Rapport de Projet

Base de Données

LAGET Antony
SACCO Aubin

Introduction

Dans ce TP nous allons développer une application java qui repose sur les datastores MongoDB et Neo4J. Nous allons manipuler des données scientifiques, initialement contenues dans un fichier et transférées dans MongoDB.

Nous effectuerons des traitements sur ces données pour ensuite les insérer dans Neo4J. A l'aide de ces deux datastores contenant des données complémentaires, nous pourrons développer des fonctions de recherches avancées.

Ce rapport présente les parties du code implémentant les fonctions demandées dans l'énoncé et les sorties de ces dernières.

1 - Exécution de Neo4J & de MongoDB

Driver Neo4J et MongoDB importés.

2 - Présentation du fichier de données

Lu.

3 - Mise en place des données

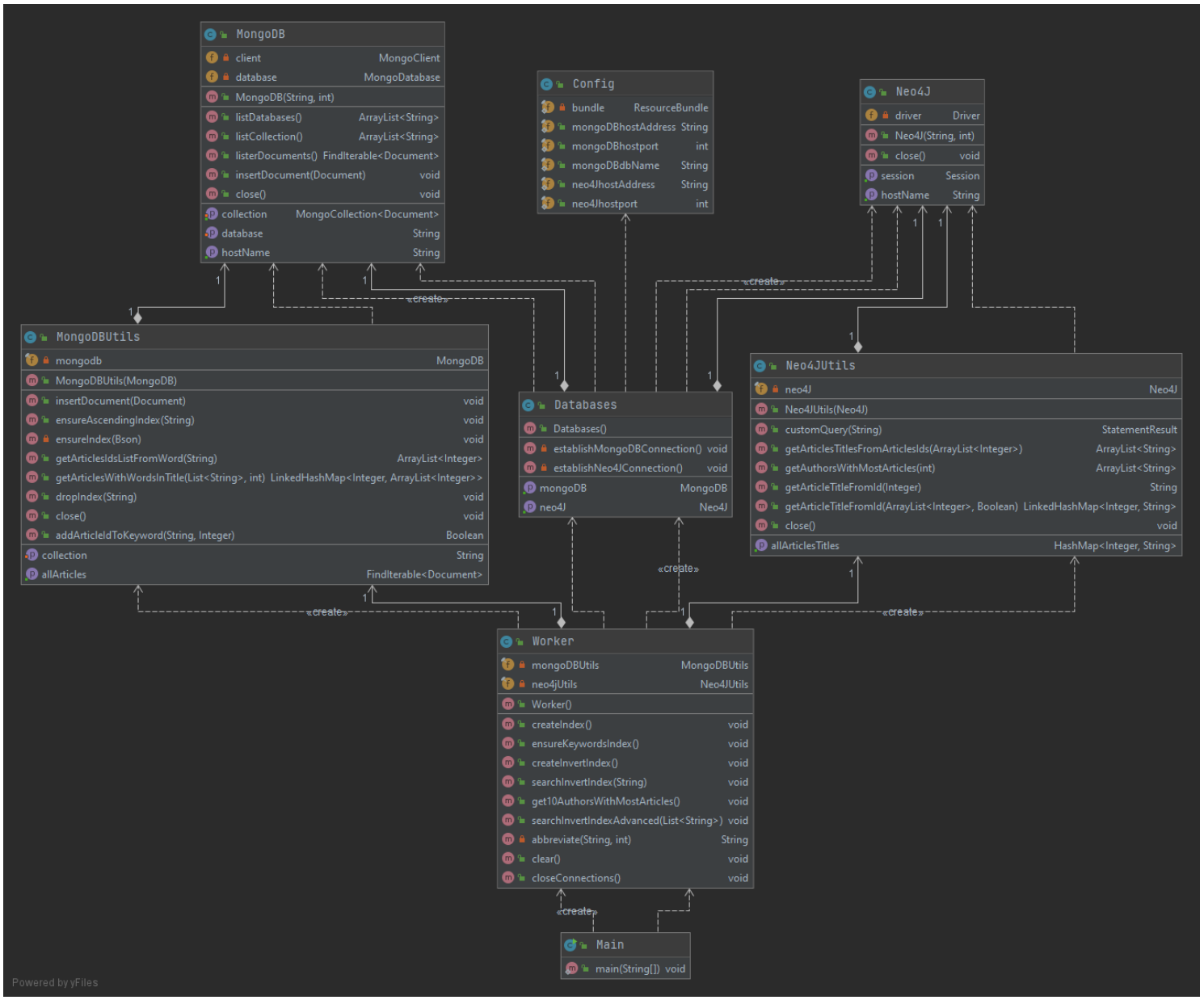
Exécution du script shell fourni sur le serveur de base de données pour ajouter les données à la base Neo4J.

3 - Développement de l'application Java

Nous allons présenter les éléments de code pertinents pour la réponse aux questions demandées. Les appels aux fonctions "utilitaires" sont accessibles dans les sources du projet jointes à ce document ou sur https://github.com/AntAub/Projet-S9-BDD

3.0 - Mise en place

3.0.1 - Architecture



3.0.2 - Main Java

Le worker contient les fonctions qui réalisent le besoin des questions posées.
Lors de son instanciation, il établit la connexion aux bases MongoDB et Neo4J puis supprime les indexes existants dans MongoDB.

```
package projet.s3;

import projet.s3.bdd.Worker;
import java.util.ArrayList;

public class Main {

   public static void main(String[] args) {

        try {
            /* Création des connexions */
            Worker worker = new Worker();

            /* 3.1. Mettre en place un datastore MongoDB */
            worker.createIndex();

            /* 3.2. Mettre en place index sur le tableau de mot clés dans MongoDB */
            worker.ensureKeywordsIndex();

            /* 3.3. Mise en place d'une « structure miroir » sur MongoDB */
            worker.createInvertIndex();

            /* 3.4. Recherche de documents */
            worker.searchInvertIndex("with");

            /* 3.5. Auteurs ayant écrit le plus d'articles */
            worker.get10AuthorsWithMostArticles();

            /* 3.6. Recherche de documents avancée */
            //TODO Vérifier que c'est la bonne réponse attendue
            worker.searchInvertIndexAdvanced(new ArrayList<>() {{
                add("new");
                add("with");
                add("systems");
            }});

            /* Fermeture des connexions */
            worker.closeConnections();

        }
        catch (Exception e){
            e.printStackTrace();
            System.err.println("Erreur lors de l'exécution : "  + e.getMessage());
        }
    }
}
```

3.0.3 - Trace d'execution

L'exécution du programme laisse la trace suivante qui atteste de sa bonne exécution et de son bon déroulement.

Les énoncés de chaque question et les retours de données importants sont mis en gras.

```
------------------------------------------------------------
Tentative de connexion à cloud.antonylaget.com ...
Connexion établie à MongoDB cloud.antonylaget.com
Base de données : dbDocuments sur cloud.antonylaget.com
Connexion établie à Neo4J : cloud.antonylaget.com
------------------------------------------------------------
Collection sélectionnée : dbDocuments.indexInverse
Index indexInverse supprimé
Collection sélectionnée : dbDocuments.index
Index index supprimé
------------------------------------------------------------
-- 3.1. Création de l'index des titres d'article dans MongoDB ---
Collection sélectionnée : dbDocuments.index
Nombre de titres d'articles indéxés : 999
------------------------------------------------------------
-- 3.2. Création index sur le tableau de mots-clé dans MongoDB --
------------------------------------------------------------
-- 3.3. Création de l'index inversé des mots-clé dans MongoDB ---
Collection sélectionnée : dbDocuments.index
-- Récupération des documents dans MongoDB ----------------------
Collection sélectionnée : dbDocuments.indexInverse
Nombre de mots-clé indéxés : 2846
------------------------------------------------------------
-- 3.4. Recherche d'un mot dans l'index inversé -----------------
Collection sélectionnée : dbDocuments.indexInverse
Liste des articles contenant le mot " with" (72 résultat(s)) :
-- Résultat 01 : 120 Hz low cross-talk stereoscopic display wi...
-- Résultat 02 : A Balanced Search Tree with  (1) Worst-case U...
-- Résultat 03 : A Formal Identification between Tuples and Li...
-- Résultat 04 : A Recursively-Adjusting Co-allocation scheme ...
-- Résultat 05 : A comparison of pebble tree transducers with ...
-- Résultat 06 : A distributed energy-efficient clustering alg...
-- Résultat 07 : A framework for integrating syntax, semantics...
-- Résultat 08 : A fuzzy matchmaking approach for Semantic Web...
-- Résultat 09 : A new compensation pixel circuit with all-p-t...
-- Résultat 10 : A nonrepudiable threshold multi-proxy multi-s...
-- Résultat 11 : A review of Multi-Agent Systems techniques, w...
-- Résultat 12 : Accepting splicing systems with permitting an...
-- Résultat 13 : Analysis of dual-hop and multiple relays coop...
-- Résultat 14 : Applying different control approaches for res...
-- Résultat 15 : Applying different control approaches for res...
-- Résultat 16 : Association-Based Active Access Control model...
-- Résultat 17 : Automatic visual inspection: An approach with...
-- Résultat 18 : Caching Support for Skyline Query Processing ...
-- Résultat 19 : Connecting business models with service platf...
-- Résultat 20 : Coping with TiVo: Opportunities of the networ...
-- Résultat 21 : Coupled Matrix Factorization with Sparse Fact...
```

```
-- Résultat 22 : Crawling Hidden Objects with kNN Queries.
-- Résultat 23 : Declarative Programs with Implicit Implications.
-- Résultat 24 : Design of reflective projection lens with Zer...
-- Résultat 25 : Developing new products with knowledge manage...
-- Résultat 26 : E-Commerce Training with Virtual Commerce Sim...
-- Résultat 27 : Elastic grid resource provisioning with WoBin...
-- Résultat 28 : Electrochemically-switchable emission and abs...
-- Résultat 29 : Empirical comparison of image retrieval color...
-- Résultat 30 : Energy efficient scheduling of real-time task...
-- Résultat 31 : Enhanced properties of organic electrolumines...
-- Résultat 32 : Experience of parallel AI programming with pa...
-- Résultat 33 : Experiments with equivalent differentiated se...
-- Résultat 34 : Fabrication and chromatic characteristics of ...
-- Résultat 35 : Finding the  Most Vital Edges with Respect to...
-- Résultat 36 : General quantitative specification theories w...
-- Résultat 37 : High colour rendering index white organic lig...
-- Résultat 38 : High reliable real-time bandwidth scheduling ...
-- Résultat 39 : Improvement in performance of GaN-based light...
-- Résultat 40 : Improving the efficiency of virtual channels ...
-- Résultat 41 : Integrating cross-sectional imaging based rev...
-- Résultat 42 : Ising spin neural networks with spatial struc...
-- Résultat 43 : New and efficient conditional e-payment syste...
-- Résultat 44 : New scoring formula to rank hypervisors' perf...
-- Résultat 45 : On Grammar Forms with Terminal Context.
-- Résultat 46 : On the closure of pattern expressions languag...
-- Résultat 47 : Order statistics on a linear array with a rec...
-- Résultat 48 : Performance improvement of third-party logist...
-- Résultat 49 : Permuting Data with the Omega Network.
-- Résultat 50 : Petri nets with name creation for transient s...
-- Résultat 51 : Probe Minimization by Schedule Optimization: ...
-- Résultat 52 : Product Life-Cycle Metadata Modeling and Its ...
-- Résultat 53 : Program reversals for evolutions with non-uni...
-- Résultat 54 : Querying Datalog Programs with Temporal Logic.
-- Résultat 55 : Querying Sequence Databases with Transducers.
-- Résultat 56 : Rarefied gas flow computational with a 3D uns...
-- Résultat 57 : Rescheduling for reliable job completion with...
-- Résultat 58 : Right-Protected Data Publishing with Provable...
-- Résultat 59 : Satisfaction with interpersonal and internet ...
-- Résultat 60 : Scheduling tasks of a parallel program in two...
-- Résultat 61 : Smart cyber society: Integration of capillary...
-- Résultat 62 : Stable Duplicate-Key Extraction with Optimal ...
-- Résultat 63 : Temperature-dependent electrical and dielectr...
-- Résultat 64 : The impact of founders' academic experiences ...
-- Résultat 65 : Trading polarizations for labels in P systems...
-- Résultat 66 : Unsteady flow in a 2D elastic tube with the L...
-- Résultat 67 : User transparent data and task parallel multi...
-- Résultat 68 : Verification of heap manipulating programs wi...
-- Résultat 69 : Versatile energy recovery circuit for driving...
-- Résultat 70 : Viewer experience with stereoscopic 3D televi...
-- Résultat 71 : Visual acuity and contrast sensitivity screen...
-- Résultat 72 : White organic light-emitting devices with non...
------------------------------------------------------------
Collection sélectionnée : dbDocuments.indexInverse
-- 3.5. Auteurs ayant écrit le plus d'articles ------------------
Liste des 10 auteurs ayant écrit le plus d'articles :
```

```
-- Auteur 01 : 7 - Joost Engelfriet
-- Auteur 02 : 4 - Ching-Hsien Hsu
-- Auteur 03 : 4 - Péter Kacsuk
-- Auteur 04 : 3 - Albert Y. Zomaya
-- Auteur 05 : 3 - Amir Pnueli
-- Auteur 06 : 3 - Dunbing Tang
-- Auteur 07 : 3 - Frank J. Seinstra
-- Auteur 08 : 3 - Henri E. Bal
-- Auteur 09 : 3 - Joanna Kolodziej
-- Auteur 10 : 3 - Kenli Li
----------------------------------------------------------------
Collection sélectionnée : dbDocuments.indexInverse
-- 3.6. Recherche de documents avancée -------------------------
Liste des articles contenant les mots recherchés :
5259 New and efficient conditional e-payment systems with tr... 3
3293 A new compensation pixel circuit with all-p-type TFTs f... 2
4917 A performance comparison of current HPC systems: Blue G... 2
4459 A review of Multi-Agent Systems techniques, with applic... 2
3114 Accepting splicing systems with permitting and forbiddi... 2
6154 Applying different control approaches for resources wit... 2
4140 Developing new products with knowledge management metho... 2
3626 New perspectives for the future interoperable enterpris... 2
4813 New scoring formula to rank hypervisors' performance co... 2
3089 Trading polarizations for labels in P systems with acti... 2
----------------------------------------------------------------
Fermeture de la connexion à la base Mongo (cloud.antonylaget.com)
Fermeture de la connexion à la base Neo4J (cloud.antonylaget.com)
----------------------------------------------------------------

Process finished with exit code 0
```

3.1 - Mettre en place un datastore MongoDB

Worker.java : createIndex()

La méthode "getAllArticlesTitles()" est détaillée à la page suivante.

```
/**
* Crée l'index dans la base MongoDB à partir des articles de la base Neo4J
*/
public void createIndex() {
   System.out.println("-- 3.1. Création de l'index des titres d'article dans
MongoDB ---");

   this.mongoDBUtils.setCollection("index");

   /* Récupération des articles dans la base Neo4J */
   HashMap<Integer, String> titles = this.neo4jUtils.getAllArticlesTitles();

   AtomicInteger nbArticle = new AtomicInteger(0);

   titles.forEach((id, title) -> {

       /* Création de la liste des mots clés */
       StringTokenizer tokenizedTitle = new StringTokenizer(title.toLowerCase(),
                                                            " ,'-:;()+[]{}?!.");
       BasicBSONList bsonList = new BasicBSONList();

       while (tokenizedTitle.hasMoreTokens()) {
           /* Mot suivant et retrait des espaces innutiles */
           String word = tokenizedTitle.nextToken().trim();
           bsonList.add(word);
       }

       /* Insertion dans la collection MongoDB */
       this.mongoDBUtils.insertDocument(new Document("idDocument",
                                              id).append("motsCles", bsonList));
       System.out.print("Nombre de titres d'articles indéxés : " +
                                           nbArticle.incrementAndGet() + "\r");
   });

   System.out.print("\n");

System.out.println("-----------------------------------------------------------
--");
}
```

Neo4JUtils.java : getAllArticlesTitles()

```
/**
* Retour les tous articles (id, titre)
* @return Articles (id, titre)
*/
public HashMap<Integer, String> getAllArticlesTitles() {

   StatementResult queryResult = customQuery("MATCH (n:Article) RETURN n.titre,
id(n) as id");

   HashMap<Integer, String> titles = new HashMap<>();

   queryResult.forEachRemaining(record ->
           titles.put(record.get("id").asInt(), record.get("n.titre").asString())
   );

   return titles;
}
```

3.2 - Mettre en place un index sur le tableau de motsClés dans MongoDB

Worker : ensureKeywordsIndex()

```
/**
* Met en place un index par ordre croissant sur les valeurs du tableau mots-clé
*/
public void ensureKeywordsIndex() {

   System.out.println("-- 3.2. Création index sur le tableau de mots-clé dans
MongoDB --");

   /* Création de l'index sur valeurs du tableau mots-clé */
   this.mongoDBUtils.ensureAscendingIndex("motsCles");


System.out.println("-----------------------------------------------------------------
--");
}
```

MongoDBUtils.java : ensureAscendingIndex()

```
/**
* Met en place un index par ordre croissant sur les valeurs des champs choisis
* @param fieldNames Nom des champs sur lequels ils faut crée un Index
*/
public void ensureAscendingIndex(String fieldNames) {

   /* ensureIndex() deprecated : utilisation de createIndex() */
   this.ensureIndex(Indexes.ascending(fieldNames));
}
```

MongoDBUtils.java : ensureIndex()

```
/**
* Crée un index sur les champs choisis
* @param order Ordre de l'index Indexes.ascending(fielNames)
*/
private void ensureIndex(Bson order) {

   this.mongodb.getCollection().createIndex(order);
}
```

3.3 - Mise en place d'une « structure miroir » sur MongoDB

Worker : createInvertIndex()

Les méthodes "getAllArticles()" et "addArticleIdToKeyword()" sont détaillées à la page suivante.
Le @SuppressWarning est dû au cast en ArrayList<String>.

```
/**
* Crée l'index inversé dans la base MongoDB à partir de la collection index
*/
@SuppressWarnings("unchecked")
public void createInvertIndex() {

   System.out.println("-- 3.3. Création de l'index inversé des mots-clé dans
MongoDB ---");

   /* Récupération des documents dans MongoDB */
   this.mongoDBUtils.setCollection("index");
   System.out.println("-- Récupération des documents dans MongoDB
----------------------");

   FindIterable<Document> articles = this.mongoDBUtils.getAllArticles();

   this.mongoDBUtils.setCollection("indexInverse");

   AtomicInteger nbKeyword = new AtomicInteger(0);

   for (Document article : articles) {

       ArrayList<String> keywords = (ArrayList<String>) article.get("motsCles");

       keywords.forEach(keyword ->{

           if(this.mongoDBUtils.addArticleIdToKeyword(keyword,
                                                (Integer)article.get("idDocument")))
               System.out.print("Nombre de mots-clé indéxés : " +
                                                nbKeyword.incrementAndGet() + "\r");
       });
   }

   System.out.print("\n");

System.out.println("----------------------------------------------------------------
--");
}
```

MongoDBUtils.java : getAllArticles()

```
/**
* Retourne la totalités des articles en
* @return FindIterable<Document> List de documents
*/
public FindIterable<Document> getAllArticles() {

   return this.mongodb.getCollection().find();
}
```

MongoDBUtils.java : addArticleIdToKeyword()

```
/**
* Ajoute l'id d'article à un mot clé existant sinon elle crée le mot clé
* @param keyword Mot clé
* @param articleId Id de l'article associé au mot clé
* @return vrai si un nouveau mot clé est indexé, faux sinon
*/
@SuppressWarnings("unchecked")
public Boolean addArticleIdToKeyword(String keyword, Integer articleId) {

   Document keywordDocument = this.mongodb.getCollection()
                                       .find(Filters.eq("mot", keyword)).first();

   if(keywordDocument == null){

       ArrayList<Integer> articles = new ArrayList<>();
       articles.add(articleId);
       this.insertDocument(new Document("mot", keyword).append("documents",
                                                                articles));
       return true;
   }
   else{

       ((ArrayList<Integer>) keywordDocument.get("documents")).add(articleId);
       this.mongodb.getCollection().replaceOne(Filters.eq("mot",keyword),
                                                            keywordDocument);
       return false;
   }
}
```

3.4 - Recherche de documents

Worker : searchInvertIndex()

Les méthodes "getArticlesIdsListFromWord()" et "getArticlesTitlesFromArticlesIds()" sont détaillées à la page suivante.
"abbreviate()" réduit la longueur du titre affiché pour obtenir une trace propre.

```
/**
* Rechercher dans indexInverse un mot dans MongoDB et
* afficher par ordre alphabétique les titres des documents dans Neo4J
* @param word Mot à rechercher dans l'index inversé
*/
public void searchInvertIndex(String word) {

   System.out.println("-- 3.4. Recherche d'un mot dans l'index inversé
-----------------");

   this.mongoDBUtils.setCollection("indexInverse");

   ArrayList<Integer> articlesIdsList =
               this.mongoDBUtils.getArticlesIdsListFromWord(word);

   ArrayList<String> articlesTitlesList =
               this.neo4jUtils.getArticlesTitlesFromArticlesIds(articlesIdsList);

   AtomicInteger count = new AtomicInteger(0);

   System.out.println("Liste des articles contenant le mot \" " + word + "\" (" +
                articlesTitlesList.size() + " résultat(s)) : ");

   articlesTitlesList.forEach((title) ->
           System.out.println("-- Résultat " + String.format("%02d",
                    count.incrementAndGet()) + " : " + this.abbreviate(title, 48))
   );


System.out.println("----------------------------------------------------------------
--");
}
```

MongoDBUtils.java : getArticlesIdsListFromWord()

```java
/**
* Liste les articles où leur titre contient le mot recherché
* @param word Mot recherché
* @return Liste des articles où leur titre contient le mot recherché
*/
@SuppressWarnings("unchecked")
public ArrayList<Integer> getArticlesIdsListFromWord(String word) {

   Document document = this.mongodb.getCollection()
                                          .find(Filters.eq("mot", word)).first();

   return (ArrayList<Integer>) document.get("documents");
}
```

Neo4JUtils.java : getArticlesTitlesFromArticlesIds()

```java
/**
* Retour la liste des titres des articles dont l'id est passé en paramètre
* @param articlesIdsList Liste des Id's d'articles
* @return Liste des titres des articles
*/
public ArrayList<String> getArticlesTitlesFromArticlesIds(
                                             ArrayList<Integer> articlesIdsList) {

   StatementResult queryResult = this.customQuery("MATCH (n:Article) WHERE id(n) in "
                        + articlesIdsList + " RETURN n.titre ORDER BY n.titre ASC");

   ArrayList<String> titles = new ArrayList<>();

   queryResult.forEachRemaining(title ->
                                       titles.add(title.get("n.titre").asString()));

   return titles;
}
```

3.5 - Auteurs ayant écrit le plus d'articles

Worker : get10AuthorsWithMostArticles()

La méthode "getAuthorsWithMostArticles()" est détaillée à la page suivante.

```
/**
* Affiche, par ordre décroissant de nombre d'articles écrits puis
* par ordre croissant de nom les 10 auteurs ayant écrit le plus d'articles
*/
public void get10AuthorsWithMostArticles() {

   this.mongoDBUtils.setCollection("indexInverse");

   System.out.println("-- 3.5. Auteurs ayant écrit le plus d'articles
------------------");

   ArrayList<String> authorList = this.neo4jUtils.getAuthorsWithMostArticles(10);

   AtomicInteger count = new AtomicInteger(0);

   System.out.println("Liste des 10 auteurs ayant écrit le plus d'articles : ");

   authorList.forEach((author) ->
           System.out.println("-- Auteur " + String.format("%02d",
                                   count.incrementAndGet()) + " : " + author)
   );

System.out.println("----------------------------------------------------------------
--");
}
```

Neo4JUtils : getAuthorsWithMostArticles()

```
/**
* Retourne, par ordre décroissant de nombre d'articles écrits puis
* par ordre croissant de nom les auteurs ayant écrit le plus d'articles
* @param limit Nombre d'auteurs maximum à afficher
* @return Liste des auteurs
*/
public ArrayList<String> getAuthorsWithMostArticles(int limit) {

   StatementResult queryResult =
            this.customQuery("MATCH (n:Auteur)-[e:Ecrire]->(a:Article)
                              RETURN n.nom, count(a) as nbArticles
                              ORDER BY nbArticles DESC, n.nom ASC LIMIT " + limit);

   ArrayList<String> authors = new ArrayList<>();

   queryResult.forEachRemaining(record ->
           authors.add(record.get("nbArticles").asInt() + " - " +
                                                   record.get("n.nom").asString())
   );

   return authors;

}
```

3.6 - Recherche de documents avancée

Worker : searchInvertIndexAdvanced()

Les méthodes "getArticlesWithWordsInTitle()" et "getArticleTitleFromId()" sont détaillées à la page suivante.

```
/**
* Recherche dans indexInverse plusieurs mots et afficher par ordre alphabétique les
titres des documents dans Neo4J.
* Les documents sont triés par nombre décroissant de mots de la requête contenus dans
les documents.
* @param words Tablea des mots à rechercher
*/
public void searchInvertIndexAdvanced(List<String> words) {

   this.mongoDBUtils.setCollection("indexInverse");

   System.out.println("-- 3.6. Recherche de documents avancée
--------------------------");

   /* On récupère la liste des id d'articles triés par nombre d'occurence des mots */
   LinkedHashMap<Integer, ArrayList<Integer>> articleList =
                           this.mongoDBUtils.getArticlesWithWordsInTitle(words, 10);

   System.out.println("Liste des articles contenant les mots recherchés :");

   articleList.forEach((nbOccurences, listIdsArticles)-> {

       /* On récupère les titres des articles associé aux id d'articles par
         ordre alphapétique */
       LinkedHashMap<Integer, String> titles =
           this.neo4jUtils.getArticleTitleFromId(listIdsArticles, true);

       titles.forEach((idArticle, title) ->{

           /* On affiche l'id de l'article, son titre et le nombre d'occurance
           du mot*/
           System.out.println(this.abbreviate(idArticle + " " + title + " ", 63) +
                                                                   nbOccurences);
       });
   });

System.out.println("-----------------------------------------------------------------
--");
}
```

MongoDBUtils.java : getArticlesWithWordsInTitle()

```
/**
* Recupère la liste des IDs d'articles où les mots apparaissent ainsi que leur nombre
d'occurence dans chaque titre
* @param words Liste des mots à rechercher
* @param limit Nombre maximum de résultats
* @return liste des id d'articles trié par nombre d'occurances des mots cherchés dans
leur titre
*/
@SuppressWarnings("unchecked")
public LinkedHashMap<Integer, ArrayList<Integer>>
                        getArticlesWithWordsInTitle(List<String> words, int limit) {

    /*
    * On formate la liste des mots en ajoutant des guillemets pour permettre
    * la recherche et éviter les injections de code
    */
    ArrayList<String> wordsLocal = new ArrayList<>();
    for (String word : words)
        wordsLocal.add("'" + word + "'");

    /* On requête mongoDB */
    AggregateIterable<Document> documents =
            this.mongodb.getCollection().aggregate(java.util.Arrays.asList(
                    //Liste de id d'articles contenant les mots recherchés
                    Document.parse("{$match : { mot : {$in : "+wordsLocal+"}}}"),
                    //Tri par ordre alphabétique
                    Document.parse("{$sort :{mot : -1}}"),
                    //Liste des mots associés à l'id d'articles
                    Document.parse("{$unwind : \"$documents\" }"),
                    //On groupe par id d'article pour compter
                    //le nombre d'occurances du mot par article
                    Document.parse("{$group : { _id: \"$documents\",
                                                nbOccurances : { $sum: 1 }}}"),
                    //On tri par nombre d'occurence des mots dans le titre
                    Document.parse("{$sort :{ nbOccurances : -1 }}"),
                    //On limite au N premier article
                    Document.parse("{$limit : " + limit + "}"),
                    //On groupe par nombre d'occurance
                    Document.parse("{$group : { _id : \"$nbOccurances\",
                                        documents: { $addToSet : \"$_id\" }}}"),
                    //On tri par nombre décroissant d'occurance
                    Document.parse("{$sort :{ _id : -1}}")
            ));

    LinkedHashMap<Integer, ArrayList<Integer>> articleList = new LinkedHashMap<>();

    for (Document document : documents) {
        articleList.put((Integer) document.get("_id"),
                                    (ArrayList<Integer>) document.get("documents"));
    }

    return articleList;
}
```

Neo4JUtils : getArticleTitleFromId()

```
/**
* Récupère le titre d'un article à partir de son Id
* @param id Id de l'article
* @return Titre de l'article
*/
public String getArticleTitleFromId(Integer id) {

   return this.customQuery("MATCH (n:Article)
                            WHERE id(n) = "+ id +"
                            RETURN n.titre ORDER BY n.title ASC")
                        .next().get("n.titre").asString();
}

public LinkedHashMap<Integer, String> getArticleTitleFromId(ArrayList<Integer>
                                                            ids, Boolean ascOrder){

   String order = (ascOrder) ? "ASC" : "DESC";
   StatementResult result = this.customQuery("MATCH (n:Article)
                                              WHERE id(n) in " + ids + "
                                              RETURN id(n) as id, n.titre as title
                                              ORDER BY title " + order);

   LinkedHashMap<Integer, String> articleTitles = new LinkedHashMap<>();

   while (result.hasNext()) {
       Record record = result.next();
       articleTitles.put(record.get("id").asInt(), record.get("title").asString());
   }

   return articleTitles;
}
```

Conclusion

Grâce à TP nous avons pu voir comment mettre en œuvre des appels aux datastores Neo4J et MongoDB dans le cadre d'un projet Java en utilisant leur driver et les différentes fonctions qu'il propose.

Nous avons rencontré des difficultés pour mettre au point la fonction de recherche avancée car la consigne n'était pas assez explicite et le retour affiché dans le rapport ne correspondait pas au résultat demandé.

Nous avons essayé de prendre le temps d'architecturer le programme. Ainsi nous aimerions avoir un regard critique sur ce qui a été bien ou moins bien fait. Alors n''hésitez pas à nous faire un retour sur ce dernier si vous avez le temps.

FIN DU DOCUMENT