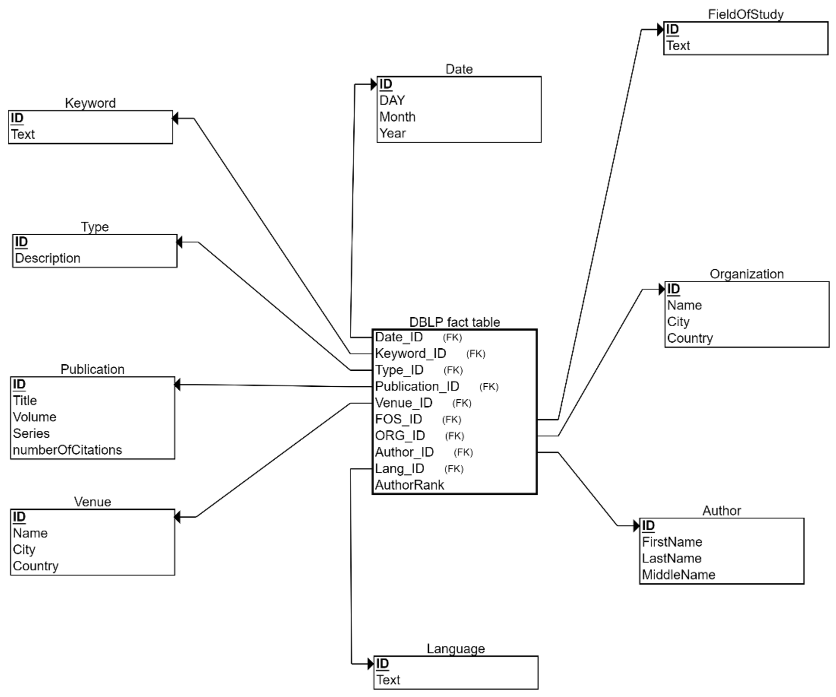
# Scientific Publications Data Warehouse

Project 1: A Data cube on top of Delta lake (ETL)

## Purpose

The purpose is to extract data about scientific publications from JSON data that describe, title, topic, authors, etc., about a large number of papers and populate a data warehouse to issue analytics queries using SQL. We will use Spark DataFrames to extract and transform the data. We will also use Spark tables (delta tables) to be used for dimensions and fact tables as will be shown below.

## DWH Schema



The above schema is a suggestion, i.e., not obligatory. You may want to change it and justify your decision. For example, the schema is optimized for less number of joins. However, in the DBLP fact table, for any paper with more than one author, the data will be repeated as many times as there are authors. Some of the columns depend on the publication, like Lang_ID, and Keyword_ID and some others depend on the author like ORG_ID, if this refers to the affiliation of the author. You might want to change the schema design to reduce redundancy.

## Data set

The data source is https://www.aminer.org/citation, version 13, as it is the most detailed one in JSON format. You can also check the schema of the respective data set on the same page under the "Description" link – note that the schema may not correspond to the schema in the JSON file.

## Extract (ETL)

The V13 data set from the link above is about 16 GB of JSON data. You will need to unzip the file, split it into smaller files, and preprocess it as a text file.

**JSON parsers commonly need JSON to have the fully correct syntax. This JSON file, however, is not conforming to common JSON syntax – it has integers listed as NumberInt(#), e.g. "year": NumberInt(2008). You need to find a way to replace such strings to make it into a conforming JSON, which Spark can parse**.

You can, at the same time as you are doing the replacement, split the file into smaller files (e.g., 250 000 JSON objects per file). This will enable you to run the initial setup faster, and you can use the different files later for setting up incremental loading. **Techniques we discuss when talking about Spark tuning can help with this part.**

Please note that if you use the Databricks Community Edition, the file size limit on DBFS is 10GB. However, you can run the initial preprocessing on the driver node's local storage.

## Transform (ETL)

The following are initial suggestions. You need to explore the data and come up with other transformation rules.

- You can drop publications with very short titles, e.g., one word, with empty authors
- Similarly, for the number of citations, you need to explore the data a bit and visualize the content of sample records.
- ISSN is sometimes filled with wrong values, e.g., supplying the journal name rather than the ISSN. You can either drop or make more effort to resolve the ISSN with the help of other attributes, e.g., DOI value in the same record.
- You can exclude forewords of conference proceedings, journals' volumes, or special issues.
- Defining the type of publication. The data set does not tell whether this is a conference, workshop, symposium, article, or book. We can have some heuristics to infer the type.
    - The description of the schema mentions the attribute "doc_type". But, it is not present in most of the records,
    - The default could be that this is a conference paper,
    - In the record.venue.raw value, if we find the @ Symbol, it usually means that this is a workshop at a conference,
    - If volume and/or issue values are not empty, then this is a journal publication.
- Resolving ambiguous author names. You can use external systems like Google scholar, DBLP, MS Research
    - We have in this repo some Python scripts that we used in a slightly similar context to resolve authors and refine their publications, https://github.com/DataSystemsGroupUT/Minaret. You can also read the paper of Minaret to help you search through the repository: https://openproceedings.org/2019/conf/edbt/EDBT19_paper_210.pdf
- Resolving ambiguous or abbreviated conference or journal names: You can use Google scholar or DBLP database
    - Google scholar APIs: https://serpapi.com/google-scholar-api
    - DBLP API: https://dblp.org/faq/How+to+use+the+dblp+search+API.html

    - You can use this also to resolve authors and publication type. Remember that this will be more costly computationally and due to data transfer over the network. So, use only when no other means is helpful
- Refining venues
    - For computer science-related publications, you can use DBLP APIs
    - You can also use Google Scholar APIs for other scientific disciplines
- Author gender: You can infer the gender from an author's first name. You can use an API like https://gender-api.com/ or can have a lookup table.
- H-index of authors, organizations, and journals. The H-index is an indicator of scholars' achievements. In simple words, it is the number of publications 'h' written by the author such that each of them received at least 'h' citations. You can check this blog for a discussion of different implementations, Fastest way to calculate h-index of publications | by Sourabh Jain | Towards Data Science. One idea in Spark is to use analytical window functions and sort the papers by their citations count. The challenge is how to partition the papers so that the paper lands in the partition of each author.

Normalization:

- Field of study: In many of the papers, the attribute "fos", the field of study, is defined. But the values seem ad-hoc. So, one way to pick relevant and normalized value for the field of study, is to have the following classification, https://confluence.egi.eu/display/EGIG/Scientific+Disciplines , as a look up table and match the paper's fos list to it and pick from the lookup table.
- (optional) Keywords: You might want to control the number of keywords as they are free form. You can drop keywords within the same publication that have overlaps. You can drop keywords that refer to cities, locations, persons etc. You can match the keywords to the normalized field of study

**NOTE: Many of the suggested APIs above might have limited free access. Therefore, you can try using them on a very small sample of the data. The objective is to learn how to enrich the data using these APIs, not to enrich ALL the data. If some APIs provide renewable free daily quotas, you can schedule daily calls on different subsets of the data. This is also a relevant exercise in data engineering.**

## Load (ETL)

From the data above, you need to extract the data for each dimension table, remove duplicates and populate the table. Then, you need to tweak the main data frame by adding the author rank column and flattening the authors, organizations, keywords, and fields of study. The author rank in a paper is his position in the author list

## Operations

We need to emulate incremental updates to our data warehouse hosted on Spark delta tables. So, you can use some of the split files to make the initial population for DW. The other files you can use for incremental updates. To trigger incremental updates, you can use Spark structured streaming features reading from folder. You need to think about your policy to handle changes in dimensional tables records. You can recall that from the Data Engineering course or refer to online resources about slowly changing dimensions.

## Queries

You need to come up with queries against the DW schema like querying the top-k authors, based on e.g. their H-index that publish in a specific fos.