Report: Phase-2: Implementation of BioJoin Basic

BiS332 course by Professor Doheon Lee

Lana Abu Hassan (20170842), Surayouth Phuksawattanachai (20180813), Alexander Semenov (20226159)

Introduction

This project phase aims to create a database with data about SNP, genes, and disease information from primary bioinformatician databases. Namely, the databases are SNP, EntrezGene, and OMIM databases. For that reason, in phase one, we designed the database architecture together with the ER diagram that will guide us in creating, manipulating, and searching for records in database tables for this phase.

In addition, we should implement software that works on top of the database and helps the user to operate the database content. In particular, the software should allow the user to perform basic database operations such as search, update, and deletion.
More specific, in this project phase, we are expected to fulfill the following tasks:

1. Table creation
2. Record insertion from data files
3. Record update on the DB tables
4. Record deletion from the DB tables
5. Record search on the DB tables
    a. Given a gene symbol, find all gene information stored in the gene table
    b. Given a chromosome id, find all gene symbols located in the chromosome
    c. Given a SNP ID, find all diseases associated with the SNP
    d. Given a disease name, find all SNP IDs associated with the disease

Using the skills we learned and compiled during class, we can design and implement an integrative bio-information inference system using the databases for us. As well as giving our result interface and code, we will also explain and justify our reasons for choosing this specific way of implementation.

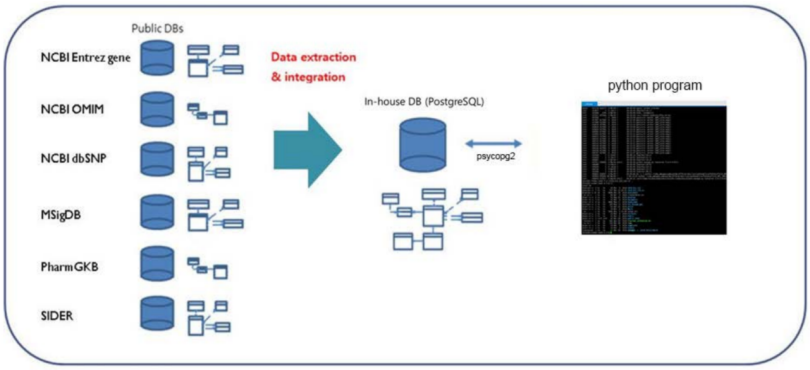


Figure 1: Overview of the project

Tools

We decided to work on this project using the following platforms and libraries:

- Python 3.8.13
- Psycopg2 (PostgreSQL adapter for the Python programming language)
- Pandas

More python native build-in modules and libraries are used e.g. json, io

Database Architecture

Given raw data files

TA's provided us with some raw data files from primary bioinformatician databases. Namely, OMIM.txt, gene_OMIM.txt, Homo_sapiens_gene_info.txt and SNP.txt files. All the files are tab-separated csv files with a given header (see examples below).

```
SNP.txt
data > SNP.txt
SNP id  Chromosome  Position of SNP on the chromosome   Genes at the same position on the chromosome   Ancestral allele   Minor allele
538 1   6100898 KCNAB2  C   A
546 1   93151989    TMED5   C   T
665 1   23854551    FUCA1   G   T
699 1   230710048   AGT C   A
751 1   87392286    LOC105378833    G   A
836 1   160998434   F11R    G   A
844 1   161677743   FCGR2B  G   A
873 1   235977439   NID1    C   A
898 1   78537573    PTGFR   A   C
```

Figure 2: Example of SNP information text file - SNP.txt

```
Homo_sapiens_gene_info.txt
data > Homo_sapiens_gene_info.txt
#tax_id GeneID  Symbol  Synonyms    chromosome  map_location    description type_of_gene    Modification_date
9606    1   A1BG    A1B|ABG|GAB|HYST2477    19  19q13.43    alpha-1-B glycoprotein  protein-coding  20170402
9606    2   A2M A2MD|CPAMD5|FWP007|S863-7   12  12p13.31    alpha-2-macroglobulin   protein-coding  20170402
9606    3   A2MP1   A2MP    12  12p13.31    alpha-2-macroglobulin pseudogene 1  pseudo  20170402
9606    9   NAT1    AAC1|MNAT|NAT-1|NATI    8   8p22    N-acetyltransferase 1   protein-coding  20170403
9606    10  NAT2    AAC2|NAT-2|PNAT 8   8p22    N-acetyltransferase 2   protein-coding  20170402
9606    11  NATP    AACP|NATP1  8   8p22    N-acetyltransferase pseudogene  pseudo  20170312
9606    12  SERPINA3    AACT|ACT|GIG24|GIG25    14  14q32.13    serpin family A member 3    protein-coding  20170403
9606    13  AADAC   CES5A1|DAC  3   3q25.1  arylacetamide deacetylase   protein-coding  20170402
9606    14  AAMP    -   2   2q35    angio associated migratory cell protein protein-coding  20170403
```

Figure 3: Example of Gene information text file - Homo_sapiens_gene_info.txt

```
disease_OMIM.txt
data > disease_OMIM.txt
disease_OMIM_ID disease_name
1   $Deafness, Y-linke$
10  46XY sex reversa
100100  Prune belly syndrome
100300  Adams-Oliver syndrome 1
100800  Achondroplasia
101000  $Neurofibromatosis, type 2$
101200  Apert syndrome
101400  Saethre-Chotzen syndrome
101600  Craniofacial-skeletal-dermatologic dysplasia
101800  $Acrodysostosis 1, with or without hormone resistance$
101900  Acrokeratosis verruciformis
102200  $Acromegaly, somatic$
102370  Acromicric dysplasia
102500  Hajdu-Cheney syndrome
```

```
gene_OMIM.txt
data > gene_OMIM.txt
gene_symbol disease_OMIM_ID
1C7 609148
2E4 615637
3H11AG  610188
3H11AG  610189
3H11AG  611134
3H11AG  611755
3H11AG  615991
3M1 273750
3M2 612921
3M3 614205
3MC1    257920
3MC2    265050
3PK 617111
8D6 613646
```

Figure 4: Example of OMIM information text file - disease_OMIM.txt and gene_OMIM.txt

Tables creation

As said in the beginning we have used the structure of the database from our first report. Nevertheless, after exploring the data we got we had to slightly adjust our table's attributes slightly. Thus, we have updated the SQL statement for tables creation:

dbSNP

snp_id {PK}
snp_type
snp_chr
snp_pos
snp_orient
snp_alleles
snp_min_allele_f
gene_symb

```
CREATE TABLE dbSNP (
 snp_id int NOT NULL,
 snp_chr varchar,
 snp_pos varchar,
 gene_symb varchar,
 anc_allele varchar,
 min_allele varchar,
 PRIMARY KEY (snp_id)
);
```

Removed: snp_type/snp_orient/

Added: None

Edited: snp_chr (from integer to varchar)

Gene

gene_symb {PK}
gene_id
gene_type
gene_sum
gene_chr
gene_pos
gene_org
gene_mod_date
gene_syn
snp_id {FK}

```
CREATE TABLE Gene (
 tax_id varchar,
 gene_id int UNIQUE,
 gene_symb varchar NOT NULL,
 gene_syn varchar,
 gene_chr varchar,
 gene_pos varchar,
 gene_sum text,
 gene_type varchar,
 gene_mod_date date NOT NULL,
 PRIMARY KEY (gene_id)
);
```

Removed: None

Added: tax_id

Edited: gene_chr (from integer to varchar)

OMIM

omim_id {PK}
omim_name
omim_sum
omim_chr
gene_symb {FK}

```
CREATE TABLE OMIM (
 omim_id int,
 omim_name varchar,
 gene_symb varchar,
 PRIMARY KEY (omim_id, gene_symb)
 );
```

Removed: omim_sum & omim_chr

Added: None

Edited: None

Most of the columns were removed because of a lack of information in the data files. For some reasons (e.g., gene_chr), the datatype had to be changed. In gene_chr, the datatype was changed from integer to string because of a possible X chromosome. Next, we decided to split the major and minor alleles into two columns, as it is provided in the SNP.txt file. We explain this

decision that searching in the database based on allele type would be impossible with our initial solution “major_allele”>”minor_allele.”
In addition, we had to remove some of the foreign keys because of the fact that some gene symbols from the file gene_OMIM.txt and SNP.txt were not available in the primary gene table Homo_sapiens_gene_info.txt.

Database creation and filling

The database tables have to be created manually using the above-provided SQL statements. This step is a prerequisite for using the software to operate the database.

The python script fill.py connects to the database using the configuration data such as database username, password, host, and name from the config.json file. After a preprocessing step, the data from the tables is inserted into the database tables (the tables should have already been created in the database manually). An essential step in the preprocessing is merging the two files gene_OMIM.txt and disease_OMIM.txt with an outer join method. This step is crucial since we have a relational table in our database with attributes omim_id int, omim_name, and gene_symb. The join result tsv file is merged.txt, which is then directly used to insert the data into the omim table.

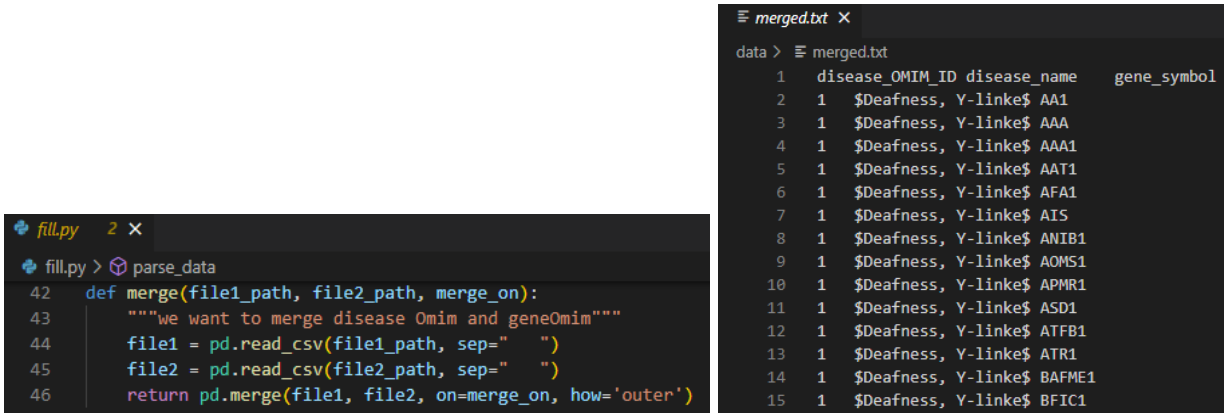


Figure 5: Example of OMIM information text file - merged.txt with its corresponding merge function

In the end, Homo_sapiens_gene_info.txt was inserted into the Gene table, SNP.txt into dbSNP and merged.txt into the OMIM table of our database.

Database manipulation software

Overview

As part of our project we implemented a program that allows user to search, modify and delete entries in the database. The programm starts by executing the cmdline.py file. The program then starts enquiring the users with a command-line interface which allows them to choose the

action they want to proceed with. The user may search, update and delete entries in the created database which was explained in the previous part.

```
cmdline
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 1
Searching action...
Available tables: ['student', 'professor', 'dbsnp', 'gene']
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: gene
Please choose between a, b or q
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: a
What template do you want to try out?

                            1. Find all gene information
                            2. Find all gene symbols located in the chromosome
                            3. Find all diseases associated with the SNP
                            4. Find all SNP IDs associated with the disease
                            q. Quit

Enter your choice: 3
```

Figure 6: Example of Command-line Interface

Prerequisites

There is some software needed to be installed on your computer before you can run the program. Please refer to the tools section. You can install those python libraries by running
→ *pip install -r requirements.txt*
And, as expected, a tables in the database with same structure has to be created and its credentials filled in the config.json file.

```
{} config.json M
{} config.json > ...
1  {
2      "db_host" : "biostar.kaist.ac.kr",
3      "db_name" : "u20180813",
4      "db_user" : "u20180813",
5      "db_pw": "bipro20180813"
6  }
```

Figure 6: Example of config.json file

Usage of the software

Record Insertion on the DB tables

In order to insert the data from files we may proceed as the following input:

```
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 2
********************
Update action...
********************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
What table do you want to update in: omim
What do you want to update?

                        a. Modify a row
                        b. Add a row
                        q. Quit

Enter your choice: b
Available cols: ['omim_id', 'omim_name', 'gene_symb']
Please provide the new values
omim_id: 2
omim_name: headache
gene_symb: AAA
Table was updated
------------------
```

Figure 7: Example of inertion action

By choosing action ‘2. Update the database’, we may then proceed to insert the data in the database. In this case, I choose to insert an OMIM with the info of [‘omim_id’,’omim_name’, ‘gene_symb’] = [2, ‘headache’, ‘AAA’]. As a result, the particular OMIM is inserted into the database, which can be checked with sql statement as followed:

Record Search on the DB tables

```
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 1
********************
Search action...
********************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: b
What table do you want to search in: omim
Available cols: ['omim_id', 'omim_name', 'gene_symb']
What do you want to search? Please provide in SQL format: omim_name = 'headache';
[(2, 'headache', 'AAA')]
------------------
Do you want to continue?
            y - yes
            n, q - no, quit
```

Figure 8: Example of search action

By selecting ‘1. Search in the database’, we may be able to make our own search query in the omim table and search for the omim with omim_name = ‘headache’. As a result, we found the entry we inserted in the previous step: (2, ‘headache’, ‘AAA’). Hence, this process confirms our insertion of the entry.

Record Update on the DB tables

As we have inserted a new entry in the previous part, we will update the particular entry from files by inputting the input as followed:

```
------------------
Do you want to continue?
            y - yes
                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 2
********************
Update action...
********************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
What table do you want to update in: omim
What do you want to update?

                        a. Modify a row
                        b. Add a row
                        q. Quit

Enter your choice: a
Available cols: ['omim_id', 'omim_name', 'gene_symb']
What do you want to modify? Please provide in SQL format: omim_name = 'headache';
Please provide the new values
omim_id: 3
omim_name: headache
gene_symb: AAA
UPDATE omim SET omim_id=3, omim_name='headache', gene_symb='AAA' WHERE omim_name = 'headache';
Table was updated
------------------
```

Figure 9: Example of update action

We start by choosing ‘2. Update the database’, in order to update the entry, and continue by choosing the specific entry with the omim_name ‘headache’. After that, we then modify the omim_id from omim_id = 2 to omim_id = 3.

```
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 1
*******************
Search action...
*******************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: b
What table do you want to search in: omim
Available cols: ['omim_id', 'omim_name', 'gene_symb']
What do you want to search? Please provide in SQL format: omim_name = 'headache';
[(3, 'headache', 'AAA')]
------------------
Do you want to continue?
            y - yes
            n, q - no, quit
```

Figure 10: Confirmation of successful update of entry

To confirm the updating process, we search for the particular entry and check for its information. As a result, we found that the entry has changed from (2, ‘headache’, ‘AAA’) into (3, ‘headache’, ‘AAA’) as we predicted, thus confirming the entry updating process.

Record Deletion on the DB tables

In deletion part, we may proceed by choosing input as followed:

```
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 3
*******************
Delete action...
*******************
Do you want to delete table or entry in table?

                        a. Entry in table
                        b. Whole table
                        q. Quit

Enter your choice: a
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
What table do you want to delete in: omim
Available cols: ['omim_id', 'omim_name', 'gene_symb']
What do you want to search? Please provide in SQL format: omim_name = 'headache';
Deleting in omim with condition
omim_name = 'headache';
An entry in table was deleted
------------------
Do you want to continue?
            y - yes
            n, q - no, quit
```

Figure 11: Example of deletion action

By choosing option ‘3. Delete from the database’, we can then choose to delete ‘a. Entry in table’ and choose the omim table as the entry we inserted before is located. After that, we search for the entry with omim_name = ‘headache’ to delete the corresponding entry.

```
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 1
*******************
Search action...
*******************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: b
What table do you want to search in: omim
Available cols: ['omim_id', 'omim_name', 'gene_symb']
What do you want to search? Please provide in SQL format: omim_name = 'headache';
[]
------------------
```

Figure 12: Example of successful entry deletion

To confirm the deletion, we then input the same search condition as when we try to confirm all the previous processes. Consequently, the omim entry (3, ‘headache’, ‘AAA’) cannot be found in the database anymore as the entry has been deleted by this process, thus confirming the deletion program.

Record Deletion of whole tables

Furthermore, we may also delete the whole table from the database by choosing the following input statements:

```
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 3
*******************
Delete action...
*******************
Do you want to delete table or entry in table?

                        a. Entry in table
                        b. Whole table
                        q. Quit

Enter your choice: b
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
What table do you want to delete: omim
Deleting table omim
Table omim was deleted
------------------
```

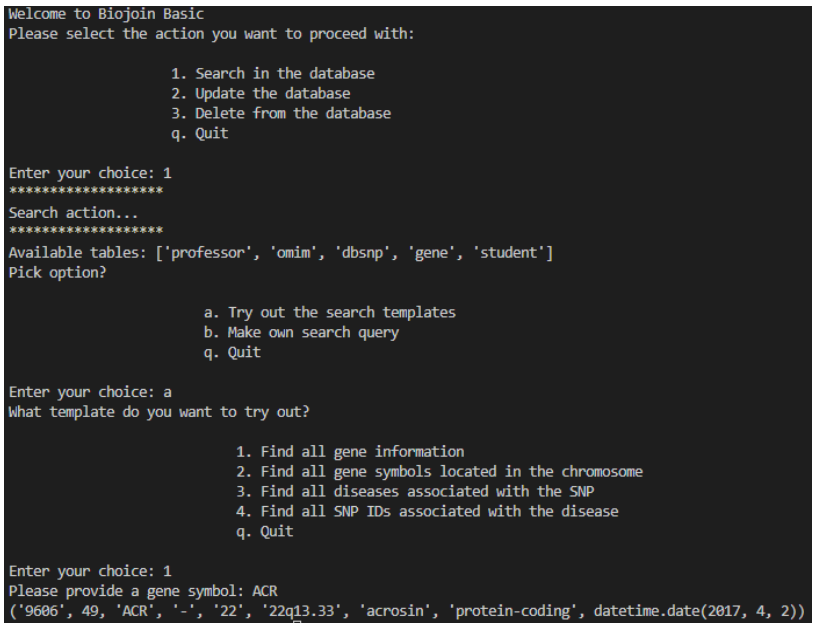
```
u20180813=> \dt
            List of relations
 Schema |   Name    | Type  |   Owner
--------+-----------+-------+-----------
 public | dbsnp     | table | u20180813
 public | gene      | table | u20180813
 public | professor | table | u20180813
 public | student   | table | u20180813
(4 rows)
```

Figure 13: Example of successful table deletion

Practise examples of usage of program (Record pre-defined search on the DB tables)

A part of the phase task to record search on the DB tables. Here are some pre-made templates to find important association in the data.
In this part, the user may be able to select pre-made templates to search for some important association between data in the table. The user may need to input a proper SQL statement for each query according to the template in order to search for specific entries.
This is a list of 4 template seach requests:

1. Given a gene symbol, find all gene information stored in the gene table
   `SELECT * FROM gene WHERE gene_symb = {gene_symbol};`



```
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 1
********************
Search action...
********************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: a
What template do you want to try out?

                            1. Find all gene information
                            2. Find all gene symbols located in the chromosome
                            3. Find all diseases associated with the SNP
                            4. Find all SNP IDs associated with the disease
                            q. Quit

Enter your choice: 1
Please provide a gene symbol: ACR
('9606', 49, 'ACR', '-', '22', '22q13.33', 'acrosin', 'protein-coding', datetime.date(2017, 4, 2))
```

Figure 14: Example result of template 1

2. Given a chromosome id, find all gene symbols located in the chromosome
   `SELECT gene_symb FROM gene WHERE gene_chr = {chromosome};`

```
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 1
*******************
Search action...
*******************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: a
What template do you want to try out?

                            1. Find all gene information
                            2. Find all gene symbols located in the chromosome
                            3. Find all diseases associated with the SNP
                            4. Find all SNP IDs associated with the disease
                            q. Quit

Enter your choice: 2
Please provide a chromosome number: 1
['ABCA4', 'ABL2', 'ACADM', 'ACTA1', 'ACTG1P6', 'ACTG1P7', 'ACTG1P8', 'ACTN2', 'ADAR
'AMY1A', 'AMY1B', 'AMY1C', 'AMY2A', 'AMY2B', 'AMYP1', 'APCS', 'APOA2', 'FASLG', 'AR
1', 'ATP6V0B', 'AVPR1B', 'ADGRB2', 'BCL9', 'BCYRN1P2', 'BGLAP', 'BMP8B', 'BRDT', 'C
```

Figure 15: Example result of template 2

3. Given an SNP ID, find all diseases associated with the SNP

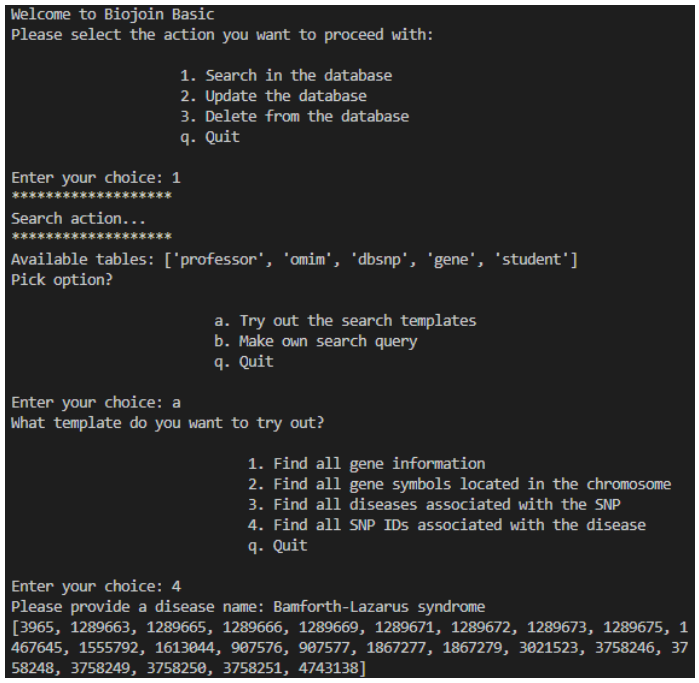
```
SELECT omim_name FROM omim WHERE omim.gene_symb
IN (SELECT gene_symb FROM dbsnp WHERE snp_id = {snp_id});
```

```
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 1
*******************
Search action...
*******************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: a
What template do you want to try out?

                            1. Find all gene information
                            2. Find all gene symbols located in the chromosome
                            3. Find all diseases associated with the SNP
                            4. Find all SNP IDs associated with the disease
                            q. Quit

Enter your choice: 3
Please provide a SNP id: 1467645
['Bamforth-Lazarus syndrome', '$Thyroid cancer, nonmedullary, 4$']
```

Figure 16: Example result of template 3

4. Given a disease name, find all SNP IDs associated with the disease

```
SELECT snp_id FROM dbsnp WHERE dbsnp.gene_symb
IN SELECT gene_symb FROM omim WHERE omim_name = {disease_name});
```



```
Welcome to Biojoin Basic
Please select the action you want to proceed with:

                    1. Search in the database
                    2. Update the database
                    3. Delete from the database
                    q. Quit

Enter your choice: 1
********************
Search action...
********************
Available tables: ['professor', 'omim', 'dbsnp', 'gene', 'student']
Pick option?

                        a. Try out the search templates
                        b. Make own search query
                        q. Quit

Enter your choice: a
What template do you want to try out?

                            1. Find all gene information
                            2. Find all gene symbols located in the chromosome
                            3. Find all diseases associated with the SNP
                            4. Find all SNP IDs associated with the disease
                            q. Quit

Enter your choice: 4
Please provide a disease name: Bamforth-Lazarus syndrome
[3965, 1289663, 1289665, 1289666, 1289669, 1289671, 1289672, 1289673, 1289675, 1
467645, 1555792, 1613044, 907576, 907577, 1867277, 1867279, 3021523, 3758246, 37
58248, 3758249, 3758250, 3758251, 4743138]
```

Figure 17: Example result of template 4

Further information and source code

For more technical information and the access to the source code of the application, please visit our GitHub repository: https://github.com/Tsatsch/Biojoin

Conclusion

As we moved to the second phase of the BioJoin project, BioJoin Creative, we efficiently implemented the database following the design in Phase 1, BioJoin Basic. This is because phase 1 provided a sort of map we can follow in order to reach our final goal of code implementation. However, Phase 1 was not enough, as we observed some inevitable changes because of incomplete information. Furthermore, we implemented a command-line interface software to operate with the database. The software gives the user the possibility to search thru the tables in the database and modify its content. Therefore, what would be expected in Phase 3 would be an upgraded system, a continuation instead, to Phase 2 system.

References

[1] National Library of Medicine, National Center for Biotechnology Information. ***dbSNP.*** https://www.ncbi.nlm.nih.gov/snp/. Retrieved on 13rd April 2022.

[2] National Library of Medicine, National Center for Biotechnology Information. ***Gene.*** https://www.ncbi.nlm.nih.gov/gene. Retrieved on 13rd April 2022.

[3] Johns Hopkins University. ***OMIM.*** https://omim.org/. Retrieved on 13rd April 2022.

[4] ***Entity–relationship model***. Entity–relationship model - Wikipedia. Retrieved on 13rd April 2022.

[5] Supreya Saxena. ***Relation Schema in DBMS.*** Relation Schema in DBMS - GeeksforGeeks. Retrieved on 13rd April 2022.

[6] Alvaro Monge. ***Basic SQL statements: DDL and DML.*** Database Design - DDL & DML (csulb.edu). Retrieved on 13rd April 2022.