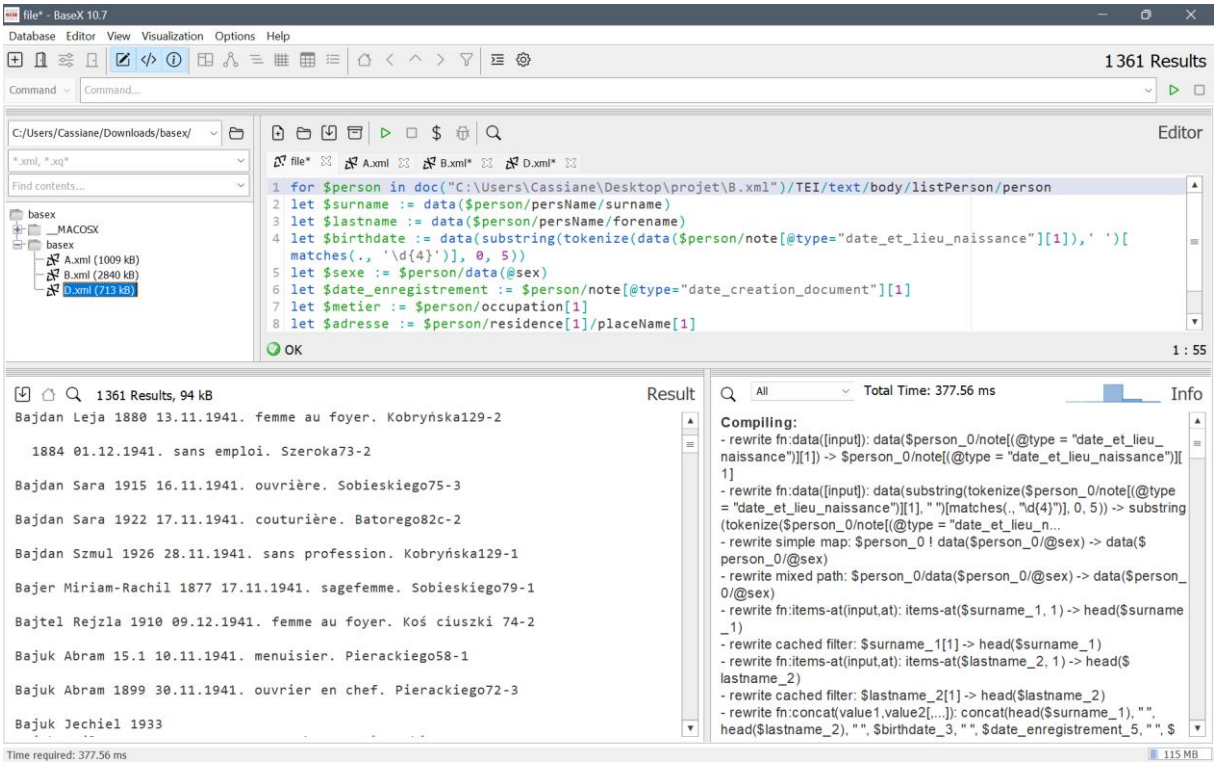
return concat($surname[1], " ", $lastname[1], " ", $birthdate, " ", $date_enregistrement, " ", $metier, " ", $adresse) → *le résultat sera constitué du nom, du prénom, de l'année de naissance, de la date d'enregistrement, du métier et de l'adresse d'une personne*.

N.B : [1] sert à ne prendre que la première donnée, si il y en a plusieurs.

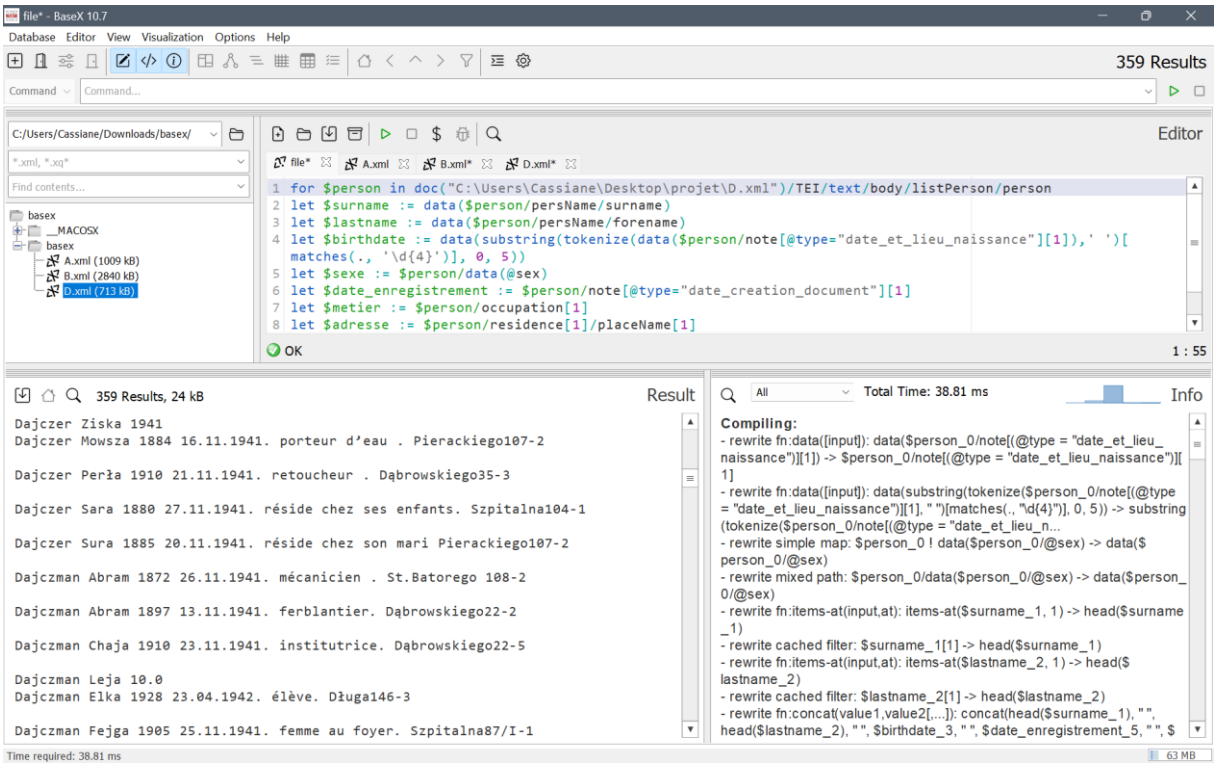
Voici le résultat pour la base de données A, nous obtenons bien les données disponibles pour chaque personne :



Voici les résultats obtenus pour la base de données B :



Voici les résultats obtenus pour la base de données D :



Conclusion

Le projet

Le projet de reconstitution du ghetto de Brest-Litovsk consiste à regrouper des données sur les habitants afin de rendre leur identité aux victimes de la Shoah.

Afin de faire des recherches à travers cette base de données XML, nous pouvons utiliser XQuery et FLOWR, pour faire des recherches avec critères. Nous pouvons notamment trier les personnes par sexe, par date et lieu de naissance, etc...

Les requêtes

Nous pouvons utiliser cette requête pour classer les personnes par sexe :

```
for $person in doc("path/to/xml")//person
order by $person/data(@sex)
return $person//surname
```

Voici la requête pour classer les personnes par année de naissance et sexe :

```
for $person in doc("path/to/xml")/TEI/text/body/listPerson/person
let $surname := data($person/persName/surname)
let $lastname := data($person/persName/forename)
let $birthdate := data(substring(tokenize(data($person/note[@type="date_et_lieu_naissance"][1]),' ')[matches(., '\d{4}')], 0, 5))
order by $person/data(@sex), $birthdate
return concat($surname[1], " ", $lastname[1], " ", $birthdate)
```

Cette requête n'est pas exacte, car l'expression régulière détecte les 4 premiers caractères qui sont assimilés à l'année de naissance. Mais les premiers caractères trouvés ne sont pas toujours l'année. Le jour et le mois de naissance sont parfois connu, pour d'autres, aucune date de naissance n'est enregistrée.

Les requêtes par lieu de naissance et par langue de signature demandent également l'utilisation d'expressions régulières. Je n'ai pas pu les réaliser car nous ne pouvons pas extraire ces données de la même manière que pour la requête avec la date de naissance, car les lieux de naissances et les langues de signatures (polonais, russe...) ont un nombre de lettres différent. Il n'est donc pas possible de chercher dans le notetype une suite précise.

Voici une requête qui regroupe plus globalement des informations des personnes :

```
for $person in doc("path/to/xml")/TEI/text/body/listPerson/person
let $surname := data($person/persName/surname)
let $lastname := data($person/persName/forename)
let $birthdate := data(substring(tokenize(data($person/note[@type="date_et_lieu_naissance"][1]),' ')[matches(., '\d{4}')], 0, 5))
let $sexe := $person/data(@sex)
let $date_enregistrement := $person/note[@type="date_creation_document"][1]
let $metier := $person/occupation[1]
let $adresse := $person/residence[1]/placeName[1]
return concat($surname[1], " ", $lastname[1], " ", $birthdate, " ", $date_enregistrement, " ", $metier, " ", $adresse)
```

Commentaires sur le code XML utilisé pour les bases de données

Les bases de données A, B et D regroupent les éléments complexes. Par exemple, pour une seule personne, il y a beaucoup d'éléments notetype :

```
<note type="based_document">passeport soviétique délivré à Brest en1940.</note>
<note type="origine">résident permanent.</note>
<note type="date_creation_document">20.11.1941.</note>
<note type="avec_photographie">oui.</note>
<note type="avec_empreinte_digitale">oui.</note>
<note type="avec_signature_individuelle">oui (en polonais).</note>
<note type="sources">201.1.26, pages 33, 33 verso. </note>
<note type="lien">
```

Chaque notetype aurait pu être redivisé en éléments à part entière. Pour le notetype « date_et_lieu_de_naissance », deux éléments auraient pu être créé : un élément date, et un élément lieu. Il aurait été ainsi plus aisé de récupérer les données.

Avis personnel

Le projet de reconstitution du ghetto de Brest-Litovsk est absolument passionnant. Mélangeant histoire et informatique, ce projet est un parfait exemple du domaine des Humanités Numériques.

Le projet de recherche sur le ghetto de Brest-Litovsk est remarquable à plusieurs égards. Il s'agit d'une entreprise multidisciplinaire qui vise à reconstruire l'histoire des Juifs ayant vécu dans ce ghetto pendant la Seconde Guerre mondiale. Ce projet cherche à donner une voix individuelle à chaque victime en retrouvant et en reconstituant leur identité, leur expérience et leur histoire personnelle.

L'approche micro-historique est vraiment intéressante car elle permet de dépasser les simples statistiques pour se concentrer sur les destins individuels. En considérant le ghetto non seulement comme un lieu géographique mais aussi comme un espace

social, le projet offre une compréhension plus profonde de la vie quotidienne et des interactions au sein de cette communauté.

La dimension mémorielle et pédagogique du projet est également cruciale. En mettant à disposition du grand public une base de données visuelle et interactive, le projet garantit que les souvenirs des victimes ne seront pas oubliés avec le temps. De plus, il offre aux chercheurs la possibilité d'approfondir leur compréhension de l'Holocauste et de tirer des conclusions transversales sur l'organisation et la vie dans ce ghetto spécifique.

Le projet de recherche sur le ghetto de Brest-Litovsk est une initiative précieuse qui honore la mémoire des victimes de la Shoah tout en enrichissant notre compréhension de cette période sombre de l'histoire. C'est un exemple inspirant de la façon dont la recherche académique peut contribuer à la préservation de la mémoire collective et à la transmission de l'histoire aux générations futures. Je suis très fière d’avoir pu travailler sur ce sujet.

Annexes

Annexe 1 – Le projet

Interroger un corpus avec XQuery

Julia Roger

julia.roger@unicaen.fr

Contexte Le Pôle Document Numérique est partenaire d'un programme de recherche intitulé « Le Ghetto de Brest-Litovsk » visant à étudier le ghetto de Brest-Litovsk (1941-1942) et porté par Boris Czerny (ERLIS, Unicaen). Ce projet s'inscrit à l'intersection de plusieurs champs et méthodes de recherche : études des civilisations, histoire et micro-histoire, sociologie, sciences du numérique ainsi que géographie. Il a pour ambition de redonner une identité aux Juifs du ghetto de la ville de Brest-Litovsk et de recomposer la géographie physique de l'espace où ils furent contraints de résider. Le projet repose sur deux principes essentiels :

1. À l'instar de l'historien de la Shoah, Serge Klarsfeld, nous considérons qu'il est vital de dépasser une approche globalisante focalisée sur le nombre, afin de retracer la tragédie comme étant celle d'une victime, plus une victime, plus une victime, afin que soient restitués à chacune d'entre elles son état civil, son itinéraire, sa dignité.

2. Cette démarche micro-historique est associée à une approche géographique du ghetto en tant qu'espace physique. Le ghetto est considéré à la fois comme un lieu autonome avec ses propres lois de fonctionnement et comme un espace hétéronome en relation avec le monde extérieur et régi en partie par lui : la ville, les ateliers et chantiers de travaux forcés, les environs, fermes, villages proches, villes et autres ghettos et, éventuellement, des lieux plus éloignés.

du grand public une base de données permettant de visualiser un ghetto et les membres de sa population afin de conserver dans le temps le caractère individuel de chacune des victimes de la Shoah à Brest-Litovsk.

Il s'appuie sur des méthodes de structuration de données textuelles (XML), d'interrogation de données (XQuery) et de mapping cartographique.

№ p/p	nom, prénom	Document	contenu
1.	Daiksel Abram (m)	Formulaire d'obtention d'une carte d'identité	Adresse – Batorego 41-1. Parents- père Szmul et mère Chana. Date et lieu de naissance– 1870 à Brest. Profession – cordonnier. N° de la carte d'identité délivrée– 8996. Document attestant de l'identité – passeport soviétique délivré à Brest en 1940. Statut de résidence– résident permanent. Document rédigé le– 25.11.1941. Photographie – oui. Empreinte digitale– oui. Signature personnelle– oui (en russe).

```
<person xml:id="d_p0" sex="m">
   <!--Daiksel Abram#m#-->
   <persName>
      <surname>Daiksel</surname>
      <forename>Abram</forename>
   </persName>
   <ab type="document">Formulaire d'obtention d'une carte d'identité </ab>
   <residence>
      <placeName type="street">Batorego<num n="41"><!--41-1-->41-1</num>
      </placeName>
   </residence>
   <note type="parents">père Szmul et mère Chana.</note>
   <note type="date_et_lieu_naissance">1870 à Brest.</note>
   <!--1870 à Brest.-->
   <occupation>cordonnier.</occupation>
   <idno>8996.</idno>
   <note type="based_document">passeport soviétique délivré à Brest en 1940.</note>
   <note type="origine">résident permanent.</note>
   <note type="date_creation_document">25.11.1941.</note>
   <note type="avec_photographie">oui.</note>
   <note type="avec_empreinte_digitale">oui.</note>
   <note type="avec_signature_individuelle">oui (en russe).</note>
```

Problématique et/ou attendus Le travail consistera à formuler différentes requêtes en XQuery exploitant la structuration faite en XML TEI des documents d'archives réunis dans le cadre du programme de recherche.

Ces documents archivistiques sont les 17 000 protocoles d'enregistrement des juifs par les SS à leur arrivée dans le ghetto. Ces protocoles sont constitués d'un certain nombre de rubriques qui donnent des informations socio-démographiques sur chaque juif habitant le ghetto.

Ces documents ont été numérisés, transcrits puis encodés selon les standards en vigueur dans le domaine des Humanités numériques. L'encodage XML de l'information délivrée par ces protocoles est donc requêtable et permettra de faire des analyses croisées de plusieurs protocoles. Par exemple, l'extraction de toutes les professions (<occupation>) croisées avec le lieu de résidence (<residence>) des titulaires de protocoles, permettra d'étudier les tâches auxquelles étaient assignés les juifs par les SS et leur répartition dans le ghetto.
Un autre exemple : l'extraction des dates de naissance (<date_naissance>) et des sexes (<sex>) des habitants du ghetto permettra de dresser une pyramide des âges.

Toutes ces informations permettront aux chercheurs impliqués dans le programme de recherche d'étayer des hypothèses et de tirer des conclusions transversales sur l'organisation de ce ghetto encore méconnu par les études sur l'holocauste.

Le dossier de travail remis à l'étudiant sera constitué :

- des fichiers XML d'un échantillon de ces protocoles ;
- de consignes avec le type d'informations à rechercher et à formuler en XQuery.

Livrables L'étudiant proposera d'une part la formulation de ces requêtes et d'autre part, les résultats de ces requêtes. Un commentaire sur la méthode de travail utilisée et les éventuelles difficultés rencontrées constituera également une partie du projet tutoré soumise à l'évaluation.

Annexe 2 – Exemple d'unité familiale dans le ghetto de Lodz

Recherche du nom de famille RUBLACH
(code DM 978500 ou 978400)
Nombre de résultats : 17

Nom	Jeune fille/Autres noms de famille / État civil	Né/Âge / Genre	Résidence / Adresse	Maison de la rue du ghetto / Adresse suivante	Date d'enregistrement / Taper	Déporté/ Type / Transport/ Destination	Décédé / Lieu	Remarques / Profession
RUBLACH, Haïm		20/02/1882 M	Lodz, Pologne Allemagne	Storchen Gasse 9 Appartement 6		25/03/1942 / AUSG TR25		Kaufmann
RUBLACH, Haïm		20/02/1882 M	Lodz, Pologne	Storchen Gasse 9 Appartement 6				
RUBLACH, Chaja		1922 F	Lodz, Pologne Tal Weg 14	Buchbinder Strasse 40 Appartement 20		24/03/1943		Schneider
RUBLACH, Chaja		26/02/1921 Âge : 23 ans F	Lodz, Pologne Marché Alt	Storchen Gasse 9 Appartement 4			Décédé le 16/04/1944 Ghetto de Lodz	Arbeiterin
RUBLACH, Chaja		1922 F	Lodz, Pologne Tal Weg 14	Buchbinder Strasse 40 Appartement 20		10/06/1944 / AG		Schneiderin
RUBLACH, Dyna		20/12/1905 F	Lodz, Pologne Muhl Gasse 2	Cranach Strasse 13 Appartement 24				Schneider
RUBLACH, Hinda Jochwet		20/04/1925 F	Lodz, Pologne Storchen Gasse 9	Appartement Hamburger Strasse 5	13/04/1944 FR			Arbitre
RUBLACH, Ita		01/07/1924 F	Lodz, Pologne Storchen Gasse 9	Appartement Hamburger Strasse 5	13/04/1944 FR			Farbe de strump
RUBLACH, Ita		01/07/1924 F	Lodz, Pologne Storchen Gasse 9	Storchen Gasse 9 Appartement 4 Hamburger Strasse 5	04/07/1944 ABG			Arbitre
RUBLACH, Joseph Natan	Enfant	11/02/1939 M	Lodz, Pologne Muhl Gasse 2	Cranach Strasse 13 Appartement 24				
RUBLACH, Liba Ides	Enfant	02/03/1939 F	Lodz, Pologne Storchen Gasse 7	Storchen Gasse 9 Appartement 4		19/09/1942 / AUSG		
RUBLACH, Luba		28/10/1899 F	Lodz, Pologne Allemagne	Storchen Gasse 9 Appartement 6		25/03/1942 / AUSG TR25		Maison-Fausse
RUBLACH, Mordka Icek		14/11/1884 Âge : 59 ans M	Lodz, Pologne Storchen Gasse 7	Storchen Gasse 9 Appartement 4			Décédé le 10/08/1943 Ghetto de Lodz	Arbitre
RUBLACH, Mordka Idel		14/11/1884 Âge : 59 ans M	Lodz, Pologne Storchen Gasse 9	Storchen Gasse 9 Appartement 4			Décédé le 10/08/1943 Ghetto de Lodz	Arbitre

Annexe 3 - Bibliographie

Sources écrites :

BOULANGER Thierry, *XML par la pratique - Bases indispensables, Concepts et cas pratiques* (3ième édition), 2015, Editions ENI

LE MAITRE Jacques, BRUNO Emmanuel, BASES DE DONNÉES - XQuery pour interroger des données XML - Eléments du langage et mise en œuvre - Cours et exercices corrigés , 2013, Ellipses

Sources internet :

Site de documentation des expressions régulières XQuery : https://isolution.pro/fr/t/xquery/xquery-regular-expressions/xquery-expressions-regulieres

Site de documentation BaseX : https://fr.wikipedia.org/wiki/BaseX

Site de documentation XQuery : https://www.w3schools.com/xml/xquery_syntax.asp

Site de la base de données centrale des noms des victimes de la Shoah : https://www.yadvashem.org/fr/archives/la-salle-des-noms/la-base-de-donnees.html

Site du projet du ghetto de Brest-Litovsk : https://ghettobrest.hypotheses.org/

Site du projet du ghetto de Lodz : https://www.jewishgen.org/databases/poland/lodzghetto.html#P2.3