BIOL4292: Assignment 2 – Course Work 3

Introduction

This report describes the creation of a bioinformatics database containing multiomics data measurements and their related annotations. The multiomics data includes transcript, protein and metabolic data measurements from a cohort of 106 subjects from a study on aging markers (Ahadi et al., 2020). The database was built using Python 3.10.9 and implemented using the sqlite3 package.

Conceptual Database Design

The given datasets fell into roughly three classes of information. Firstly, the subjects file consisted of basic information about the individuals who made up the cohort such as age, sex and BMI. There were also three files containing numerical measurements, each from a different omics type. Finally, the metabolomics peaks had annotation data associated with them.

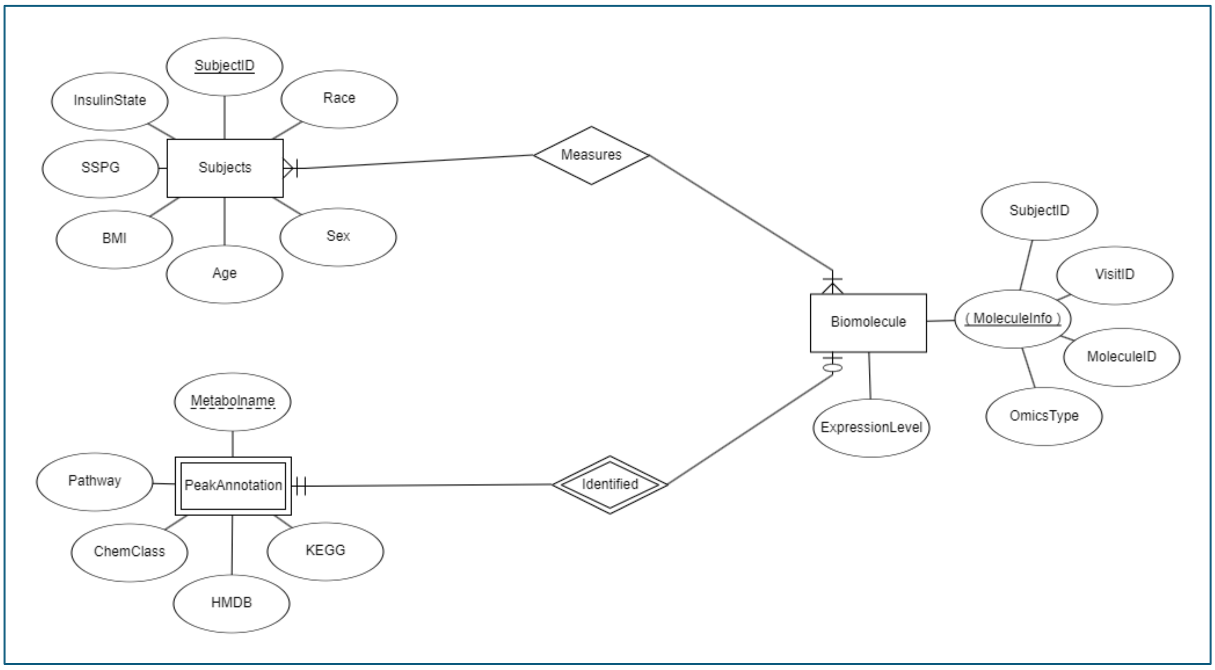

Fig 1: Entity-Relationship Diagram of Multiomics Database. Made using https://erdplus.com/standalone

Based on this information, three entities were planned:

- A strong entity called "Subjects" which was uniquely defined by a simple attribute known as "SubjectID". Its other attributes were simple and consisted of the basic information headers given in the subjects file i.e Race, Sex, Age, Body Mass Index (BMI), Steady-State Plasma Glucose (SSPG) level, and Insulin Status. Even though most of these columns are not required for the predefined queries given, all information has been stored in the interests of code reusability.
- A strong entity called "Biomolecule" which was uniquely defined by a "Molecule Info" attribute. This attribute was a compound attribute consisting of four parts- SubjectID, VisitID, Biomolecule and Omicstype. This entity contained expression level data for all biomolecular measurements generated by subjects during various visits. The reason for a four-part compound attribute was that even after fixing a particular subject, visit and biomolecule it was found that some molecules

such as A1BG were measured at different omic levels and thus, the omics type also needed to be specified to uniquely identify a measurement. “Biomolecule” shares a many-to-many relationship with the “Subjects” entity, known as “Measures”. There is total participation of members of both entities since every biomolecule is derived from a subject and every subject has contributed to the measurement of at least one biomolecule.

- A weak entity called “PeakAnnotation” which is identified by a partially unique attribute known as “MetabolName”. Its existence is dependent on a subset of “Biomolecules” with an “Omicstype” of “Metabolite”. This entity contains information about the metabolites such as their KEGG and HMDB IDs, their chemical classes and biological pathways they are involved in. Thus, there is a one-to-one identifying relationship between “Biomolecule” and “PeakAnnotation” with partial participation from “Biomolecule”. This relationship is known as “Identified”

Mapping Model to Schema

Each entity was converted to a single table. Since none of the relationships had attributes of their own, there was no need for any other tables.

- The “Subjects” entity was directly converted into a table with each attribute becoming a single column. The unique “SubjectID” column became the primary key.
- The “Biomolecule” entity was converted into a table with a column for measuring omics data. Each sub attribute of the compound attribute “MoleculeInfo” became a separate column, and all four columns together formed the primary key. The total participation of all subjects and biomolecules in the “measures” relationship has been implemented via a foreign key referencing the “SubjectID” column in “Subjects”.
- The weak entity “PeakAnnotation” was converted into a table containing all its attributes as well as a “PeakID” column which corresponded to a subset of Biomolecules. Due to uncertainties in the identification of the peaks, the combination of the partially unique “Metabolname” column and the “PeakID” column could be used to form a two-part primary key. The existence dependency of “PeakAnnotation” on “Biomolecule” is implemented via a foreign key constraint from the “PeakID” attribute of “PeakAnnotation” which refers to the “MoleculeID” sub-attribute in the “Biomolecule” entity.

The subjects data fields were mostly numerical with two decimal places or single characters. Data was read line by line, all unknowns and NA were replaced with NONE in each line and then loaded into the database.

For loading the biomolecule table, each expression level file was assigned an omics type and all three files were loaded one by one using a user defined function.

The peakannotation file first required the input data to be searched for uncertainties in peak identification or annotation. These were joined or split as required using regexes. The data was then loaded into the table. Most columns used the varchar datatype.

In all cases files were iterated through only once and loaded line-by-line using the execute command instead of execute many to avoid storing the whole file in memory. The long, multi-column primary keys reduced the need for joins during querying.

Program Design

The program consisted of three files.

An sql file(schema.sql) containing the database schema. This was executed by the main program file using the sql executescript command.

A user-defined dataprocessing module (OmicsDataProcessor.py). It had classes for two main repetitive parts of the program. Firstly, loading the three omics measurement files, each of which required the same procedure. Secondly, using the __str__ special method, a method was created to print the result of any sql select statement which was used to execute each of the nine predefined queries.

A main program(main.py) which carried out the following tasks:

- Implemented input via argparser
- Connected to the database using the sqlite3 package
- Created the database using the schema file
- Loaded the subjects and peak annotations files directly while calling the dataprocessing module to load the three files containing omics data using sql insert statements.
- Fed the correct query to the display function based on the input query number
- Implemented error-handling for simple errors such as missing files and database or wrongly entered query number etc
- Query 9 was handled using the new seaborn.objects interface

Along with these three files, it is required that *Subject.csv, HMP_transcriptome_abundance.tsv*, *HMP_proteome_abundance.tsv*, *HMP_metabolome_abundance.tsv* and *HMP_metabolome_annotation.csv* are placed in the same directory as the three program files before the program is run.

In this manner, the program can create, load a multiomics database and query it for the required information.

References

1. Ahadi, S., Zhou, W., Schüssler-Fiorenza Rose, S. M., Sailani, M. R., Contrepois, K., Avina, M., Ashland, M., Brunet, A., & Snyder, M. (2020). Personal aging markers and ageotypes revealed by deep longitudinal profiling. *Nature Medicine*, *26*(1), 83–90. https://doi.org/10.1038/s41591-019-0719-5
2. Hipp, R. D. (2020). *SQLite*. Retrieved from https://www.sqlite.org/index.html
3. Waskom, M. L., (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021, https://doi.org/10.21105/joss.03021