Method for creating a database in SQL for Seal Images

1. The first step is to create the foundation of the data structure and that is to implement it in the tables, using the **all\_tables.sql** (Delete picture\_number constraint in case of ERROR)
   1. Import **data\_tags.csv** into the **data\_tags** table, it is important that only the specified columns get imported.
   2. Import **link\_path.csv** into the **data\_reference** table, link\_path is the only column selected.
   3. Import **Uncertainties.csv** into the **uncertainty** table, all columns should be selected except for picture\_number.
2. Now that the data has been imported the next step is to implement the picture\_numbers relating all the tables together:
   1. Import all FUNCTIONS
      1. See table below, also include the **generate** functions
   2. Select the query **picture\_number\_column\_generator.sql** to generate a unique picture-number column in **data\_tags.**
   3. Select the query **picture\_number\_from\_data\_tags\_to\_data\_reference.sql** to transfer the picture\_number column to **data\_reference.**
   4. Select the query **picture\_number\_from\_data\_tags\_to\_uncertainty.sql** to transfer picture\_number column to **uncertainty.**
   5. After all the rows have been added there are 3 files that need to be removed using the **remove\_rows.sql** query.
   6. Run the query **insert\_data\_reference\_oid.sql**, transcribing the JPEG files into OID files and then inserting the files to the corresponding link\_path.
   7. Run the query **insert\_scrape\_names.sql**, which will iterate through the rows In **data\_tags** and concatenate all the tags within the rows with picture numbers.
   8. There is a folder called **Table adjustments** in the **Tables** folder, this will only be important when import doesn’t work properly.

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Argument | Output | Explanation |
| Import and Export Functions: ().sql | | | |
| import\_image | p\_image\_path (TEXT),  p\_data\_tags (TEXT[]),  p\_data\_uncertainties  (BOOLEAN[]) | insert\_data\_tags().sql  insert\_data\_references().sql  insert\_data\_uncertainty().sql | This parent function takes in the required arguments for the new image to be imported and sorted into the data structures using insert functions. |
| export\_image | p\_picture\_number INTEGER,  p\_output\_directory TEXT | lo\_export() | This is technically not a parent function to any of the insert functions but outputs a JPEG file using built-in export functions |
| Insert Functions | | | |
| insert\_data\_tags | p\_data\_tags TEXT[],  p\_picture\_number INTEGER,  p\_scrape\_names TEXT | INSERT INTO:  data\_tags | This child function is called in import\_image, and fills the data\_tags table |
| insert\_data\_reference | P\_picture\_number1  INTEGER,  p\_file\_path1 TEXT,  p\_stored\_image\_oid OID | INSERT INTO:  data\_reference | This child function inserts the file\_path to the OID being used and stores it into the data\_references |
| insert\_data\_uncertainty | p\_data\_uncertainty  BOOLEAN[],  picture\_number1  INTEGER | INSERT INTO:  uncertainty | This child function inserts a row for uncertainties with the corresponding picture\_number |
| insert\_user\_data | p\_username1  VARCHAR | uuid\_generate\_v4()  generate\_random\_number\_5\_digits().sql  INSERT INTO:  User\_data | This child function takes the newly created user information and adds it to the relational table user\_data |
| perform\_user\_action\_with\_documentation | p\_username  VARCHAR,  p\_action\_type  VARCHAR,  p\_affected\_picture\_number  INTEGER | INSERT INTO:  User\_documentation  SELECT FROM:  user\_data | This insert function serves to document all the edits being made by the users to the tables |

There are a lot of modifications that can be done to the database to make it more efficient as well as make it more manageable. One topic that definitely would warrant further research would be the usage of a database backup function that would save table iterations, this could be implemented in data\_documentation, and the backed-up tables can be saved in a specific filesystem designed for backups and which can also be accessed by the database itself. Allowing for backups to be made would allow the database to visit previously saved files as well as restore and compare different file structures. Another system that would be useful for implementation would be creating file categories by creating a function that takes the file path but stores it in a specific folder about the specimen, this would allow for easier and quicker data extraction as the database increases in size. An improvement on the database currently would be removing the need to store the OID in the data\_reference itself and storing it instead in a filesystem which greatly reduces the burden on the database and would work much more efficiently.