Project Proposal

Gabriel Guerra Trigo (ggt2112) & Ritwik Goel (rg3546)

1. We aim to make a website that allows users to explore scientific literature and perform literature review through browsing entities and relationships such as papers, authors, publications, institutions, co-authorship & citation relationship graphs, etc.

2. We will do the web front-end option. Users will interact with our application by searching for papers/authors, and our website will provide papers in the paper's local citation network (papers that cite/are cited by the paper), and authors in the author’s local co-authorship network (authors that have co-authored with the author).

3. Our website will also show more data for papers/authors such as author affiliation, paper publication date, abstract, url, venue, and DOI identifiers.

4. Users will be able to create “collections” of papers, which will show up on their profile. The complete ER diagram for our project can be seen in the "ER Diagram Section".

Data Plan

1. We plan to populate our database with data from datasets provided by the Semantic Scholar API. We already requested (and were granted) an API key, which allows us to download and use this data.

2. These datasets contain large amounts of data for most of the entities/relationship sets in our scope, which we will parse to extract the fields and populate our database.

3. We may enrich this data with additional "fake attributes" (generated by us) in case they don't have some information we would like to show.

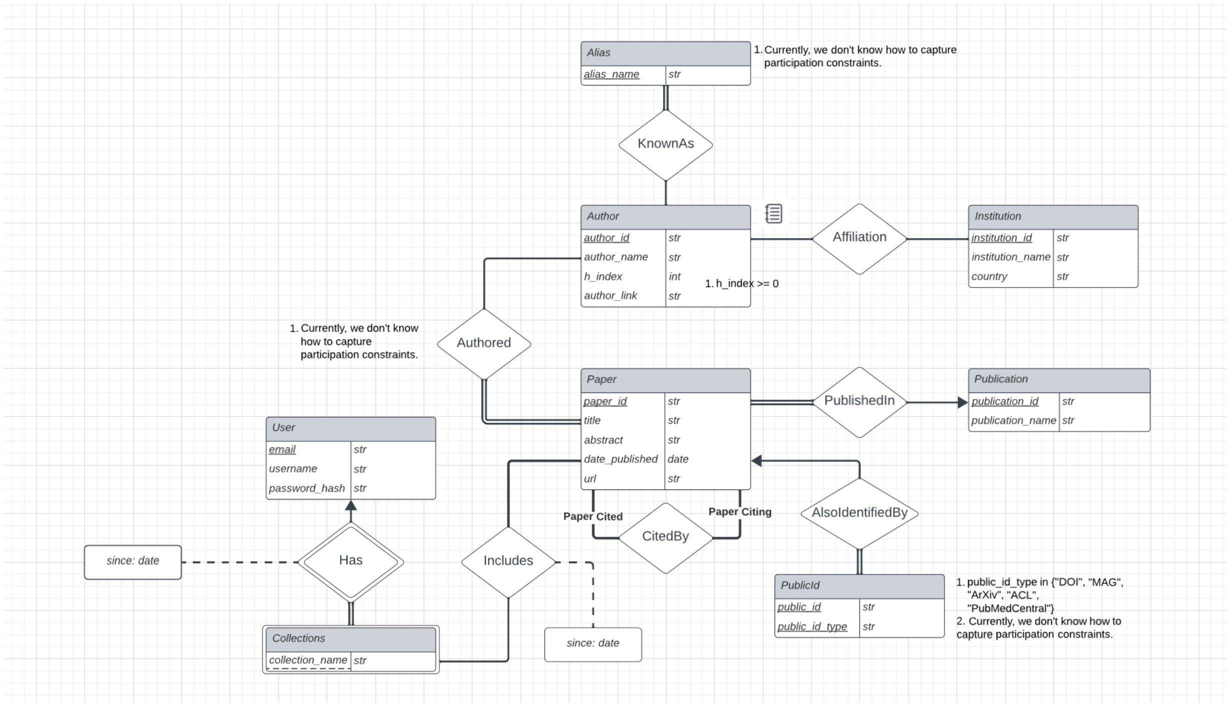
4. We might not use the entire datasets, as they are very large (>50 Gb) and may go

beyond our database resources.

5. Website user data (usernames, emails...) will be invented manually, or generated randomly.

6. We might refine the ER diagram while we parse the data, in case we find an integrity constrait that wasn't added, or realize we added an integrity constraint that the data does not obey. The ER diagram we have constructed is based on our understanding of what the constraints should be for the data in the real-life scenarios, and not on whether or not these constraints are obeyed in the datasets.

ER Diagram



Schema

In [ ]:
```
"""
    CREATE TABLE Alias(
        alias_name VARCHAR(50)
        PRIMARY KEY(alias_name)
    )

    CREATE TABLE KnownAs(
        author_id VARCHAR(10)
        alias_name VARCHAR(50)
        PRIMARY KEY(author_id, alias_name)
        FOREIGN KEY(author_id) REFERENCES Author
            ON DELETION CASCADE
            ON UPDATE CASCADE
        FOREIGN KEY(alias_name) REFERENCES Alias
            ON DELETION CASCADE
            ON UPDATE CASCADE
    )
    ***: Alias has a participation contraint on
            KnownAs, but we don't know how to model
            that yet.

    CREATE TABLE Author(
        author_id VARCHAR(10)
        author_name VARCHAR(50)
        h_index INTEGER
        author_link TEXT
        PRIMARY KEY(author_id)
    )
    ***: h-index >= 0.

    CREATE TABLE Authored(
        author_id VARCHAR(10)
        paper_id VARCHAR(10)
        PRIMARY KEY(author_id, paper_id)
        FOREIGN KEY(author_id) REFERENCES Author
            ON DELETE NO ACTION
            ON UPDATE CASCADE
        FOREIGN KEY(paper_id) REFERENCES Paper
            ON DELETION CASCADE
            ON UPDATE CASCADE
    )
    ***: Paper has a participation contraint on
            Authored, but we don't know how to model
            that yet.

    CREATE TABLE Paper(
        paper_id VARCHAR(10)
        title TEXT
        abstract TEXT
        date_published DATE
        url TEXT
        publication_id VARCHAR(10) NOT NULL
        PRIMARY KEY(paper_id)
        FOREIGN KEY(publication_id) REFERENCES Publication
            ON DELETE NO ACTION
            ON UPDATE CASCADE
    )
```

```
CREATE TABLE Publication(
    publication_id VARCHAR(10)
    publication_name VARCHAR(50),
    PRIMARY KEY(publication_id)
)

CREATE TABLE CitedBy(
    paper_cited VARCHAR(10)
    paper_citing VARCHAR(10),
    PRIMARY KEY(paper_cited, paper_citing)
)

CREATE TABLE AlsoIdentifiedByPublicId(
    public_id VARCHAR(10)
    public_id_type VARCHAR(20)
    paper_id VARCHAR(10) NOT NULL,
    PRIMARY KEY(public_id, public_id_type, paper_id),
    FOREIGN KEY(paper_id) REFERENCES Paper
        ON DELETE CASCADE
        ON UPDATE CASCADE
)

CREATE TABLE Includes(
    collection_name VARCHAR(20)
    paper_id VARCHAR(10)
    email VARCHAR(50)
    since DATE
    PRIMARY KEY(collection_name, email, paper_id)
    FOREIGN KEY(collection_name, email) REFERENCES HasCollection
        ON DELETE CASCADE
        ON UPDATE CASCADE
    FOREIGN KEY(paper_id) REFERENCES Paper
        ON DELETE CASCADE
        ON UPDATE CASCADE
)

CREATE TABLE User(
    email VARCHAR(50)
    username VARCHAR(20)
    password_hash CHAR(256),
    PRIMARY KEY(email)
)

CREATE HasCollection(
    collection_name VARCHAR(20)
    email VARCHAR(50)
    since DATE,
    PRIMARY KEY(collection_name, email),
    FOREIGN KEY(email) REFERENCES User
```

```
            ON DELETE CASCADE
            ON UPDATE CASCADE
    )
"""
```

Out[ ]:
```
'\n    ids: VARCHAR(10)\n    names: VARCHAR(50)\n    email: VARCHAR(50)\n    
username/collection_name: VARCHAR(20)\n    public_id_type: VARCHAR(20)\n    
password_hash: CHAR(256)\n'
```

In [ ]:
```
"""
    These are just some annotations for us to organize
        ourselves. We opted to have ids be up to 10 characters,
        for names to be up to 50, and so forth. These are just
        some arbitrary decisions, and ARE NOT PART OF THE SCHEMA.
    ids: VARCHAR(10)
    names: VARCHAR(50)
    email: VARCHAR(50)
    username/collection_name: VARCHAR(20)
    public_id_type: VARCHAR(20)
    password_hash: CHAR(256)
"""
```

Out[ ]:
```
'\n    These are just some annotations for us to organize\n        ourselve
s. We opted to have ids be up to 10 characters,\n        for names to be up
to 50, and so forth. These are just\n        some arbitrary decisions, and
ARE NOT PART OF THE SCHEMA.\n    ids: VARCHAR(10)\n    names: VARCHAR(50)\n
email: VARCHAR(50)\n    username/collection_name: VARCHAR(20)\n    public_i
d_type: VARCHAR(20)\n    password_hash: CHAR(256)\n'
```

Contingency Plan

We are very confident that neither of us will drop the class, so we don't have a contingency plan. We understand the risks, and accept the possible consequences: if any of us drops the class, than the other will just complete the entire project.