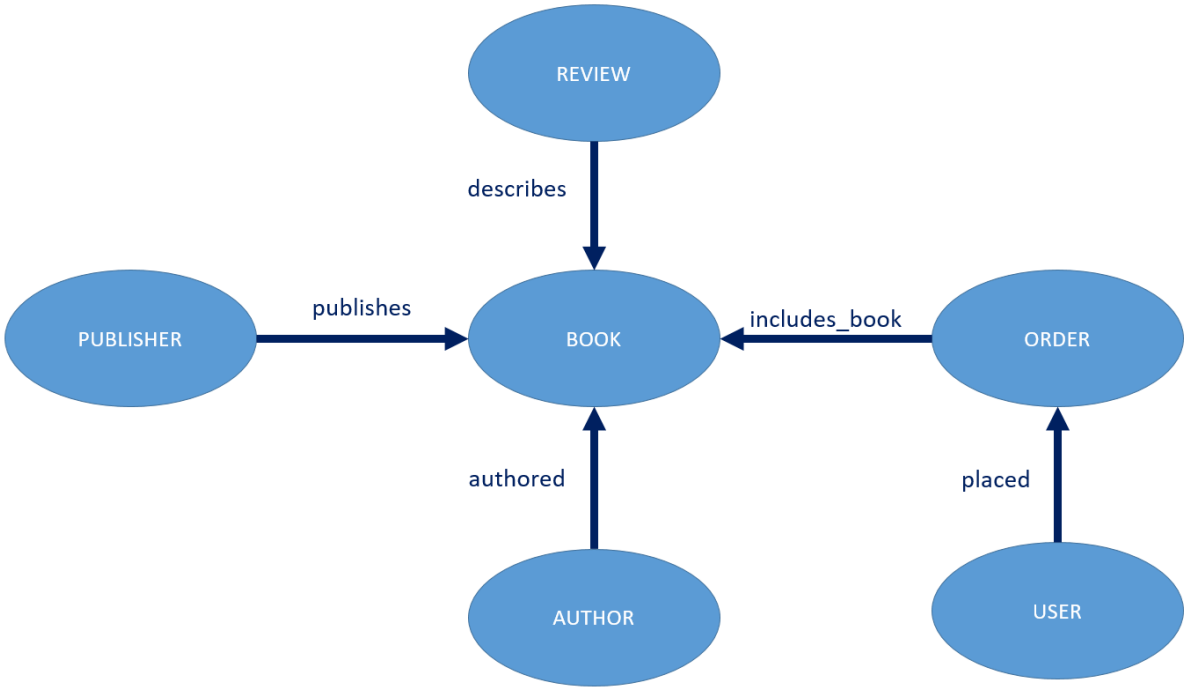
SQL Schema and Data:

In order to complete this project, some adjustments were necessary to the PostgreSQL database that was created in the previous one. Some data that were not included in the original database were generated (e.g. Publisher Country). Minor adjustments were required for the schema, namely to increase the maximum length of some text fields of variable length in order to account for the new generated data:

- Publisher - country: from VARCHAR(30) to VARCHAR(100)
- Author – nationality: from VARCHAR(15) to VARCHAR(100)
- User – password : from VARCHAR(20) to VARCHAR(50)
- User – email: from varchar(30) to VARCHAR(50)
- User-Name from varchar(30) to Varchar(50)

The adjusted schema and the new data are provided in the files “2005-newschema.sql” and “2005-newdata.sql”.

Graph Database Schema:



The relations between the different nodes closely resemble the FOREIGN KEY constraints of the SQL database. We will use the existence of these FKs to establish the relationships. However, there is no need for intermediate tables to handle the relationships with cardinality M:N, such as authors-books. Each author node is connected with all nodes of the books that she authored, while each book node is also connected with all authors that contributed to its creation. It’s worth noting that the order nodes could be instead modelled as relationships between user nodes and book nodes. However, by following the above schema, it’s easier to model orders that contain more than one books.

In order to create the database, we need to export our data in 10 .csv files, 6 to create the nodes that represent the main entities and 4 bridging files to create the relationships. We use these bridging files

in order to avoid storing the foreign keys in our database. For instance, we don't want the publisher nodes to have an attribute named 'Book_isbn'. Instead, we create a bridging file that contains the combinations between the two primary keys, 'ISBN' and 'Publisher_id', in order to use this mapping to create the relationship 'publishes' between the publisher nodes and the book nodes. The only exception is the case of the order nodes, where storing the ISBN of the book that was ordered is a requirement. Consequently, the relationship 'placed' can be created without a bridging file.

Data:

Importing all data from PostrgreSQL to Neo4j was extremely heavy computationally for my personal laptop, especially the establishment of the relations between the nodes. In order to address that, only a portion of the initial data were exported.

Entity	Number of Records
Book	60000
Publisher	6000
Author	15000
Review	10000
User	30
Order	250

The book nodes are central in our database. Therefore, the entirety of the book nodes was exported to avoid inconsistencies.