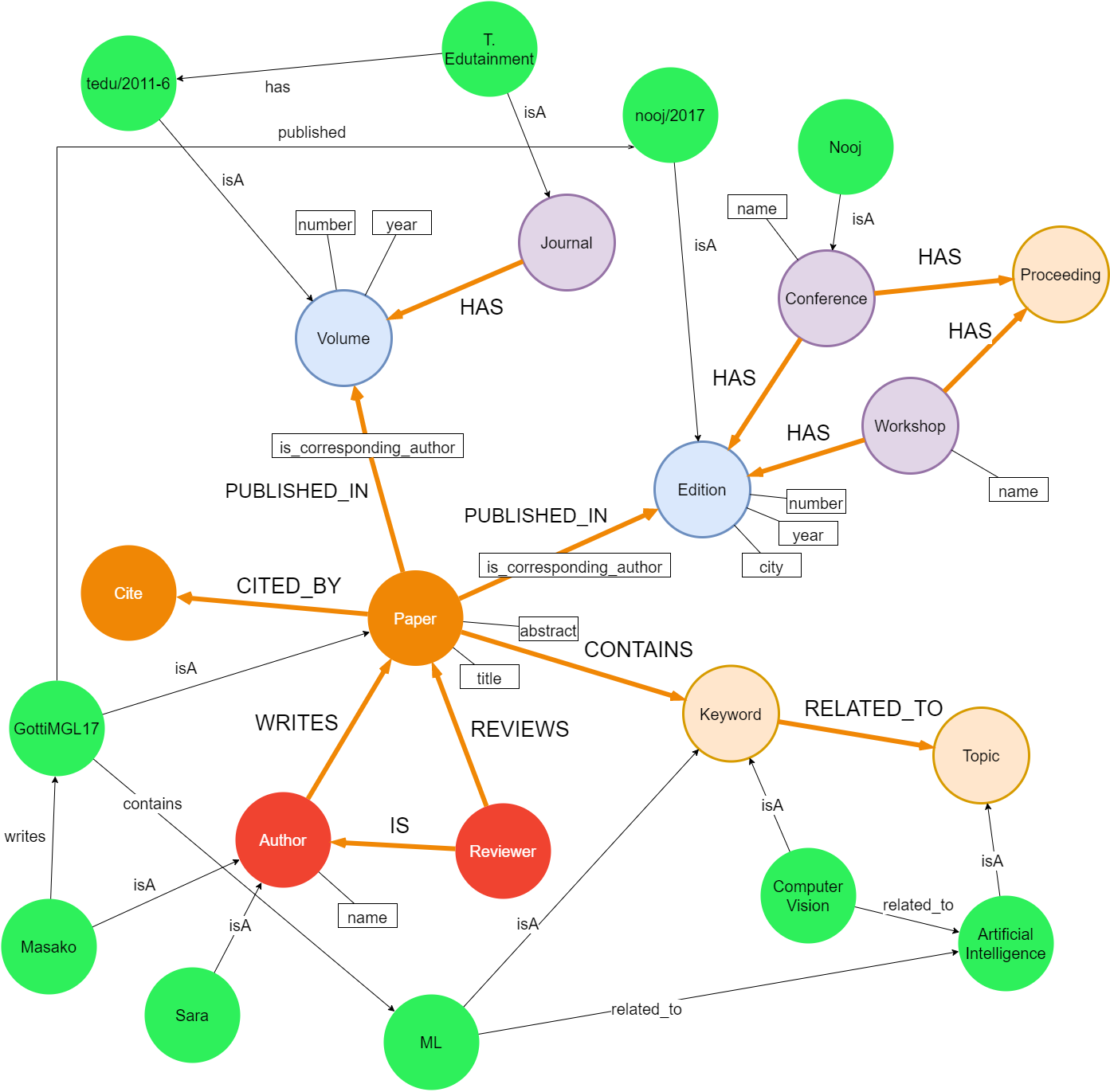
# Modeling, Loading, Evolving

## A1. Modeling



In the above graph, the green nodes are representing the data and all the other node are representing the meta-data. We have included all the meta-data and few data in our model to be easier to read. For the same reason, we do not have include all the attributes that we used while loading the real data. The attributes can be found in the loading scripts in Part B.

Regarding our design decisions, we tried to design the model in such a way that the queries in the Part B are well optimised. More specifically, we made the following decision:

* The Edition and the Volume are different nodes from the Conference and the Journal, respectively. We made this choice as most of the queries are related to Conferences (e.g. Find the top 3 most cited papers of each conference.). This means that we do not want to have the Edition as an attribute of the node Conference, as this would affect the efficiency (we would need extra I/Os).
* Similarly, Workshop and Conference are different nodes. If we had both in 1 node, we would need an extra attribute isConference (Boolean) and in this case, we would need to look up in this attribute to specify whether the node is Conference or Workshop.
* The City of the Edition of the Conference is an attribute of the Edition node, as we assume that no further analysis will be done based on the City. If we had queries that engaged the City, it would be better to have it as a node.
* A Paper may have many Keywords and a Keyword may be related to many Topics and vice versa. That is why Keyword and Topic are 2 separate nodes.
* Regarding the Citations, initially we had a self referencing edge from Paper to Paper. However, while loading the data, we realized that the dataset did not provide the information of which Paper cited each paper. The only information available was that a Paper is cited. That is why we have a separated node for Cite, which is linked to the Paper.

A.2. Instantiating/Loading:

Since there was a real dataset available (i.e. DBLP), we decided to use it for our data loading task, in order to keep our data as realistic as possible. In the same context, instead of working on the sample CSV files included in the ZIP file pointed to which from the Lab assignment, and generating artifical data to complete our model, we decided to go for real data whenever possible and data was available for the task.

As DBLP publishes its data in the form of an XML file with a corresponding DTD, we downloaded the latest one and extracted, transformed and modified the data it contains into CSV files spanning most of our data needs for the proposed model. We used the the following python script in github for this purpose:  
<https://github.com/ThomHurks/dblp-to-csv/blob/master/XMLToCSV.py>

The output of running the previous script was a Folder containing 28 csv files some of them representing content data, while others were used solely for storing headers for other CSV files.

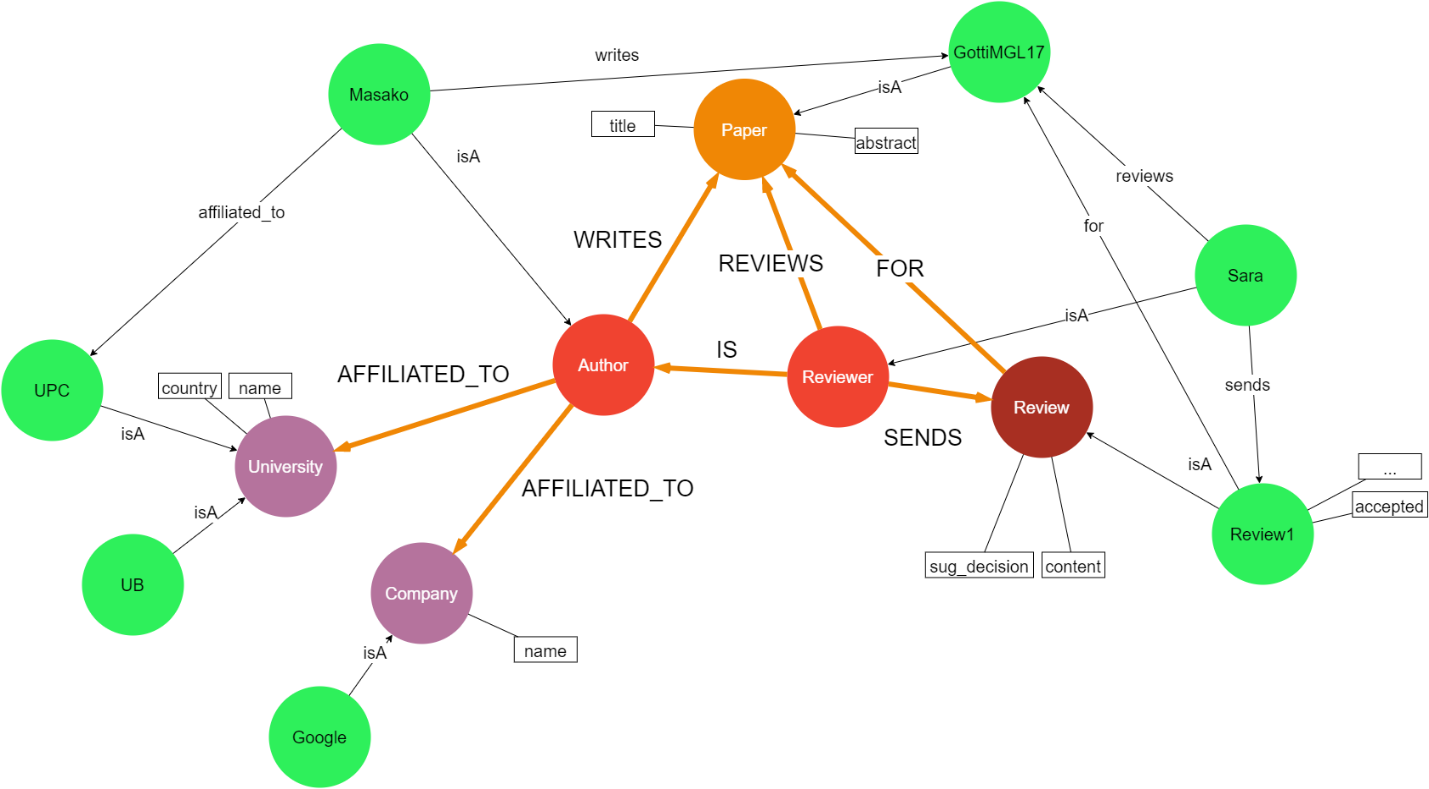
Subsequently, and for each needed file, we wrote a python script that extracts the attributes we are interested in, in our model and generates another intermediate CSV file ready to be bulk loaded in neo4j. The bulk loading scripts were written in conformance to the schema of the intermediate CSV files generated by our Python scripts. This approach helped us keeping the Cypther loading script as clean as possible. Following are the remarks on CSV files from DBLP dataset that were used for data extraction:

* **output\_proceedings.csv:** From this original file we extracted *Conference.csv*, *Editions.csv, Workshop* and *Proceedings.csv* files which represented the inputs for the Cypther bulk loading script. The relationship between Edition and Conference was created on the fly following each Edition node’s creation.
* **output\_inproceedings.csv:** From this original file along with its header file, we extracted *Papers(Edition).csv* representing conference papers and their relationships to editions. We also extracted part of *Authors.csv*  from this file, spanning the authors who wrote conference papers. Lastly the relationship file *Edition\_Paper\_Author.csv* was also extracted from here.
* **output\_article.csv:** From this original file along with its header file ,we extracted *Journal.csv, Volume(Journal).csv, Paper(Volume).csv* and part of *Authors.csv.* Taking all this information from this file, helped us store only the journals and its volumes that have published papers which was more meaningful to our model for the next tasks.

As for the missing data from DBLP, we generated the following files:

* **Cite.csv:**  this file contains 45000 artifical citations generated randomly with a Python script. The citation contains only its own identifier, and a key to the paper it cites. We here care only about how many times a certain paper was cites, with no need to keep the source paper.
* **Keyword.csv:** this file contains keywords for 10,000 papers, storing two most important keywords for each paper. Before loading this file, a Cypher initilization script was used to create a set of 12 keyword nodes. The assignment of keywords to papers was done randomly in a Python script.
* **Paper\_reviewer.csv:** This file contains the assignment of 1000 reviewers (selected randomly from Authors.csv) to 3000 papers. The assignment rule, is that no paper can be assigned more than 3 reviewers, and a reviewer can not be assigned a paper he authored. This was all achieved in a Python script.

## A.3 Evolving the graph



Regarding our modeling decision for the evolved graph, we would like to make the following notes:

* We decided to separate the Author and Reviewer nodes, are they have different semantics. However, we could also have one node for both, that would have different edges to Paper (WRITES and REVIEWS).
* The Review is connected both to Reviewer and Paper. Another option would be to connect the Review only to its Reviewer, but in this way, we would not be able to identify which Review is for which Paper, as a Reviewer may review more than one Paper.

# Querying

**Query1:** Find the h-indexes of the authors in your graph

MATCH(a:Author)-[:WRITES]->(p:Paper)-[:CITED\_BY]->(c:Citation)

WITH a as authors, p.key as papers, count(c) as number\_of\_citations

ORDER BY number\_of\_citations DESC

WITH authors as authors, collect(number\_of\_citations) as citations\_list

WITH authors as authors, citations\_list AS citations\_list

UNWIND range(0,size(citations\_list)-1) as l\_index

WITH authors as authors,

CASE

WHEN citations\_list[l\_index] >= l\_index+1 THEN l\_index+1

ELSE -1

END AS hindex

WHERE hindex <> -1

RETURN authors.name, max(hindex)

**Query2:** Find the top 3 most cited papers of each conference.

**Query3**: For each conference find its community: i.e., those authors that have published papers on that conference in, at least, 4 different editions.

MATCH(c:Conference)-[:HAS]->(e:Edition)<-[:PUBLISHED\_IN]-(p:Paper)<-[w:WRITES]-(a:Author)

WITH c as conference, a as author, count(distinct e) as number\_of\_editions

WHERE number\_of\_editions >= 4

RETURN conference.name as Conference, collect(author.name) as Community

**Query 4:** Find the impact factors of the journals in your graph