Citation Network Proposal

Group 2
MITIU14005 - Nguyen Hoang Minh
MITIU14006 - Chiem Thach Phat

Ho Chi Minh City, International University

December 23, 2014

Table of Contents

Table of Contents

Motivation

The world need a citation network for patents to:

- Analyze/Predict competitors trend
- Look up whether there is a similar patent or not to submit a new one
- Find who invent most patents
- Check uniqueness of company's products
- Find which products or which patents are most cited by other

Table of Contents

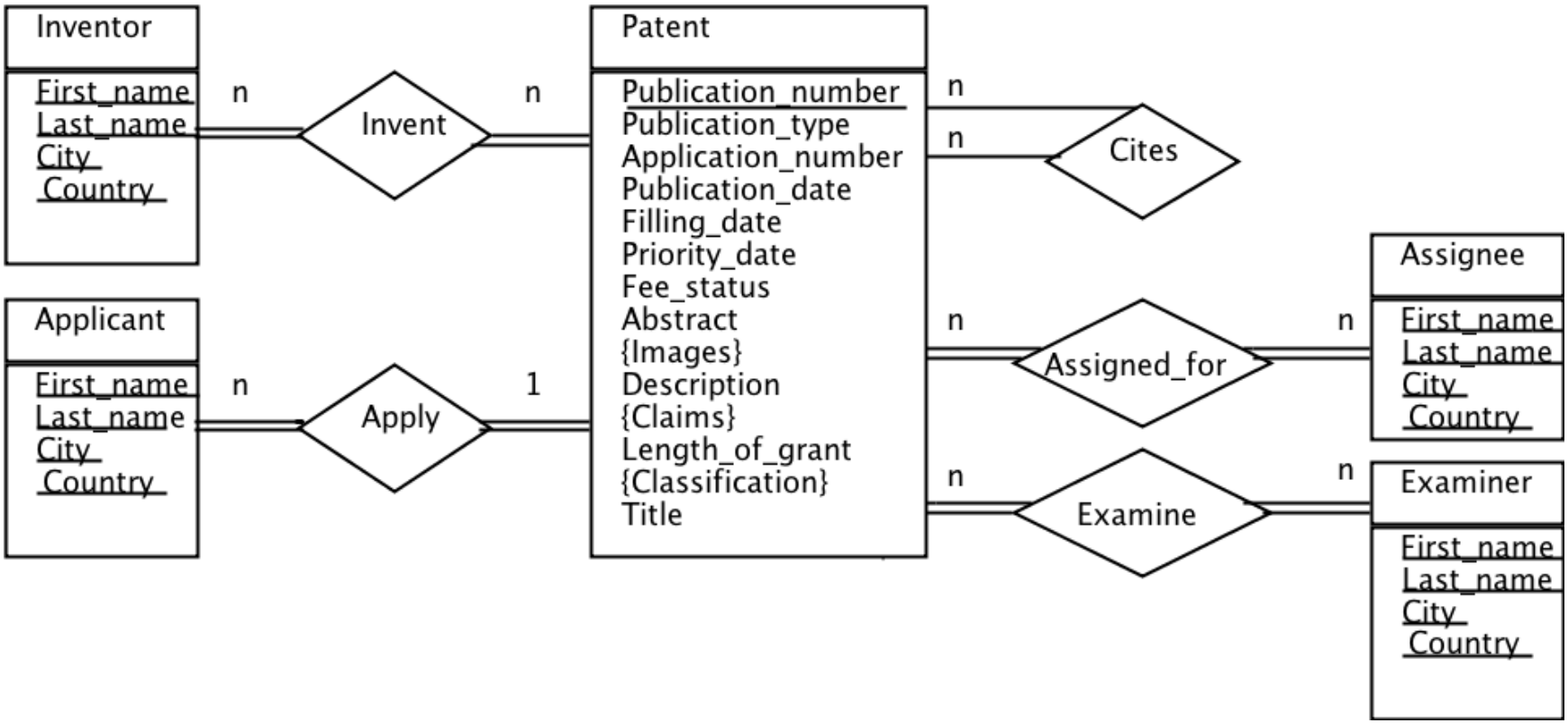
Scenario

We want to build a citation network for patents with these constraints:

- Able to store and analyze patents from USPTO
- A patent contains: an unique Publication Number, Publication Type, Application Number, Publication Date, Filling Date, Priority Date, Fee Status, Title, Length of Grant, Classification and some of its contents include Abstract, Images, Description and Claims
- A Patent may or may not cite/be cited other patents
- A Patent must be invented/applied/examined by or assigned for at least one Inventor, Applicant, Examiner or Assignee
- Inventor, Applicant, Examiner and Assignee are identified by first name, last name, city and country

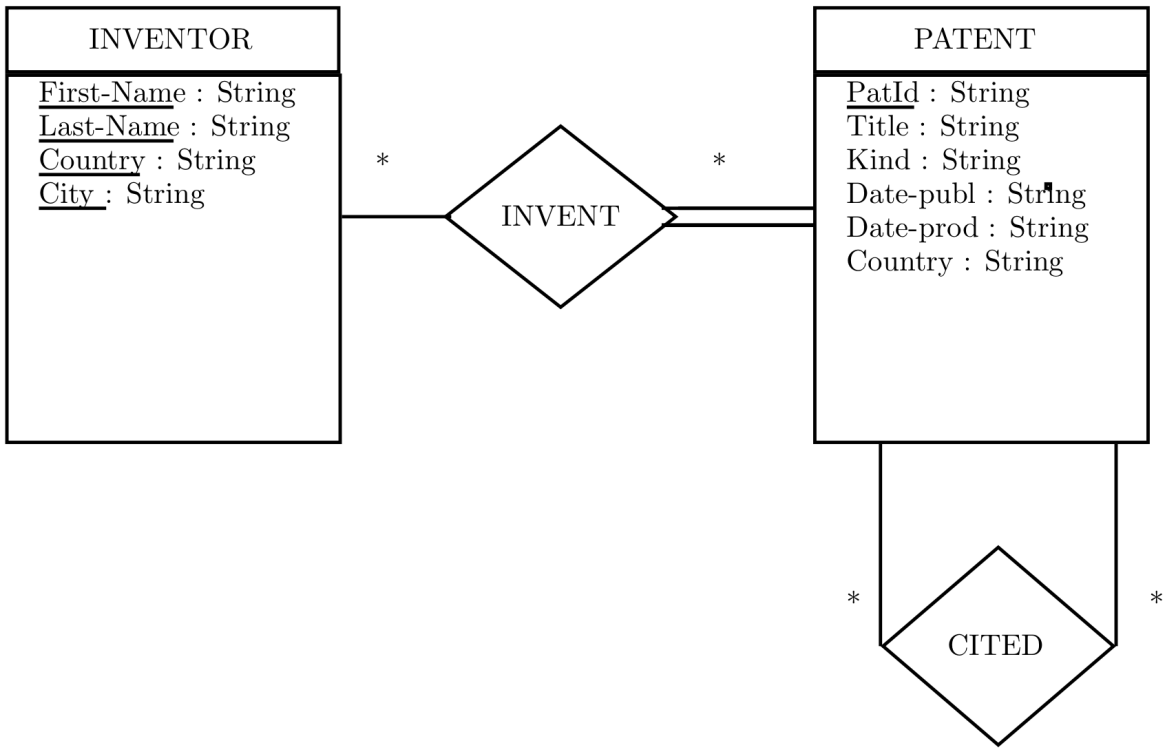
Conceptual Model

Based on the scenario, we have design this conceptual model.



Implemented Conceptual Model

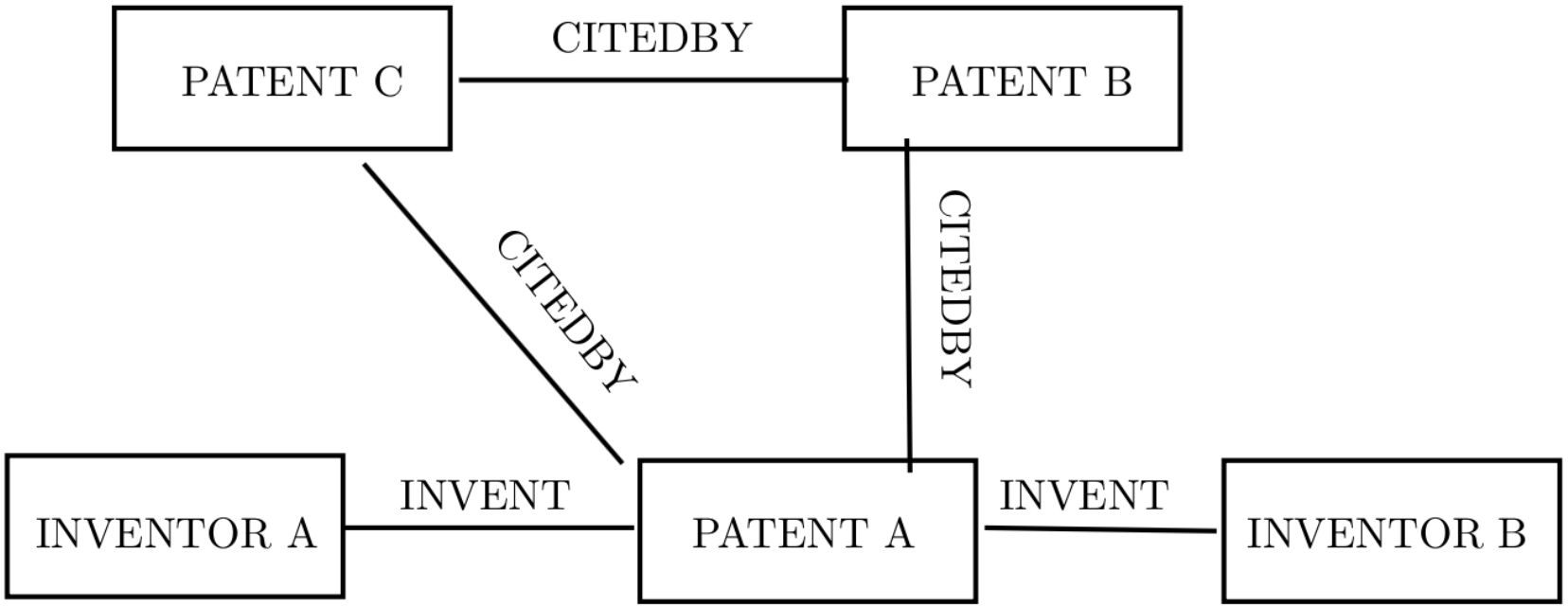
Due to time constraint, we use this conceptual model to implement the citation network.



Logical Model - Neo4j

Neo4j use the concepts of **Nodes**, **Properties** and **Relationships** as shown in the picture below, where:

- Nodes: PATENT, INVENTOR
- Properties: Nodes and Relationships attributes, such as Patent_No or Cited_date
- Relationships: CITEDBY, INVENT



Logical Model - Neo4j (Cont.)

Pros:

- Can cope with complex and highly connected data
- Offers more accurate ways to determine the relationship between each entity

Cons:

- Slow with complex search
- Server is not supported for visualizing more than 1000 rows

Logical Model - MongoDB

While Neo4j stores data in Nodes, Properties and Relationships, **MongoDB stores data as a BSON file**.

```
patent_schema = {
    kind : string,
    date-published : string,
    number-of-claims : integer,
    country : string,
    patid : string,
    main-classification : string,
    title : string,
    app-number : string,
    citations : [ { patid : string }, {
    patid : string }]
    date-produced : string,
    inventors:[
        { city : string, last-name :
    string, country : string, first-name :
    string },
    { city: string, last-name : string,
    country : string, first-name : string }
     ]
}

inventor_schema={
    city : string,
    last-name : string,
    country : string,
    first-name : string
}
```

Logical Model - MongoDB (Cont.)

Pros:

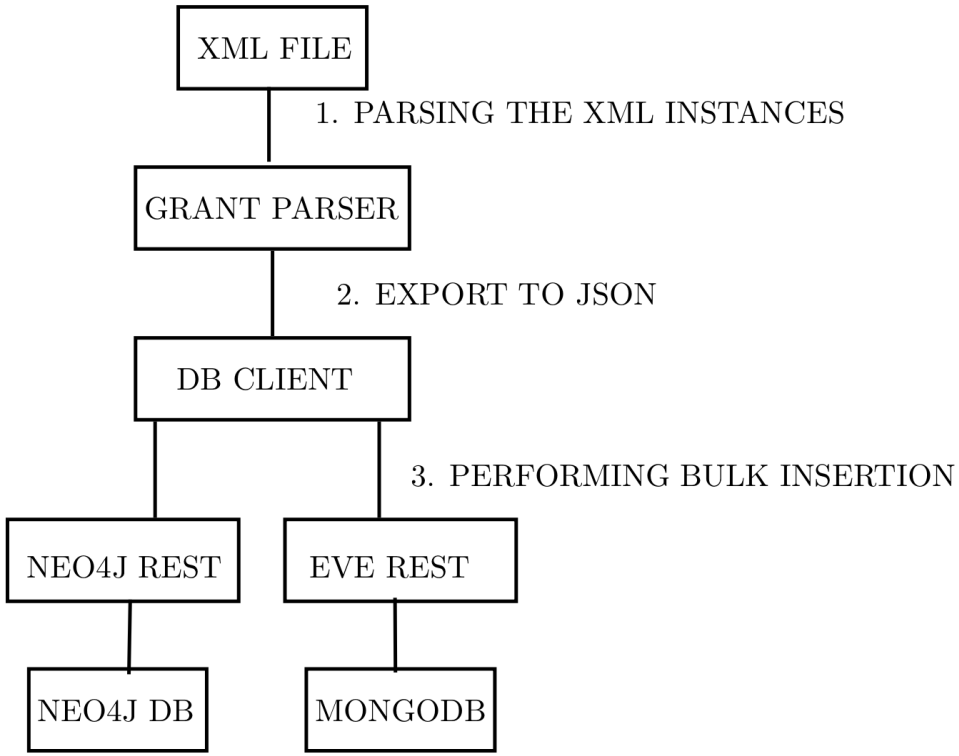
- Documents (i.e. objects) can be defined with native data types that can be seen in many programming languages. E.x: Date, String, Integer
- Embedded documents and arrays reduce need for expensive joins
- Dynamic schema supports fluent polymorphism

Cons:

- Does not support identify duplication

CINAS Structure

Our CINAS will follow the structure below, starts from the Patent XML File and ends at Neo4j DB and MongoDB



Grant Parser Module

Due to the enormous size of the Patent Bulk Download File, we will use **SAX method** instead of DOM method to **process the XML file**.

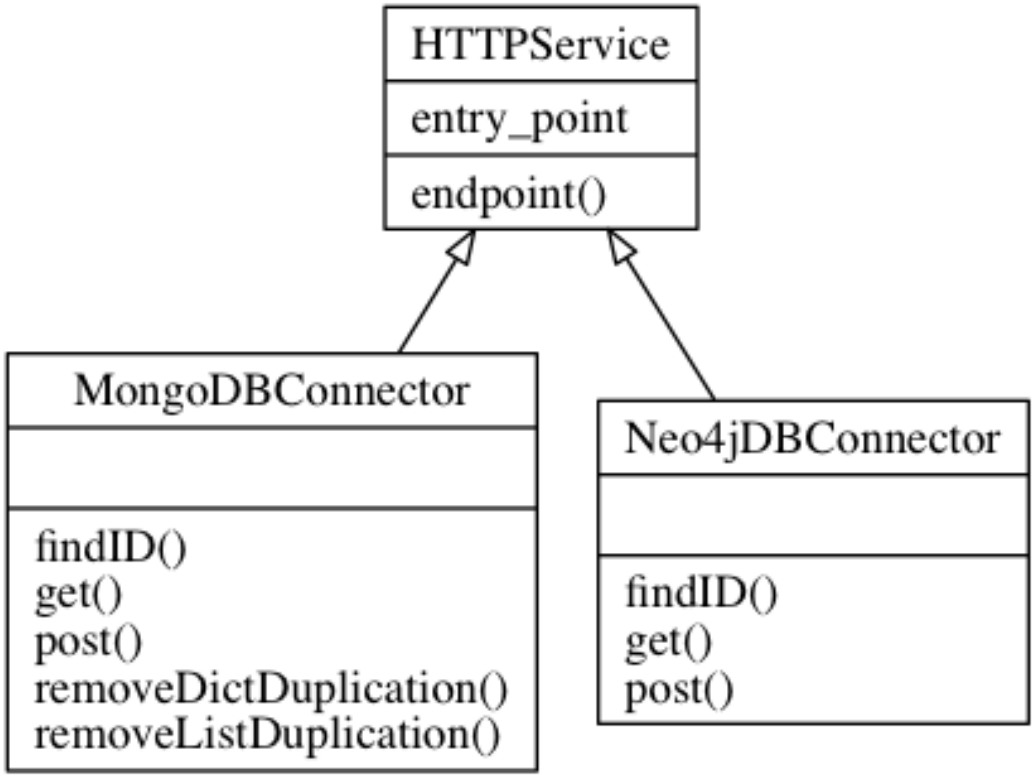
Pseudocode for **Grant Parser Module**:

1. Load XML file and break it down into separated XML instances
2. Parse each XML instance into patent handler. For each tag with nested tags, enable stack to store adjacent data
3. Tags that are explicitly defines in ignore list will be left out during parsing. Upon completion, reset the patent handler for new instance
4. Finally, an option to export the entire file to JSON format is available if chosen

DB Client Module

We chose to deploy REST services in communication with our backends instead of OM to provide more scalable solution as we just use HTTP response and request.

DB Client Class Diagram:



DB Client Module (Cont.)

Pseudocode for **POST in MongoDBConnector**:

1. Load all data from JSON file
2. Construct list of citations from group of patent
3. Validate uniqueness within the list
4. Appending statements into query. Then, perform 1st POST these citations
5. If the uniqueness is violated, the response code from EVE server will determine the next step. If error response with duplicates found, remove it from the list and resend it again
6. Finally, perform an overwrite PUT for any individual patent. This step must be linearly execute to guarantee that all the dummy patents, which are created using citations, will be updated with information

DB Client Module (Cont.)

POST in Neo4j is more simpler because it has an overwrite operation called **"MERGE"**, which **automatically detect duplication and merge together or create new one**.

Pseudocode for **POST in Neo4jDBConnector**:

1. Load all data from JSON file
2. Construct list of citations from group of patent
3. Appending statements into query. Then, perform POST these citations
4. Finally, we perform another POST for list of patents

Data Visualization

Data Visualization Process:

1. Submit a query to database server
2. Receive a JSON Response
3. Generate CSV File from JSON Response
4. Use GEPHI to load CSV and visualize data

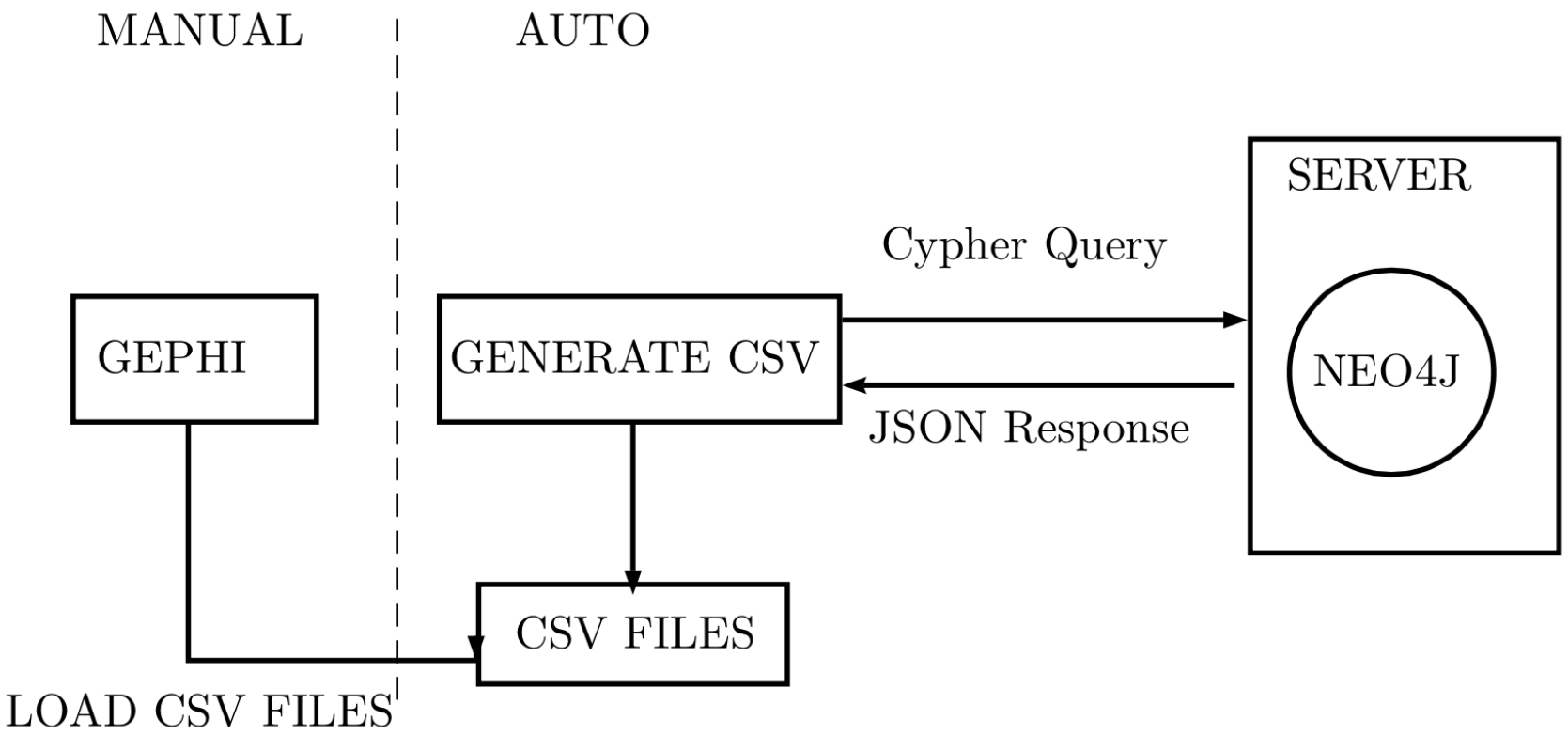


Table of Contents

Results

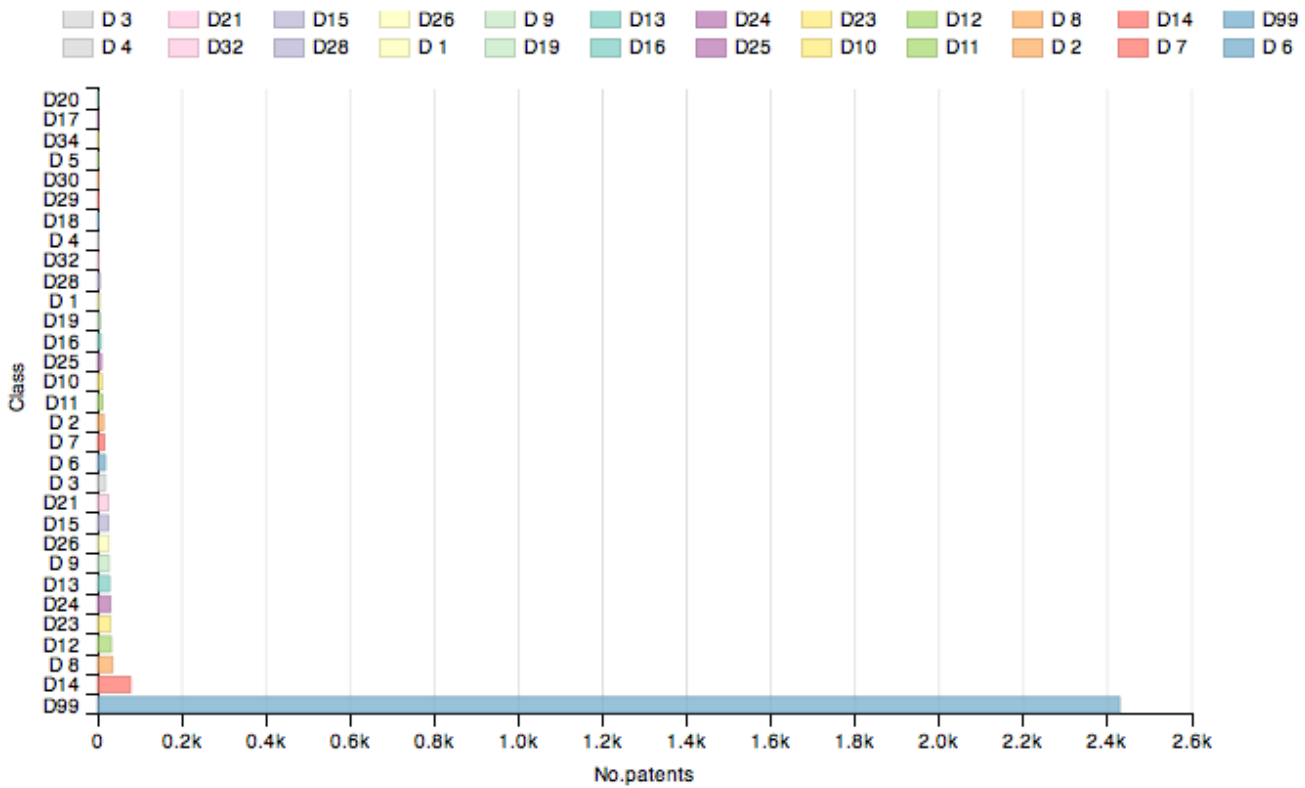
Processed patents:

	Number of Documents
Duplicated Citations	347
Patents	3000
Citations	106996
Total	110343

Results (Cont.)

The figure shows Patent Classification in October 2014.

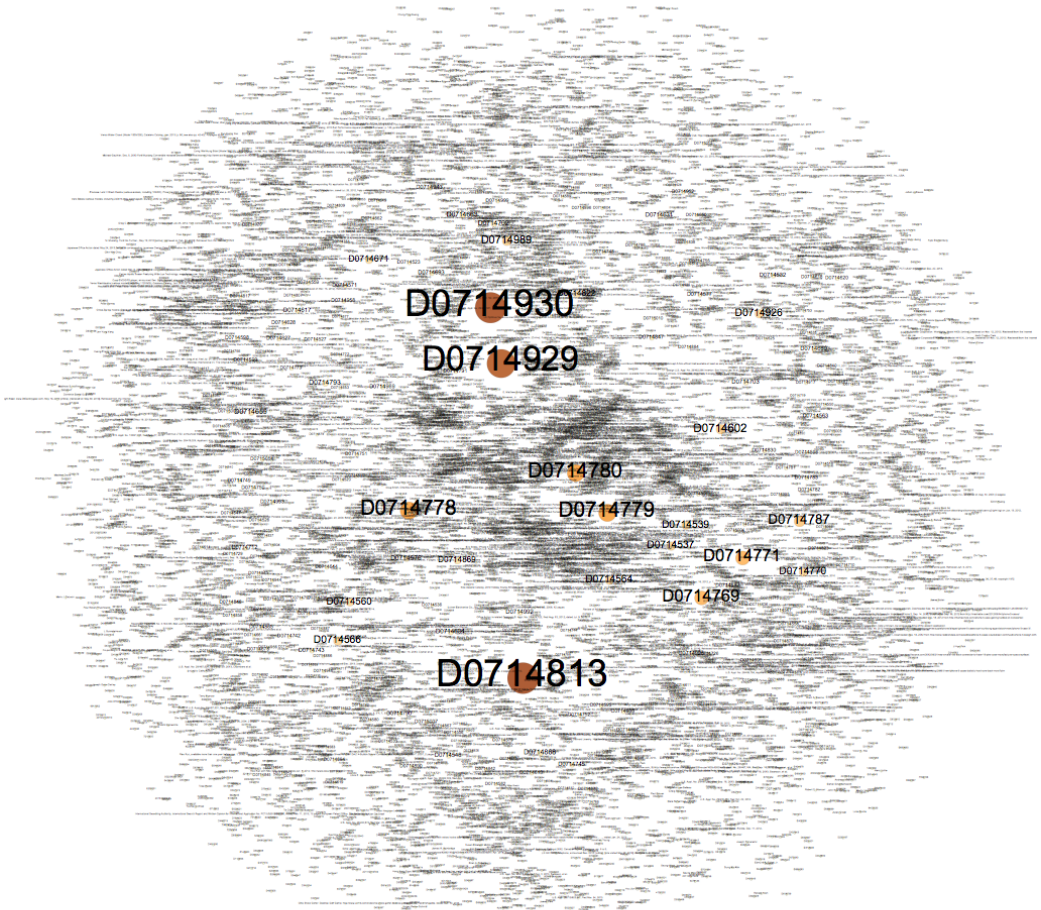
The MISCELLANEOUS (**D99**) industry were blossomed in October 2014



Citation Network

The figure shows a citation network with 500 patents in the database.

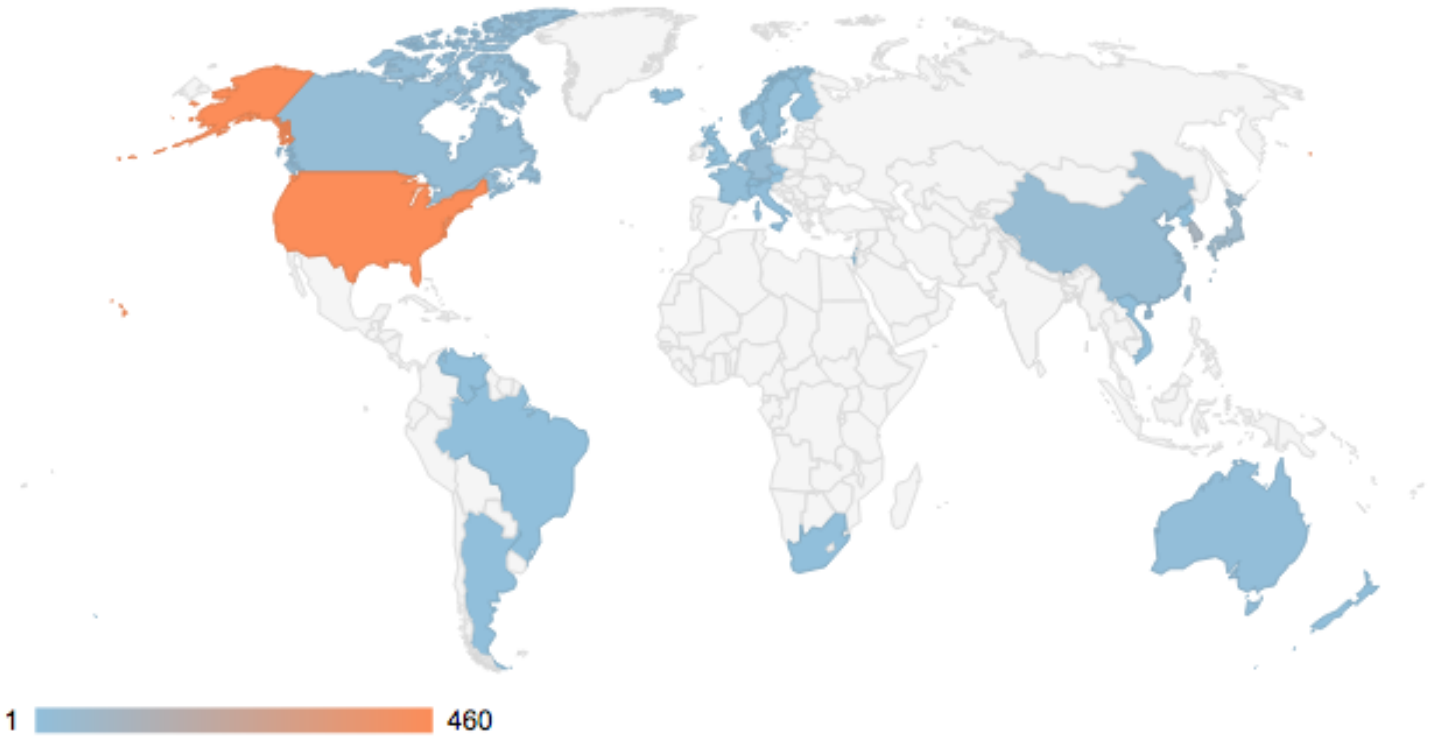
D0714930, D0714929 and D0714813 were the three most cite other patents corresponding to their node's radius.



Inventors Maps

To measure the activity of inventors community, we classify them based on their regions.

Most of inventors come from North America. Furthermore, US patents industry also attracts the numerous inventors from different parts of the world, including Asian and Australian.



Benchmark Neo4j and MongoDB

We perform the time measurements on 500 patents with total of 12016 citations.

Results:

- **DOM** parser is **faster** than **SAX**
- **OGM** is **faster** than **REST**, but **REST** is **simpler** than OGM
- Populate data in **MongoDB** is **faster** than **Neo4j**, but the drawback is MongoDB client has to implement references between documents

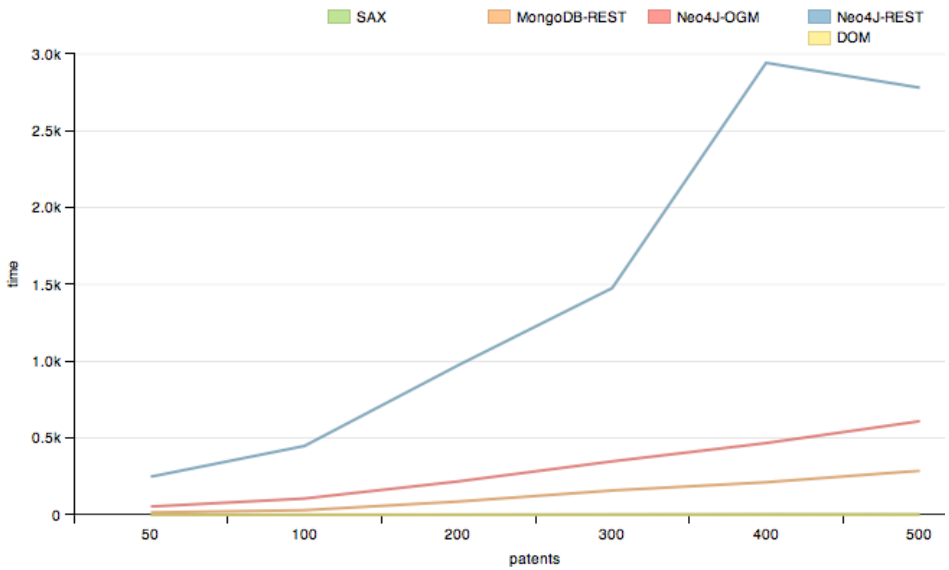


Table of Contents

Future Works

- Divide patent data into different databases instead of storing the whole file into one particular back-end
- Propose a refined structure for storing and querying patent data across multiple database systems
- Provide an end-user services that would transform natural language into our CINAS's language, process the queries, determine the keywords and then pass it to the suitable database system
- Final goals:
  1. Take advantages of multiple database systems
  2. Reduce the query costs
  3. Enhance end-users experience with patent data

References

[1] “Transitioning from Relational Databases to MongoDB - Data Models” B. Reinero, http://blog.mongodb.org/post/72874267152/transitioning-from-relational-databases-to-mongodb, Jan 2014

[2] “USPTO Patents Database Construction” I. Torvik, D. Dubin and Q.Liu, http://abel.lis.illinois.edu/UPDC/USPTOPatentsDatabaseConstruction.pdf, August 2012

[3] “Construction of Japanese Patent Database for Research on Japanese patenting activities” A. Goto, K. Motohashi, http://www.iip.or.jp/e/images/paper.pdf

Thank You

Thank you for your listening!