Manipulation of the ‘UCSD Book Graph’ dataset and importing to the DB

The provided data set was received in the form of 3 JSON files. The data required a notable amount of pre – processing in order to be ready for import to the database. A lot of missing values were present, often in the form of empty strings (‘’). Moreover, much information was not relevant to our application and had to be filtered out. Finally, a significant number of existing fields had to be reformed in order to match our application’s scope and requirements. The data was processed with the python parser and an sql file with import statements was created. In order to balance memory and speed requirements, the insert statements were executed in batches of size 1000 – 2000.

Some of the aforementioned issues were handled at the pre – processing level, namely within the code of the python parser, while others were handled at the data base level through the use of import triggers. It’s important to note that a number of triggers were necessary specifically to handle the large import statements that occurred at this stage of the data base development, while others would also serve in the future. We will now go through the different tables and describe the necessary data pre – processing and the associated data base triggers that were used to populate each table.

“PUBLISHER”:

- For this relation, only the Publisher_Name attribute was available in the field ‘Publisher’ in the comic books JSON file. The field was stripped from quotes, both single (‘ ‘) and double (“ “). Publisher fields that had empty strings were dropped and not included in the database.
- Only unique publishers were selected and a UUID was randomly generated for each publisher and stored in the attribute of ‘PublisherID’ in the ‘PUBLISHER’ relation.
- The other attributes of PUBLISHER were not included in our data set. Given that they don’t seem that important for the data base functionality, they were left as NULLs.
- The phone number was handled as VARCHAR(15) to account for different number of digits and specific domains – regional digit combinations for each country.

“BOOK”:

- The ISBN is considered the most important attribute for each book, given that it’s a unique identifier. Two relevant fields were present. ISBN and ISBN13. Books with no ISBN and no ISBN13 were aborted. When possible, ISBN13 was used to fil the ISBN field. Finally, the ISBN13 field was dropped. Moreover, some duplicate values were identified in the ISBN field. These were sorted out by an import trigger, that ignored every incoming tuple with a pre – existing ISBN, raised a notice and continued with the rest of the batch.
- As stated earlier, the Title of the book was not allowed to be null, while in cases of books with titles longer than 200 characters, all characters beyond 201 were removed by the parser.
- The attributes ‘Description’ and ‘Title’ of the BOOK relation were stripped from quotes, both single (‘ ’) and double ones (“ ”).
- Empty string (‘ ‘) publication year imports were set to NULL through an insert trigger.
- Empty string Descriptions were replaced with the string ‘There is no available description for this book’ through an insert trigger, while the default value of the field was also set to the same string.
- Publication year strings longer than 4 characters were set to NULL by the parser.
- The prices of the books were not included in the data set, but were generated randomly by the parser through sampling from a normal distribution with mean = 70 euro and standard deviation = 20 euro. The min book price was set to 10 for all books.

- The publisherID foreign key was identified through the use of a python dictionary that created a mapping between unique publisher names and the unique UUIDs that were generated for the PUBLISHER table. If the publisher field in the data set was an empty string, the ‘PublisherID’ attribute of the corresponding book was set to NULL by the parser.
- The fields ‘book_id’ and ‘author_id’ were also deemed important but were not included in the table. They were collected for later use to join reviews with books and authors with books respectively.

“AUTHOR”:

- For each author, only the ‘Author_ID’ and ‘Name’ fields were collected from the dataset and stored in a pandas dataframe. Moreover, a python dictionary was created to map each unique author_id from the data set to a newly generated UUID that will serve as a primary key for the AUTHOR table. Through the use of this dictionary, the join between books and authors became possible.
- Author names longer than 100 characters were stripped by all characters beyond 100. Finally, all names were stripped from quotes, both single (‘ ’) and double ones (“ ”).
- The ‘Gender’ and ‘Nationality’ attributes were not included in the data set and were also not deemed important. Therefore, they were left empty (NULL).

“CREATION”:

- This table serves as an intermediate table to join BOOKS and AUTHORS. The role of each author was an extra attribute that described the process of book creation. Each book in the data set had information about the authors involved in its creation in the form of a list of dictionaries: [{"author_id": "value", "role": ""}, {"author_id": "value", "role": ""}, …]. By utilising the {author_id: author UUID} dictionary that was created earlier, the “ISBN” attribute of each book was matched with the “AUTHOR_ID” (UUIDs) of the corresponding authors, as well as the involved roles.
- An insert trigger was used to identify combinations of ISBN and Author_ID that already existed in the “CREATION” table, ignore the associated tuples and continue with the rest of the batch import.
- An additional insert trigger was created to ignore import tuples with ‘Author_Role’ length greater than 100 characters.

“REVIEW”:

- The review_id field of the data set was not used. A new UUID was generated to serve as the primary key of the table in the “ReviewID” attribute.
- The ‘date_added’ field was manipulated to the correct timestamp format by the parser to serve as the ‘Review_Timestamp’.
- The ‘review_text’ field was stripped from quotes, both single (‘ ’) and double ones (“ ”), and was then imported in the ‘Review_Body’ attribute.
- The ‘rating’ field was rounded to an integer to feed the ‘Score’ attribute. Values smaller than 1 or greater than 5 were set to NULL by the parser.
- In order to connect this table to the ‘BOOK’ table, another dictionary was formed using the data from the fields of each book. Specifically, we created a mapping between ‘Book_id’ and ‘ISBN’. This allowed as to add the ISBN of each book as a foreign key to this table.
- The ‘Nickname’ attribute was not present in the data set and it was left empty (NULL).

The rest of the tables: “USER”, “USER ADDRESS” and “ORDER”

- These three tables aren’t related to the information of our data set, but will be rather populated during the e – shop operation. In order to present a more realistic version of the database, a very small number of users and orders was manually created and imported in the database.
- This generated data can be found in the excel sheet “2005_Comic_Book_Users_Orders.xlsx” that is included in this deliverable.