Fall | 2015

Technical Report

Data Intensive Workflow Development for Software Engineers (18-656)

Analyze DBML Using Graph Database

Team 6

Cici Huang, Kang Fang, Qihao Zhu, Xiatao Jin

Table of Contents

1. Introduction

In computing, a graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data.[1] In term of data size and data complexity, traditional relational database would have too many rows of records or too many content tables, which leads to low efficiency and high expense. However, graph database provides a valuable solution by providing particular data models[2]. In graph database, entities are represented by nodes and relationships are represented by connected lines between nodes. Both nodes and lines can hold properties which are key-value pairs.[3]

We use DBLP as our backend data. DBLP provides open bibliographic information on major computer science journals and proceedings. The original data sources are in XML format.

2. Motivation

For graph database, there are some reasonable and important pros over relational database:

- They allow deep traversals faster than Relational.
- They are fast when searching for relationships of the type friends of a friend
- They allow for very fast execution of complex pattern matching queries.
- They can represent multiple dimensions.
- They can easily handle sparse data properties

In this project, our goal is to build a web application for users to query DBLP database. It should provide users with multiple functions like query paper information, possible coauthor candidates and so on. The size of DBLP database is very huge and data properties in database is very sparse. We utilize these features and choose graph database as our backend database. In our design, authors and papers are displayed in nodes and publishment relationship is shown in lines. Due to the properties of our data, graph database is really efficient and reasonable in our application.

3. Related work

A. Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-Aware Service

---

[1] "Graph database - Wikipedia, the free encyclopedia." 2011. 18 Dec. 2015 <https://en.wikipedia.org/wiki/Graph_database>

[2] "Why Choose a Graph Database - O'Reilly Radar." 2014. 18 Dec. 2015 <http://radar.oreilly.com/2013/07/why-choose-a-graph-database.html>

[3] Dix, Alan. *Human-computer interaction*. Springer US, 2009.

Recommendation for Mashup Creation in an Evolving Service Ecosystem", in Proceedings of The 21th IEEE International Conference on Web Services (ICWS), Jun. 27-Jul. 2, 2014, Anchorage, AK, USA, pp. 25-32.

In this paper, they present a method to extract service evolution patterns by exploiting Latent Dirichlet Allocation (LDA) and time series prediction. A time-aware service recommendation framework for mashup creation is presented combining service evolution, collaborative filtering and content matching.

B. *J. Zhang, W. Tan, J. Alexander, I. Foster and R. Madduri, "Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse", Proceedings of IEEE International Conference on Services Computing (SCC), July 4-9, 2011, Washington DC, USA, pp. 48-55.*

This paper proposes a novel approach of proactively recommending services in a workflow composition process, based on service usage history. In contrast to existing interface-based services discovery approaches, this paper proposes a novel approach of proactively recommending services in a workflow composition process, based on service usage history.

C. *B. Xia, Y. Fan, C. Wu, K. Huang, W. Tan, J. Zhang, and B. Bai, “Domain-Aware Service Recommendation for Service Composition”, in Proceedings of The 21th IEEE International Conference on Web Services (ICWS), Jun. 27-Jul. 2, 2014, Anchorage, AK, USA, pp. 439-446.*

In this paper, a novel approach is proposed to offer domain-aware service recommendation. First, a K Nearest Neighbor variant (vKNN) based on topic model Latent Dirichlet Allocation (LDA) is introduced to cluster services into semantically coherent domains. On top of service domain clustering results by vKNN, a probabilistic matching model Domain Router (DR) based on Extreme Learning Machine (ELM) is developed for decomposing a requirement to relevant domains. Finally, a comprehensive Domain Topic Matching (DTM) model is built to mine relevant domain-specific matching patterns to facilitate service recommendation.

D. *J. Zhang, C. Lee, S. Xiao, P. Votava, T.J. Lee, R. Nemani and I. Foster, "A Community-Driven Workflow Recommendations and Reuse Infrastructure", in Proceedings of The 8th IEEE International Symposium on Service-Oriented System Engineering (SOSE), Apr. 7-11, 2014, Oxford, UK, pp. 162-172.*

This project aims to develop a proactive recommendation technology based on collective NEX user behaviors. In this way, we aim to promote and encourage process and workflow reuse within NEX. Particularly, we focus on leveraging peer scientists' best practices to support the recommendation of artifacts developed by others.

4. System design

Here we attach the architecture of the system below.

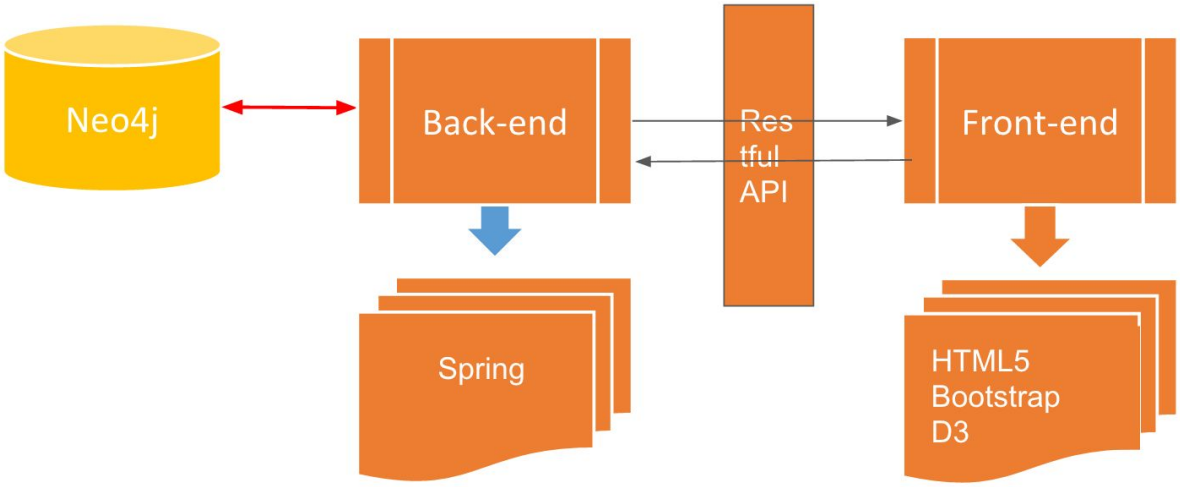


Figure 4.1 Architecture of the project

In the front end, we used HTML5 Bootstrap and D3 for the virtualization. In the back end, we used Spring structure and provided Restful API for communication. Besides that, we used Neo4j as our graph database to store all the data.

Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user shown in figure 4.2

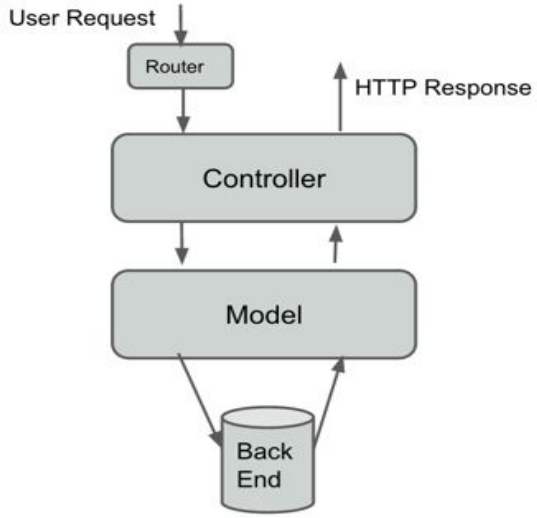


Figure 4.2 Workflow of the front end

In this project, we define:

- model: Data Model and Structure for climate service and climate service log
- view: all web page use scala template html to render.
- controller: handle and process the data from backend to frontend and from front end to back end.

5. System implementation

5.1 Development Tools

Spring Framework:

Spring is an open source web application framework, written in Java, which follows the model–view–controller (MVC) architectural pattern. It aims to optimize developer productivity by using convention over configuration, hot code reloading and display of errors in the browser.[4]

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBeans (EJB) model. The Spring Framework is open source[5].

Eclipse:

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in other programming languages: Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Natural, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

---

[4] "Play framework - Wikipedia, the free encyclopedia." 2014. 18 Dec. 2015 <https://en.wikipedia.org/wiki/Play_framework>

[5] "Spring Framework - Wikipedia, the free encyclopedia." 2011. 18 Dec. 2015 <https://en.wikipedia.org/wiki/Spring_Framework>

5.2 Front-end Implementation

Techniques used in front-end implementation include HTML5, Javascript, Bootstrap, JQuery and D3.js.

HTML is the main content shown in the webpage. To make the webpage looks fancier, we use Bootstrap framework to decorate the webpage. We also implemented flatern style with Bootstrap, thus it is neat, comfortable and fancy.

Javascript is used to handle the events triggered by user. When some text is typed into the text box and a button is clicked, it triggers an event in which, Javascript finishes the business logic behind the action. JQuery is a cross-platform Javascript library designed to simplify the client-side scripting of HTML. Ajax is used to dynamically refresh parts of the page. It calls restful api to communicate with back-end retrieve the data from server asynchronously without interfering with the display and behavior of existing page and refresh the specific part of the page.

D3.js is a Javascript library for manipulating documents based on data. D3.js has extraordinary abilities to visualize data. In this project, D3.js is mainly used to visualize the graph of publication network.

5.3 Back-end Implementation

5.3.1 What the back end does

The main responsibility of the back end is to provide the RESTful API for the front end so that the two layer of the system could entirely separated.

Here is a list of APIs provided by the back end:

	Description	HTTP Metho	Partial URL	Request Data	Response Data	Success Status Code
2	Given an author name return its co-author	POST	/getCoAuthor/:name	Author Name	List of author names	200
3	Given an author name return its coco-author	POST	/getCoCoAuthor/:name	Author Name	List of author names	200
4	Given some keywords, , generate a graph of top k related papers together with their authors.	POST	/graphTopKByKeyword/:keyword /:limit	Keyword	List of nodes and relations	200
5	Given a timeline of years for a set of authors, generate a graph showing their publications per year	POST	/timelineOfAuthors	{startYear, EndYear, AuthorList}	{[author: [Year: [publication]]]} {[{},{},{}...]}	200
6	Given a journal name, generate a graph showcasing authors contributing to each of its volume, taking volume as the base axis	POST	/journalGraph	journalName	{contribution:[author, contribution]}	200
7	Given some keywords, search for researchers who are experts in the field	POST	/findExpert	{keywords : []}	list of authors	200

8	Given some keywords, search for researchers that may like to be your collaborators	POST	/findCollaborators	{keywords : []}	list of authors	200
9	Categorize the research papers (given time period)	POST	/categorize	{startYear, endYear, channel, keywordList}	{[category : [papers]]}	200
10	Categorize the research papers (given time period, publication channels, keywords).	POST	/categorize	{startYear, endYear, channel, keywordList}	{[category : [papers]]}	200
11	Generate a Paper-Paper network, showing their citation relationships	GET	/graphPaper2Paper	none	List of papers and relations	200
12	Generate a Paper-Person network, showing their authoring relationships	GET	/graphPaper2Person	none	list of authors and paper and relations	200
13	Generate a Person-Person network, showing their collaboration relationships.	GET	/graphPerson2Person	none	list of authors and coauthors and relations	200
14	In a publication network, click on an author, show a knowledge card summarizing her past publication status	POST	/getAuthorStatus	author Name	list of papers	200
15	In a publication network, click on a paper, show a knowledge card summarizing its publication information and citation data.	POST	/getPaperInfo	Paper Name	Details of paper	200
16	Show the evolution of focused topics of a journal, in a given time frame.	POST	/getJournalEvolution	{startYear, EndYear, journalName}	{[year, topic]}	200

Figure 5.1 APIs provided by the back end

There are three parts of the APIs covering three main functionalities of system. The first part concerns about the basic operations to the climate services, which include adding, updating, retrieving and deleting a certain service. The second part is used to record the parameters of the service while a certain climate service is invoked by the user. It allows the client to add a certain service's parameter or get all of the services' parameters in the format of json or csv. The last part is about recording the execution logs of users. It enables functionalities like recording a user's execution logs using POST, getting all the execution logs from all users, and retrieving a user's execution logs within a given period of time using its user id.

5.3.2 How the back end works

Here is a work flow of the back end.

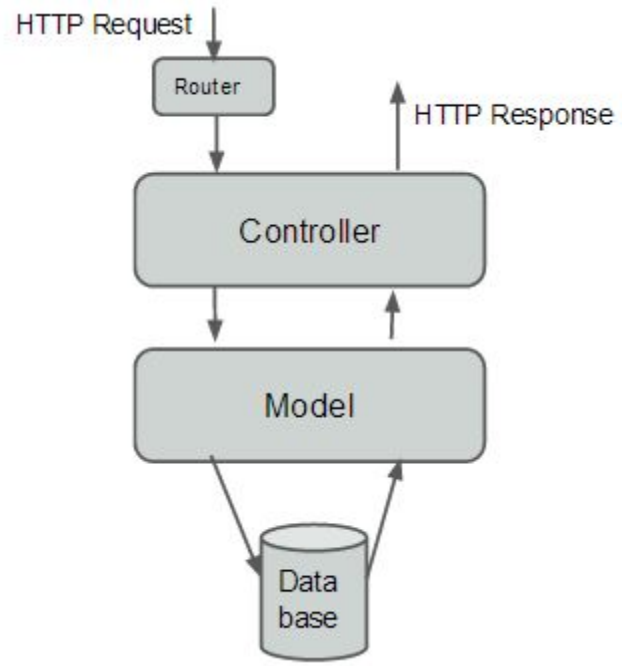


Figure 5.2 Workflow of the back end

The front end and the back end communicate and transmit data using HTTP protocols. Once the back end receives a HTTP request from the front end, the router, a component which is already built in the framework of Play, will direct the request to its corresponding methods defined in the Controller according to a user-defined configuration file, routes. The Controller will first retrieve the data (if any) from the request and parse the data. Then in terms of the request, it either directly update the Java Beans defined in the Model or invoke the methods defined in the Data Access Object (DAO) in the Model to interact with the database.

5.3.3 How the backend is implemented

Most of the workflow of the system are quite similar with what's described above. The following is an sequence diagram for getting an author status, which illustrate how different classes involved interact with each other.

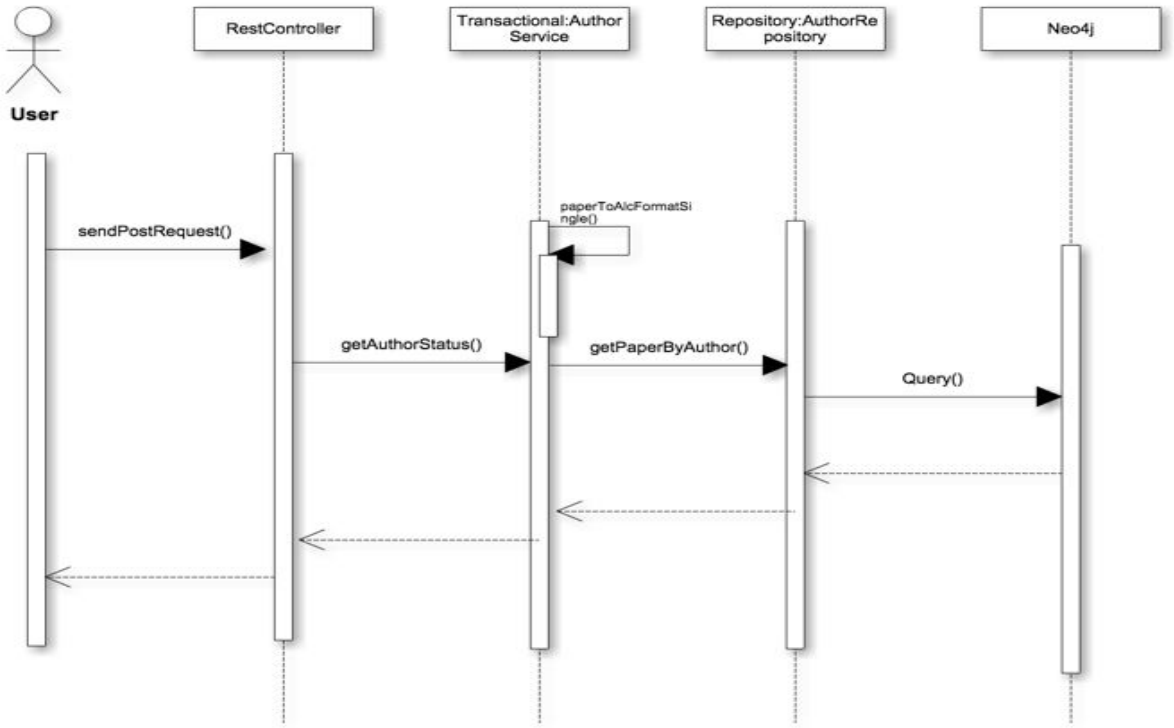


Figure 5.3 Sequence Diagram of getting an author status

First of all, the client, a browser here, sends an HTTP POST request to the back end. The data of the climate service is encapsulated in the format of JSON in the body of the request. Once RestController, which is a class defined by the Spring framework, receives the request, it calls the method of getAuthorStatus() defined in the class of controller. The controller first retrieve the JSON data from the request and parse the data to needed information. Then the controller will invoke the method of getPaperByAuthor() in AuthorService with parameters parsed from the JSON data. The service will directly query the data to the Database by invoking update method defined by the Spring framework. After the Database successfully store the information of the new service, a “true” of boolean type will be returned. Once the Controller receive the returned value, it will send a HTTP response to the browser. Here in this case, it will return a “200, created” response to the front end.

5.4 Database Implementation

5.4.1 Parse data principles

As all data stored in dblp is in XML format, we use SAX Parser to parse the XML file. We utilize JAVA API provided by Neo4j to connect to database. Once detailed data is parsed, we connect to the Neo4j database and then create graph nodes or relationships correspondingly.

5.4.2 Database size

The process of parsing data followed the following steps:

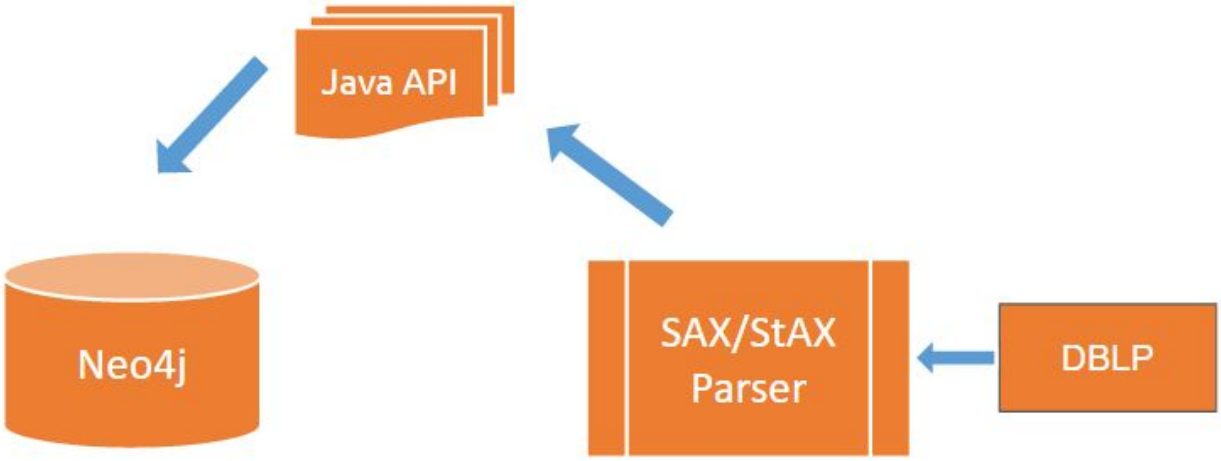


Figure 5.4 Flow graph of parsing data

Right now, we have put over 10,000 authors, 7,000 papers and 17,000 relationships in our Neo4j database.

6. Conclusion and future work

6.1 Conclusion

Based on the DBLP dataset, we parsed the data into Neo4j graph database and built the web service to help users to query. The service oriented computing technology has enabled people to publish and share reusable data analytics algorithms/software as universally accessible web services so that people can choose proper data services as components to build workflows faster.

6.2 Future work

Due to the time limitation, future work is required for refining this project. We will improve this project from those aspects: performance, scalability and synchronization.

We did not load the whole dataset into Neo4j database due to the huge size of the dataset. One of the directions for the future work could be to improve the performance of parsing data and the query time so that the users could get a better experience especially when the data is big.

Secondly, we can keep working on improving scalability of our project. It will be great if we could better leverage the API to provide more complex services.

Besides that, the data we used for this project is static data. We could also focus on how to improve the query when the data is dynamic.

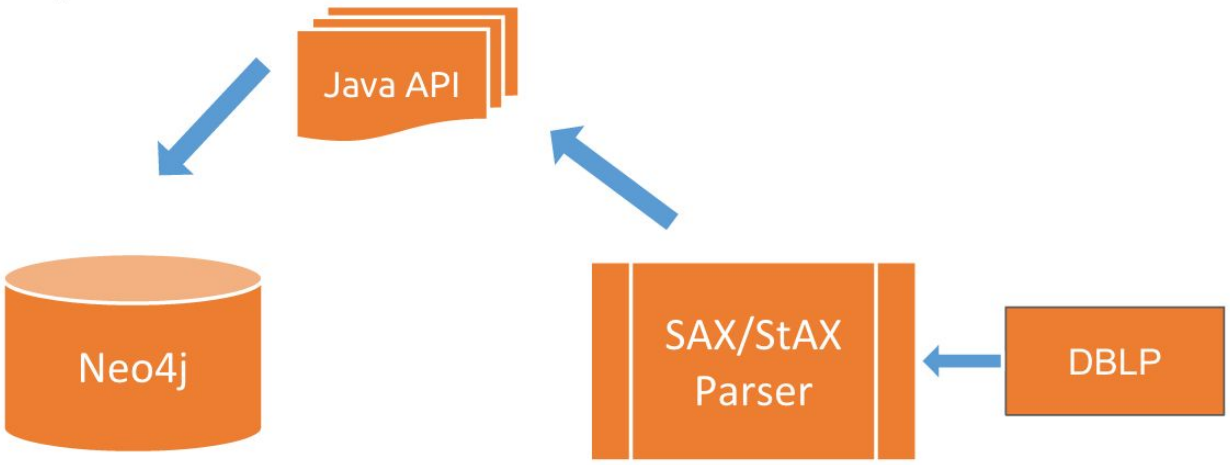
7. Appendix

7.1 Tutorial

1. Install JAVA 1.8 and Maven.
2. Install Neo4j and set up the password.
3. Download the project from github.
4. Store data into your Neo4j database by running the new folder in the github project called “XML”.
5. mvn spring-boot:run -Drun.jvmArguments="-Dusername=neo4j -Dpassword=XXX" (XXX is your neo4j database password)

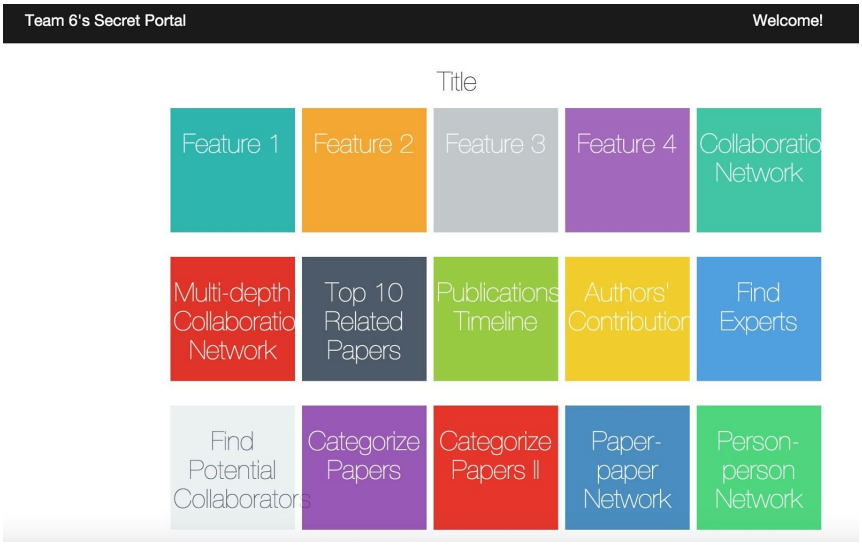
7.2 The features we implemented

1. How to store all publication information in Neo4j, including paper, author, affiliation, topics, etc.



We used SAX to load DBLP to Neo4j with the JAVA API provided.

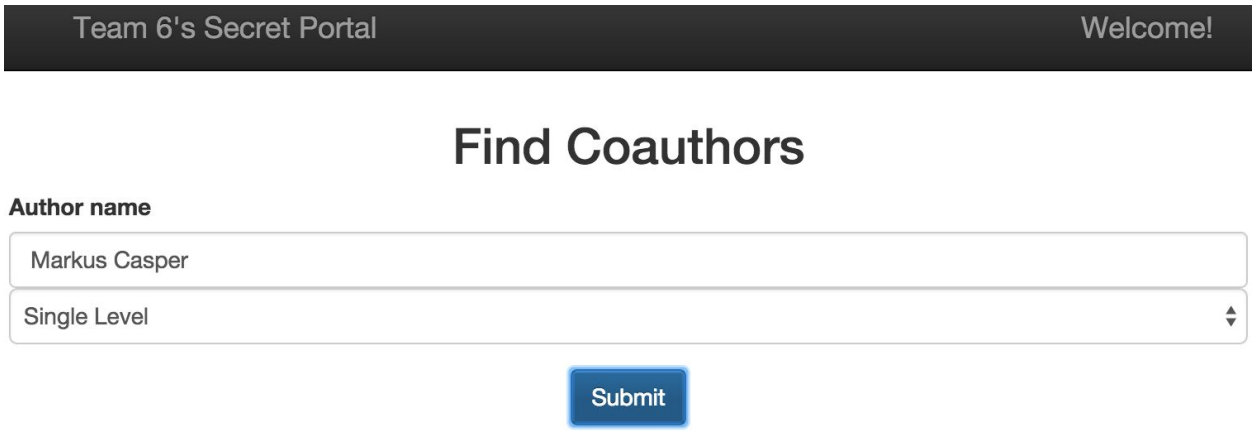
2. Develop a user-friendly web portal to support the data analytics.

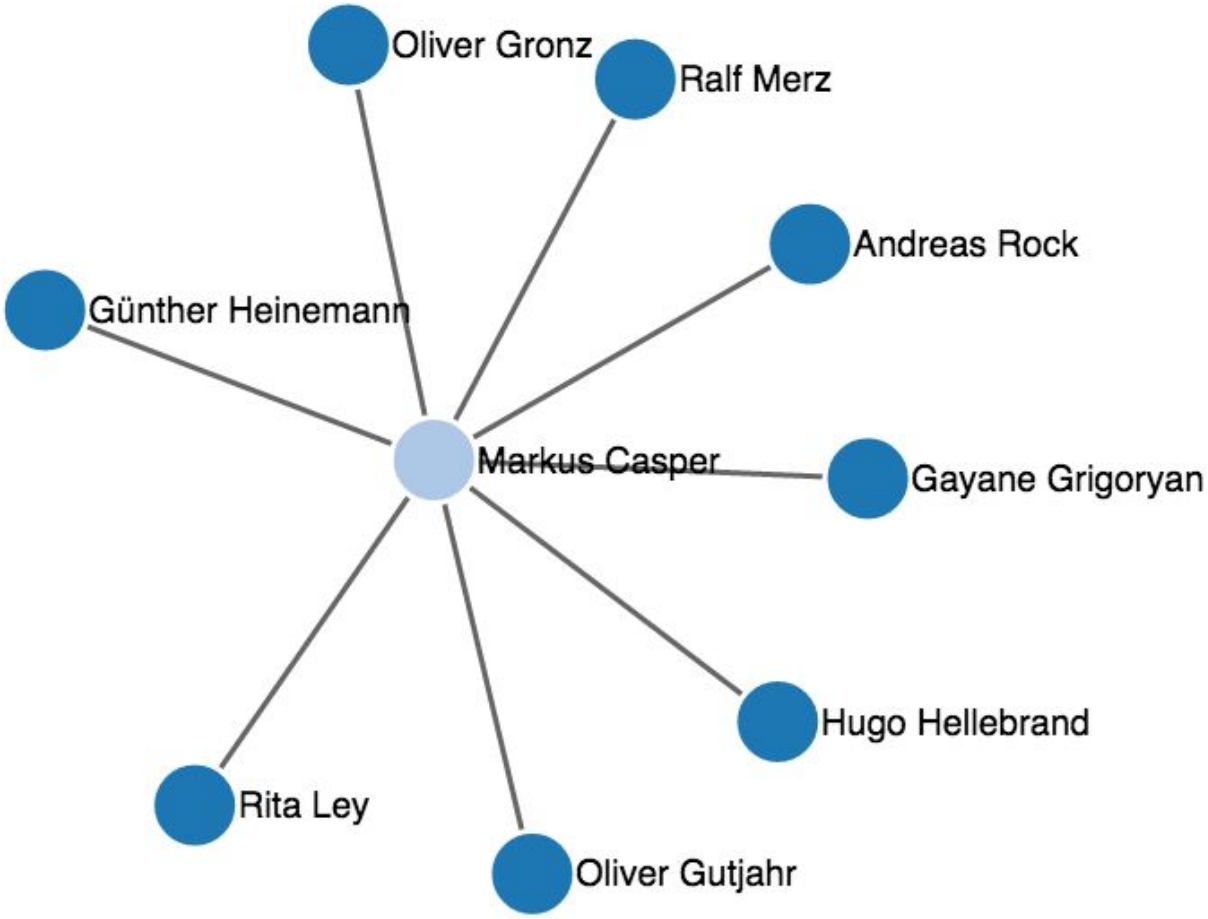


3. How to ensure scalability performance? (Hint: cache user query, create microservices)▫

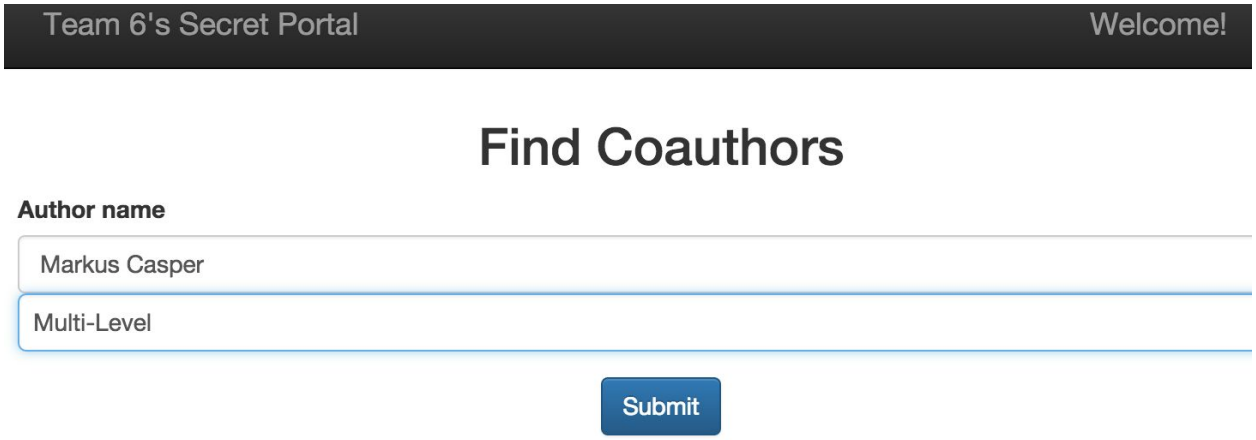
4. How to ensure performance? (Hint: periodic statistics for example)▫

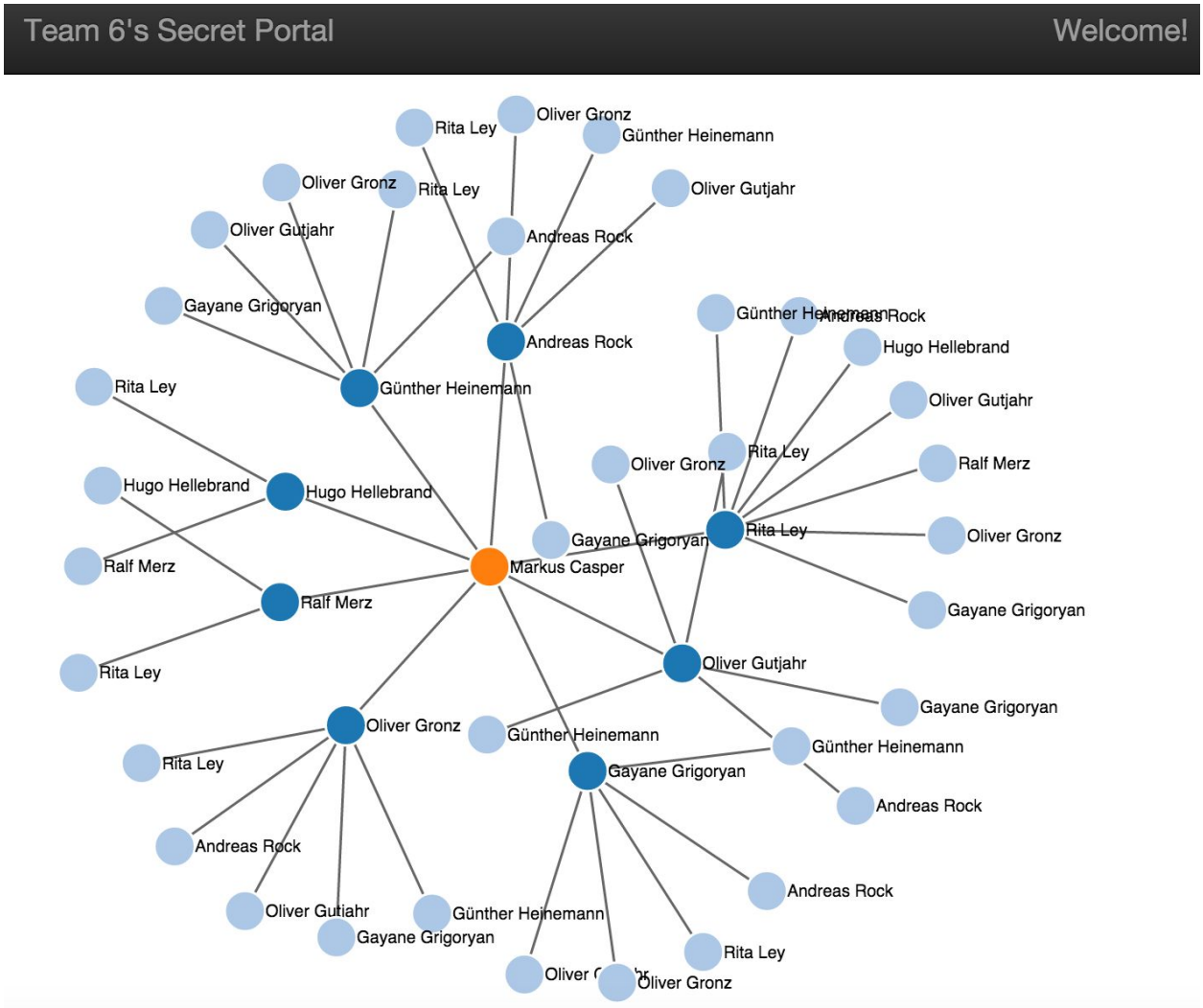
5. Given the name of a researcher, generate a graph* showing the direct collaboration network of the author (her all co-authors).▫



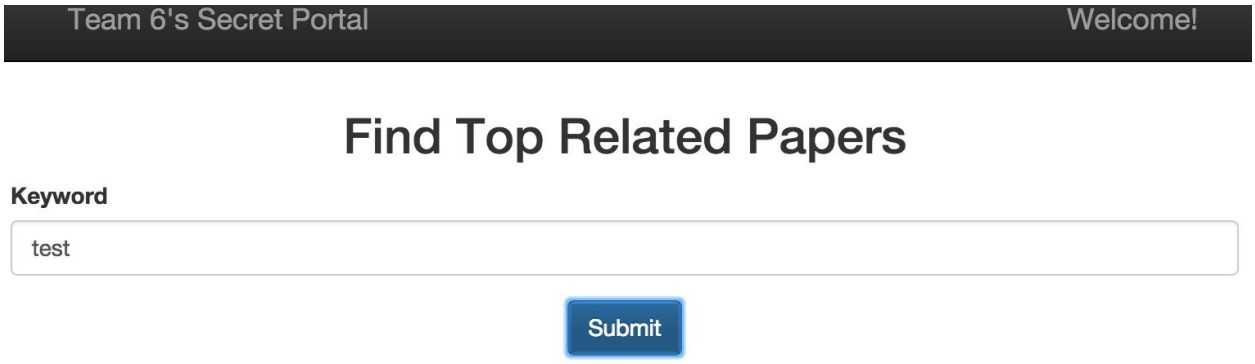


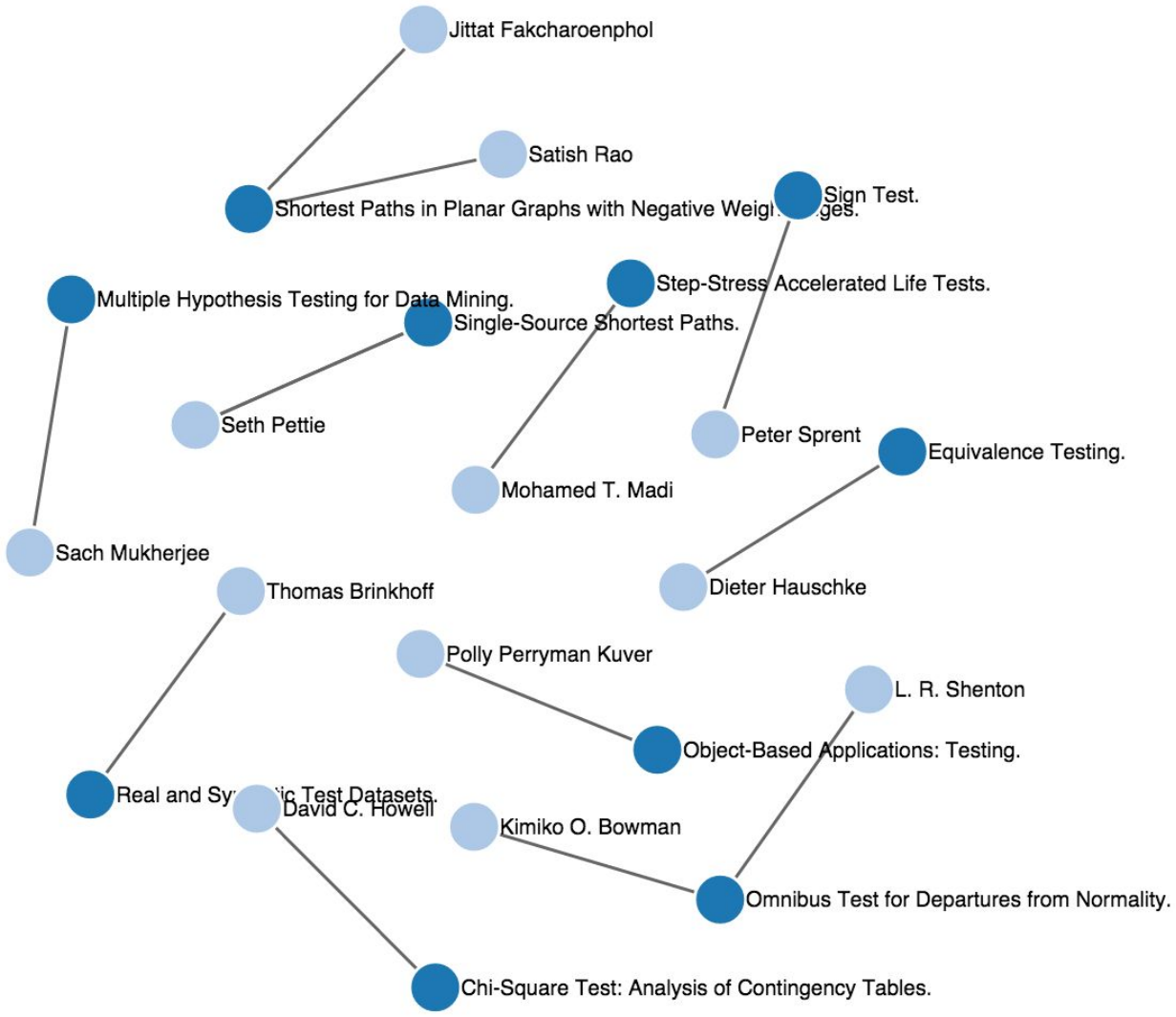
6. Given the name of a researcher, generate a graph* showing a multi-depth collaboration network of the author (her co-authors and their co-authors).





7. Given some keywords, generate a graph of top k related papers together with their authors.





8. Given a timeline of years for a set of authors, generate a graph showing their publications per year.□

Team 6's Secret Portal Welcome!

Publication Timeline

Start year

1990

End Year

2010

Author name1

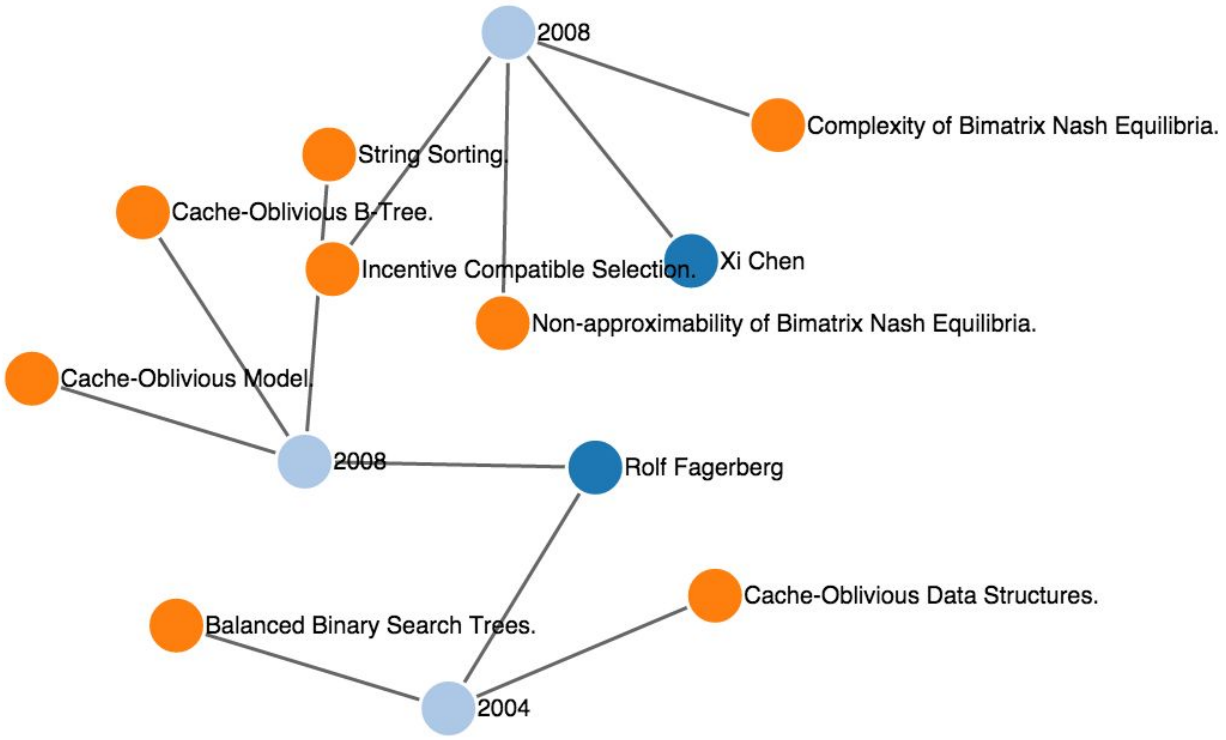
Xi Chen

Author name2

Rolf Fagerberg

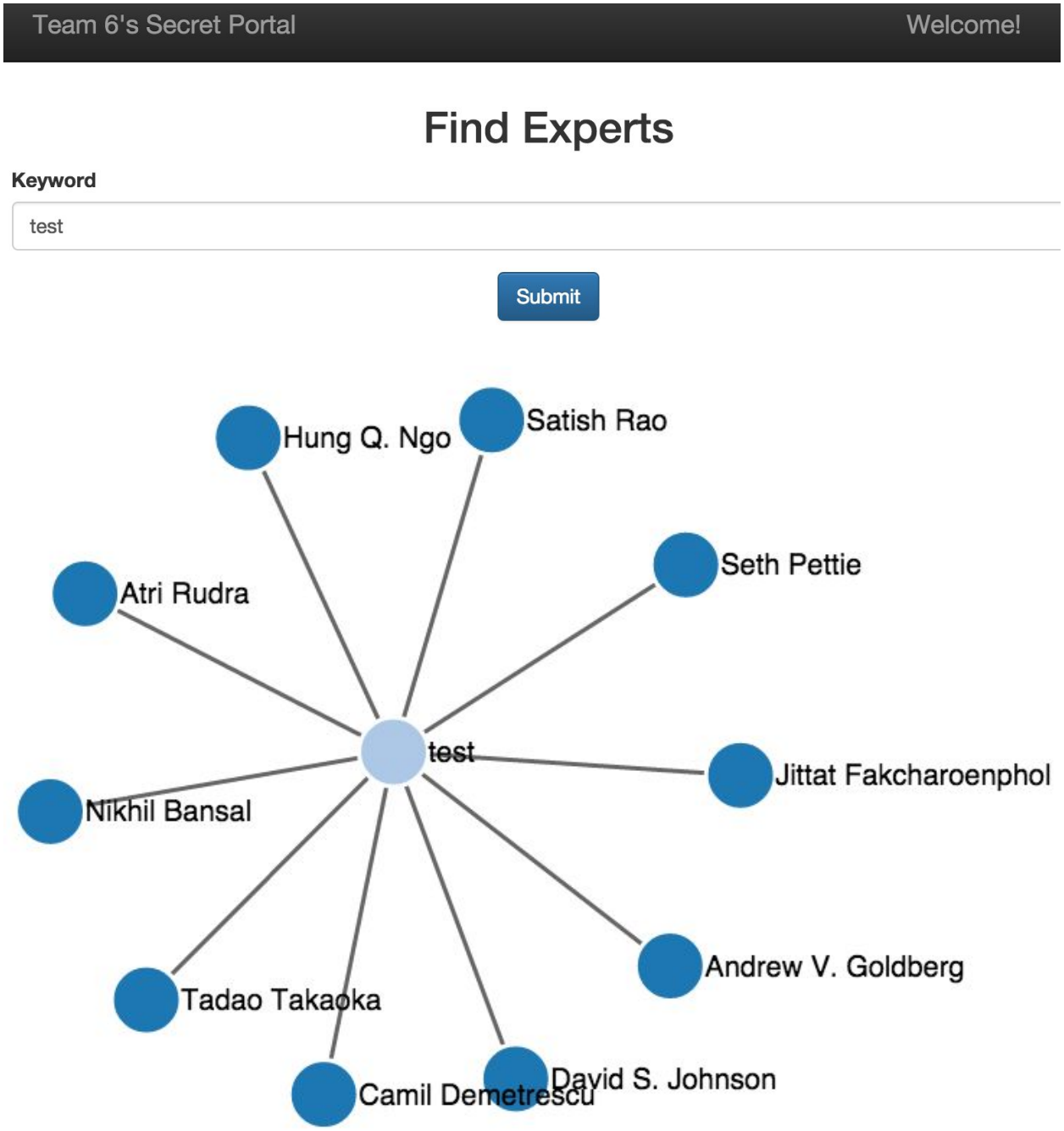
Author name3

Submit

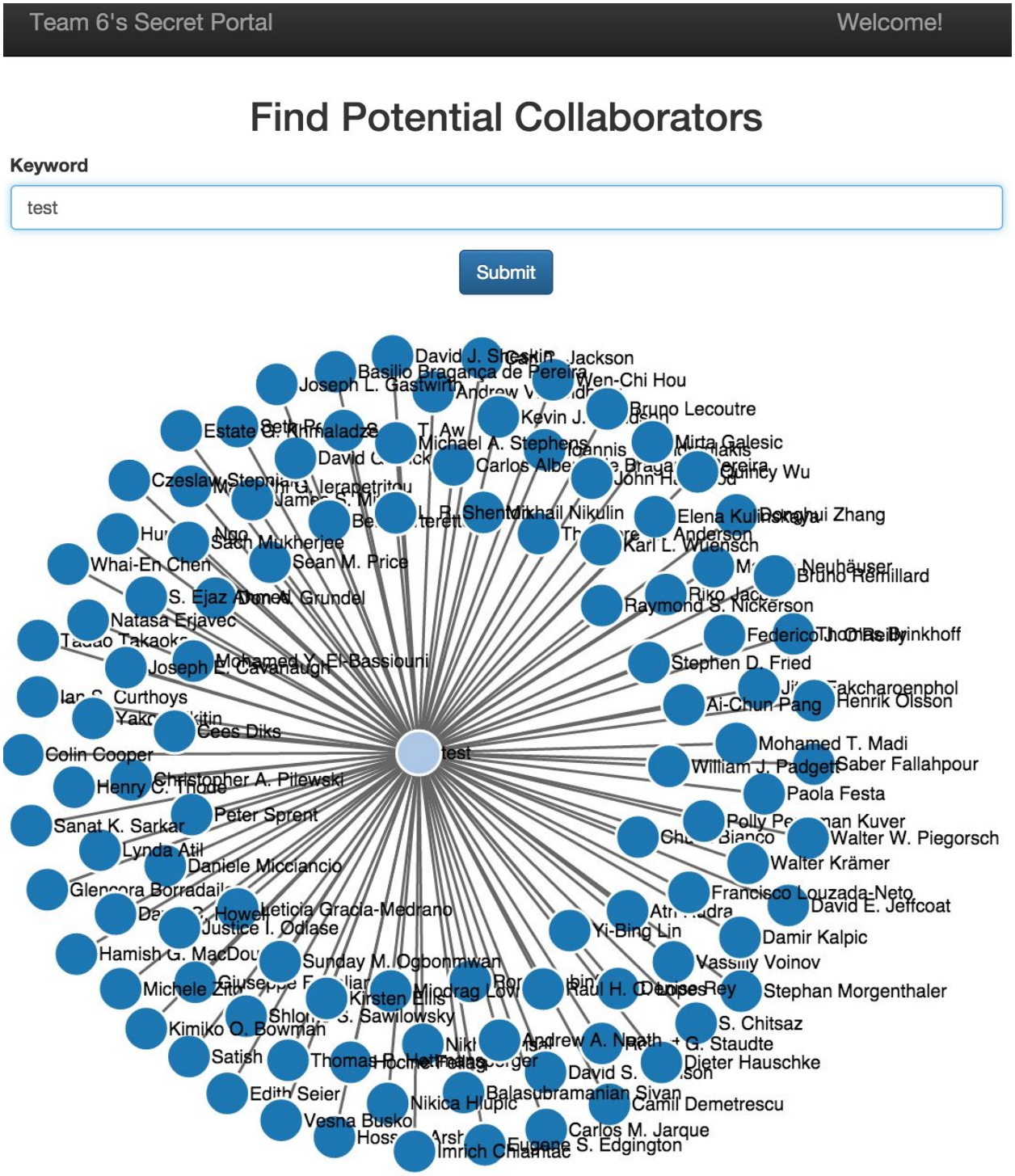


9. Given a journal name, generate a graph showcasing authors contributing to each of its volume, taking volume as the base axis.

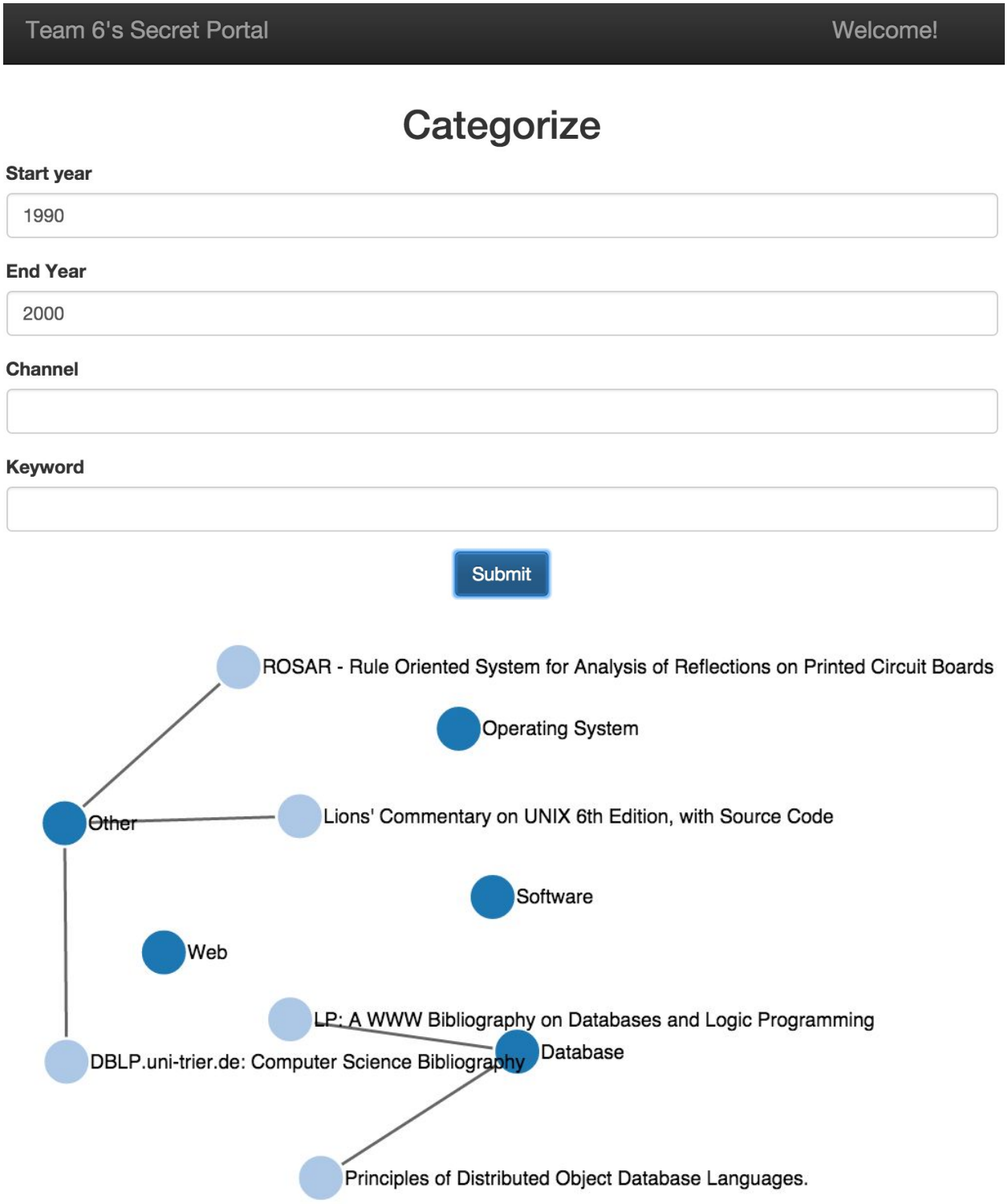
10. Given some keywords, search for researchers who are experts in the field



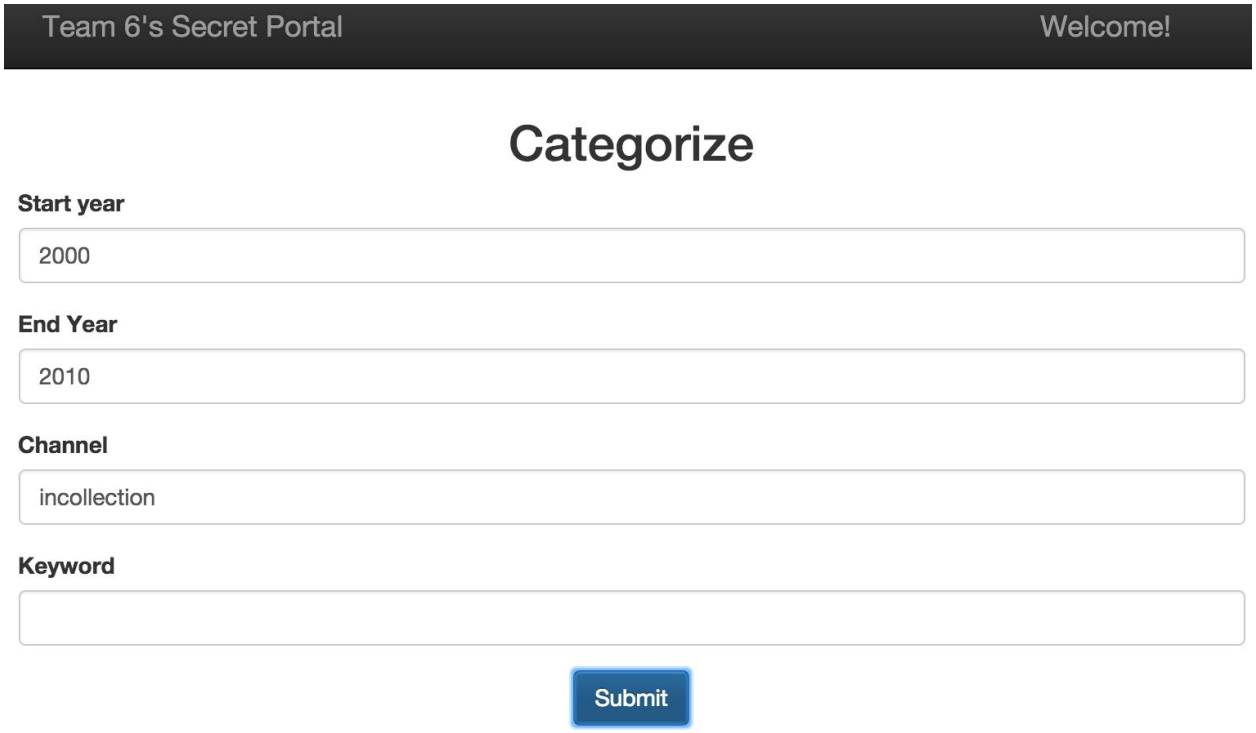
11. Given some keywords, search for researchers that may like to be your collaborators.□

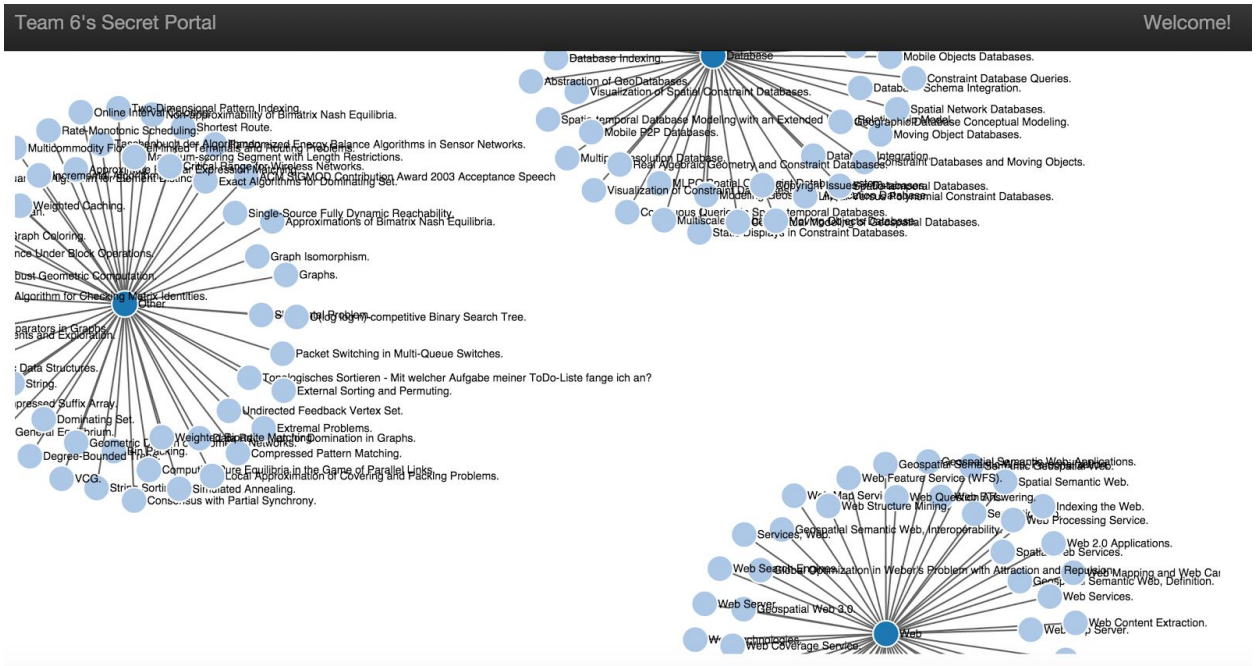


12. Categorize the research papers (given time period).□

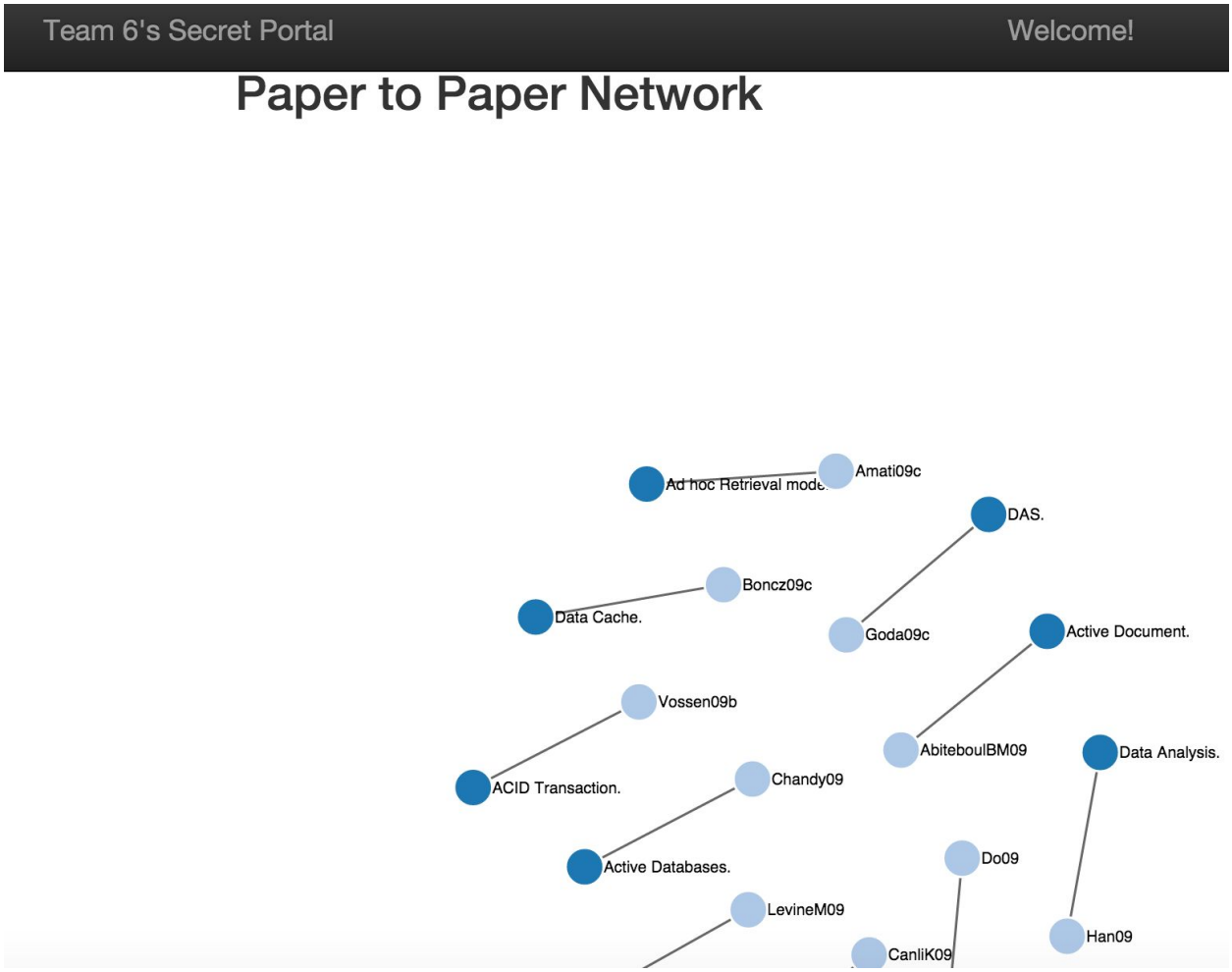


13. Categorize the research papers (given time period, publication channels, keywords).

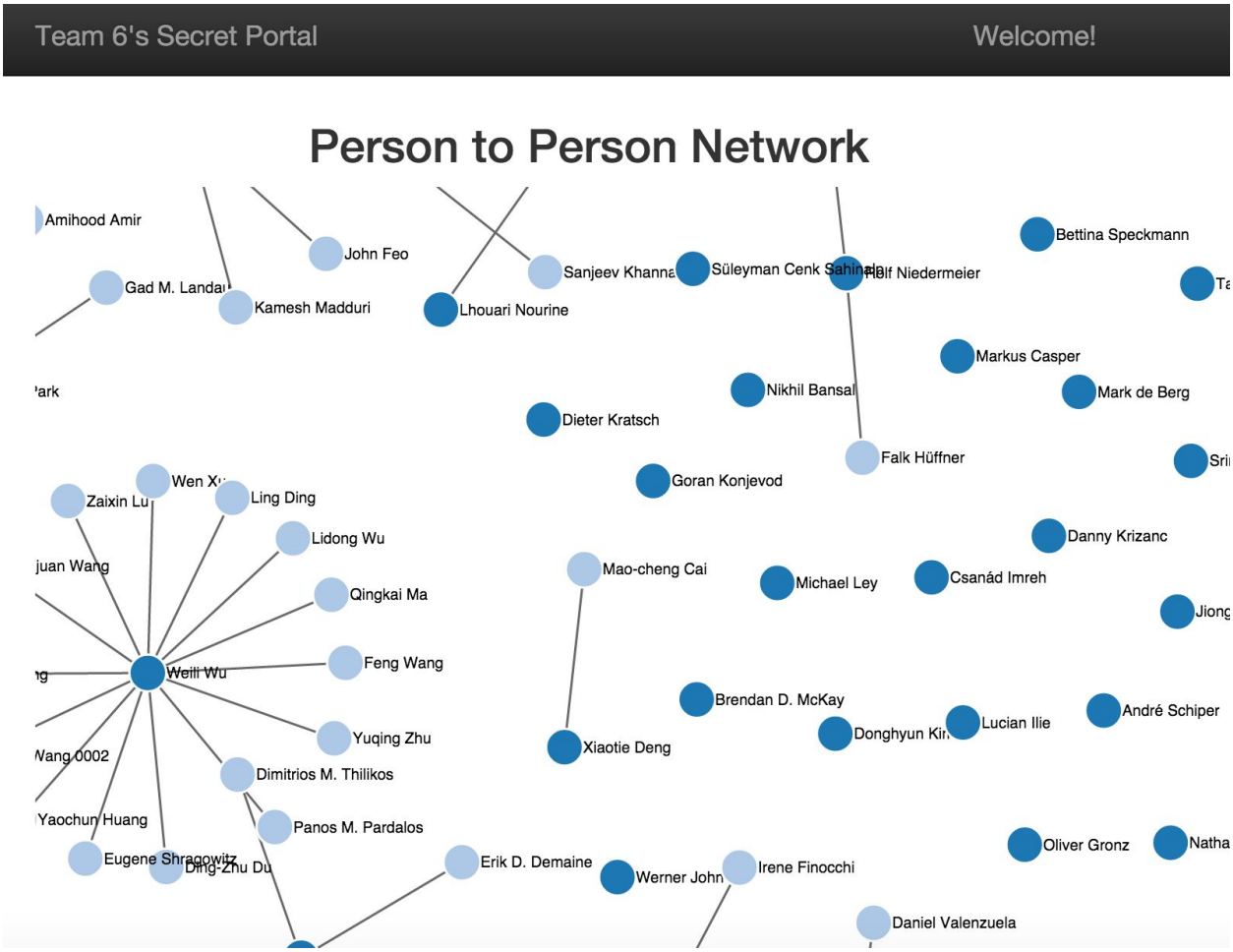




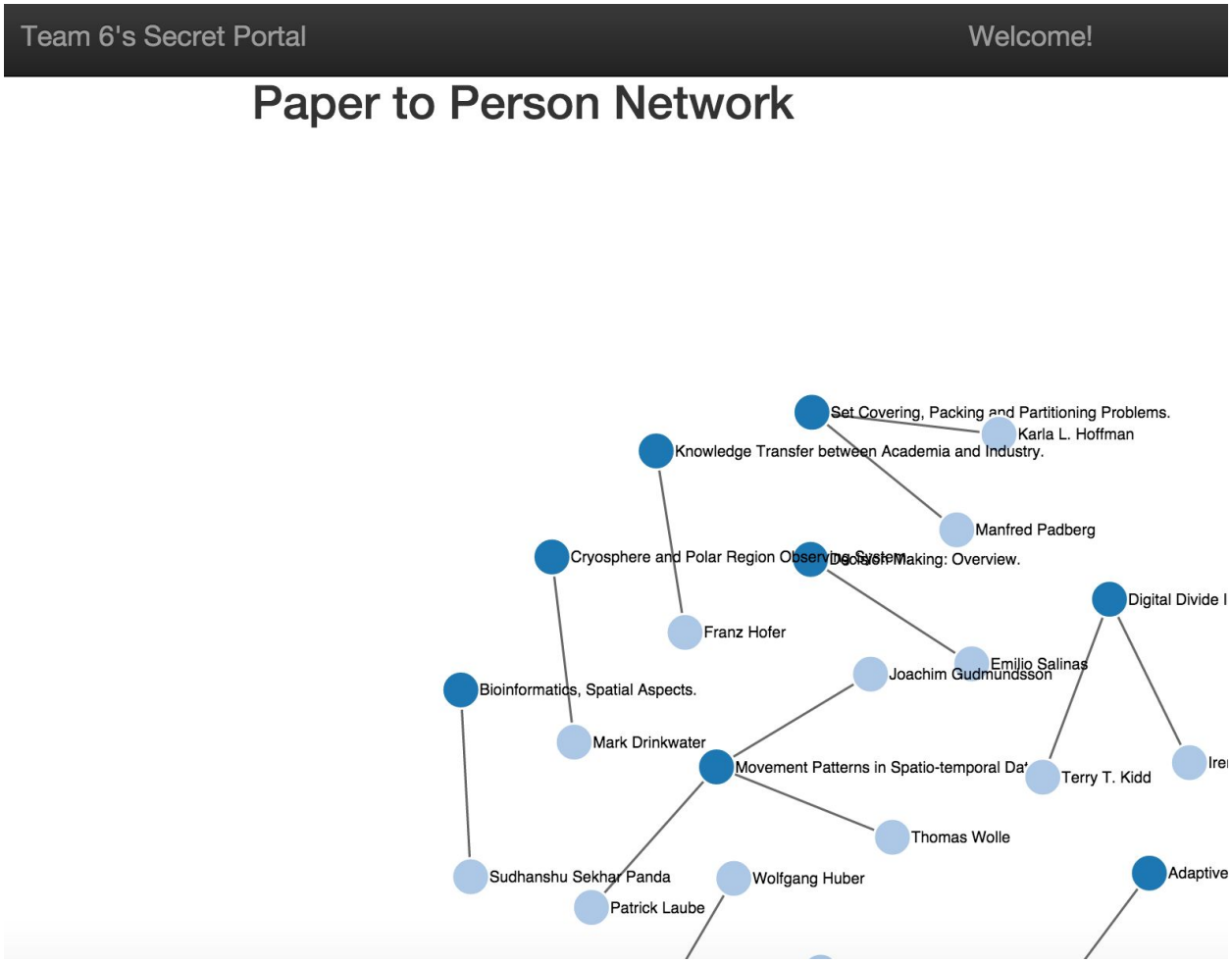
14. Generate a Paper-Paper network, showing their citation relationships.□.



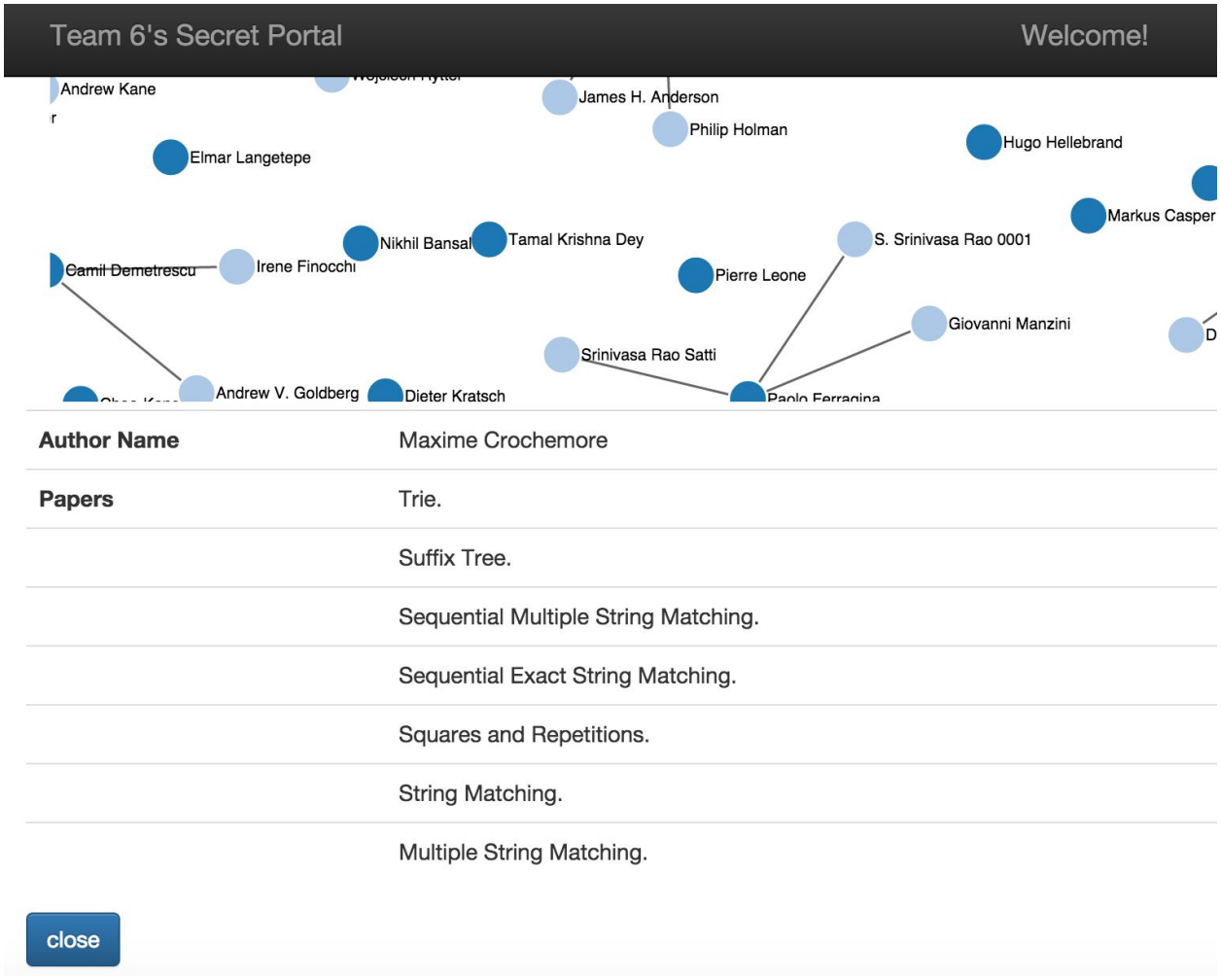
15. Generate a Person-Person network, showing their collaboration relationships.



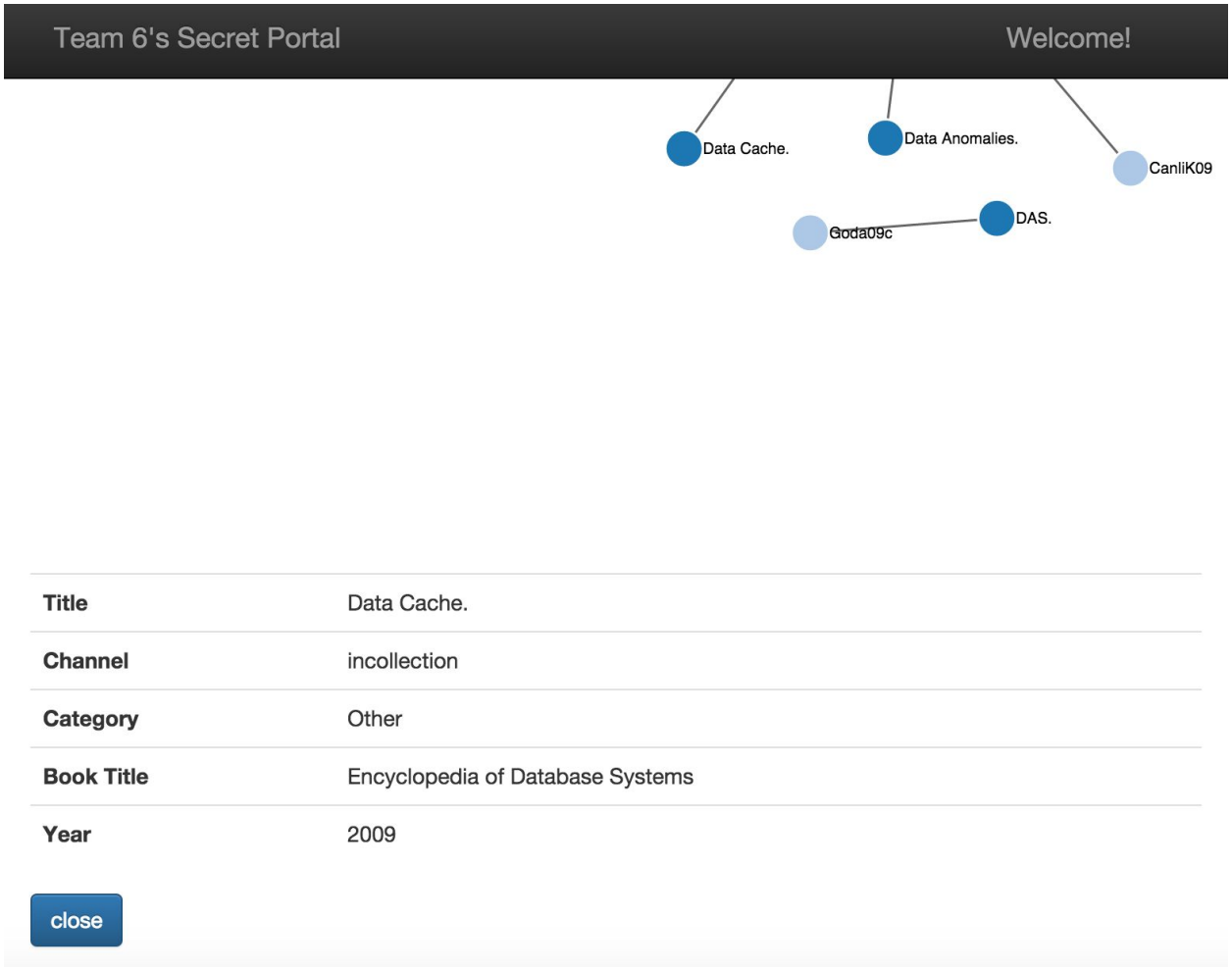
16. Generate a Paper-Person network, showing their authoring relationships.□



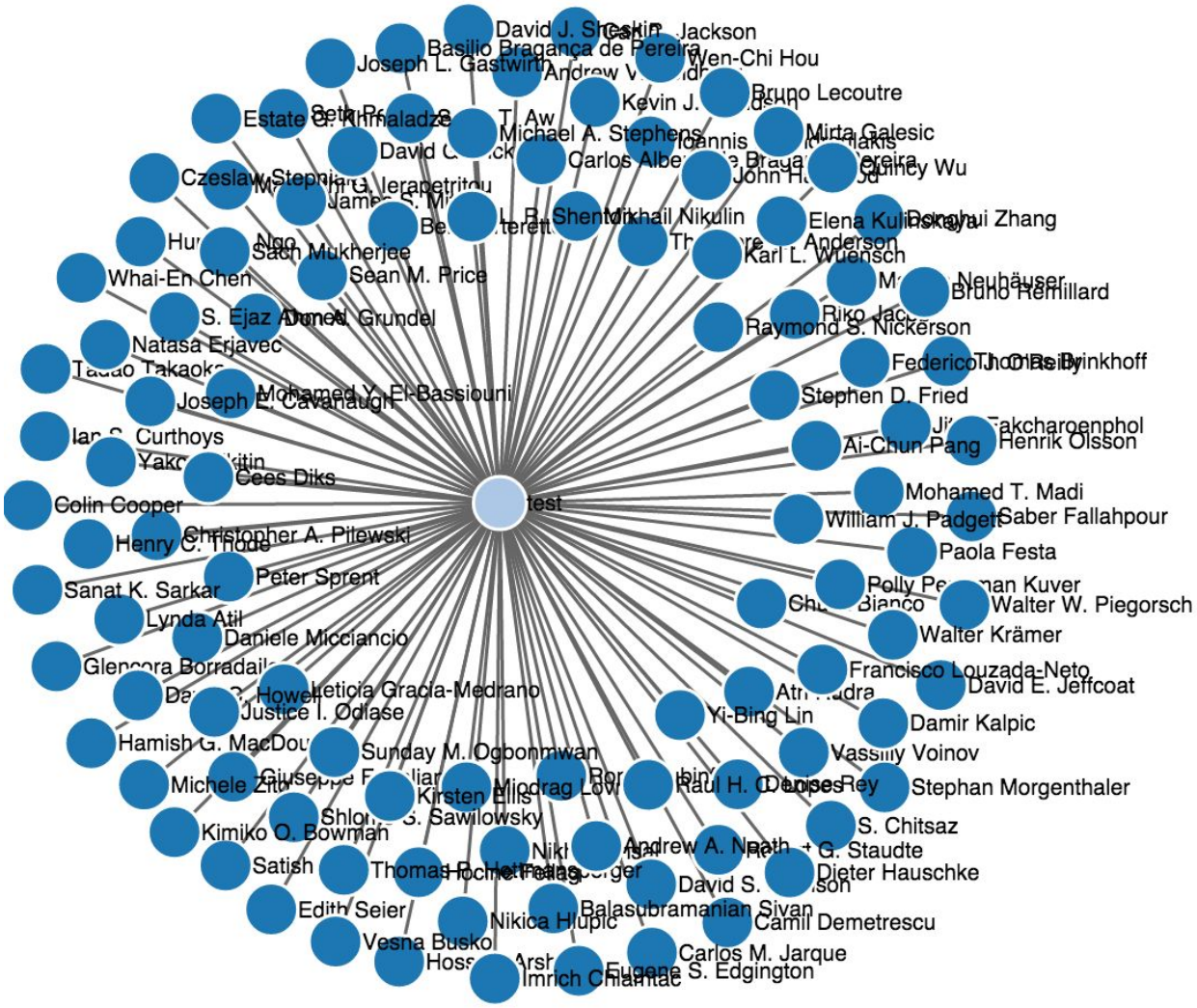
18. In a publication network, click on an author, show a knowledge card summarizing her past publication status.



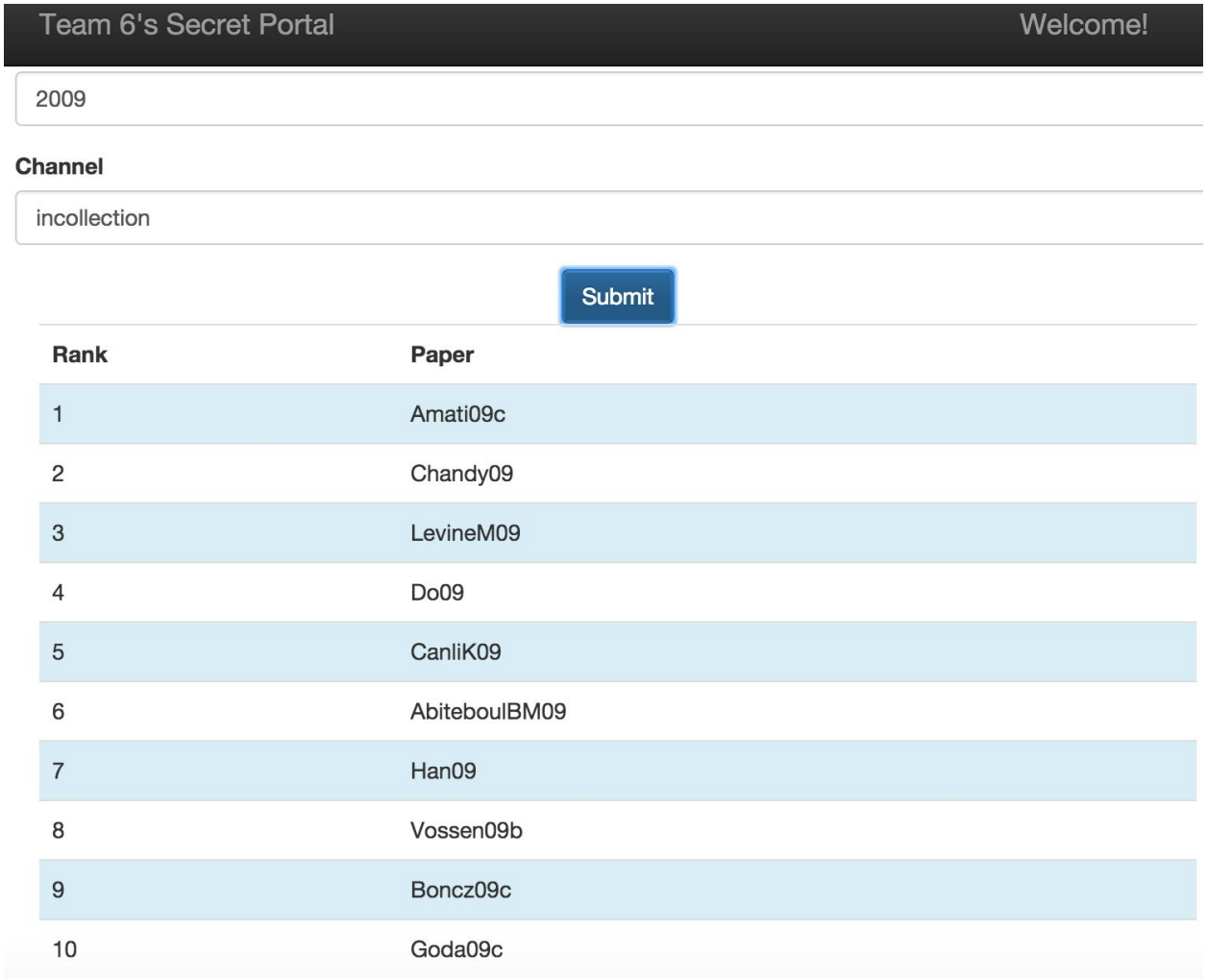
19. In a publication network, click on a paper, show a knowledge card summarizing its publication information and citation data.



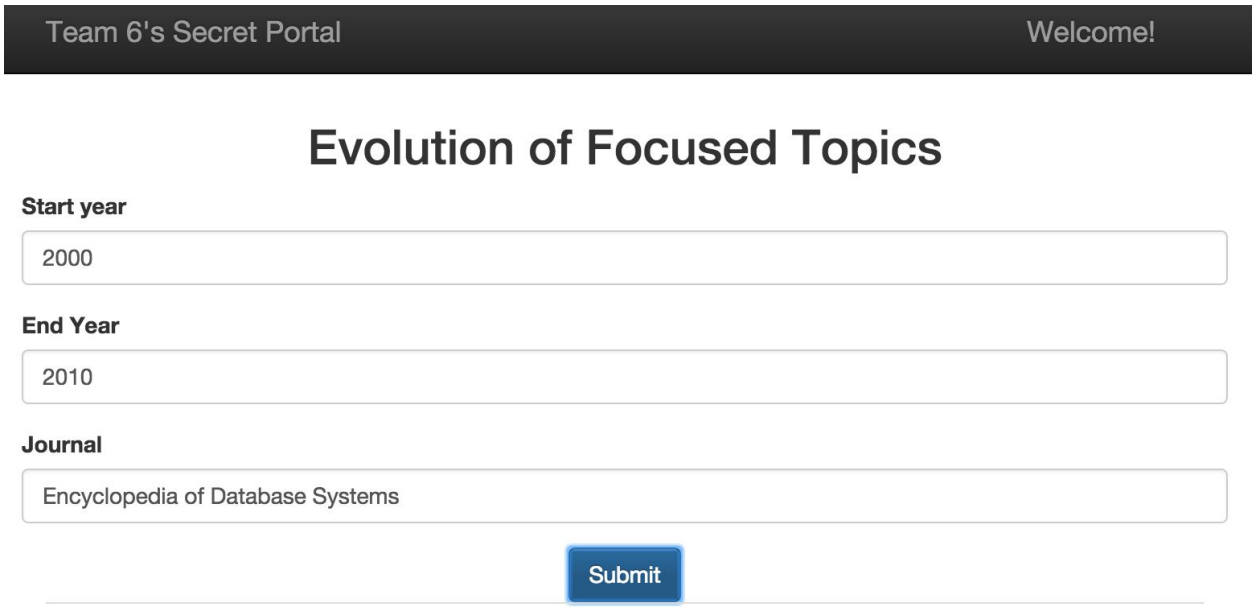
17. Given a research topic, form a possible research team, taking into consideration of their expertise in the field, their past collaboration relationships, and their possibility of willingness to join the team.



20. For each major publication channel (e.g., a journal or a conference at a year), list the top cited papers.



21.Show the evolution of focused topics of a journal, in a given time frame.



Journal

Encyclopedia of Database Systems

Submit

Year	Focused Topic
2000	Database
2001	Database
2002	Database
2003	Database
2004	Database
2005	Database
2006	Database
2007	Database
2008	Database
2009	Other
2010	Database