ER Diagram

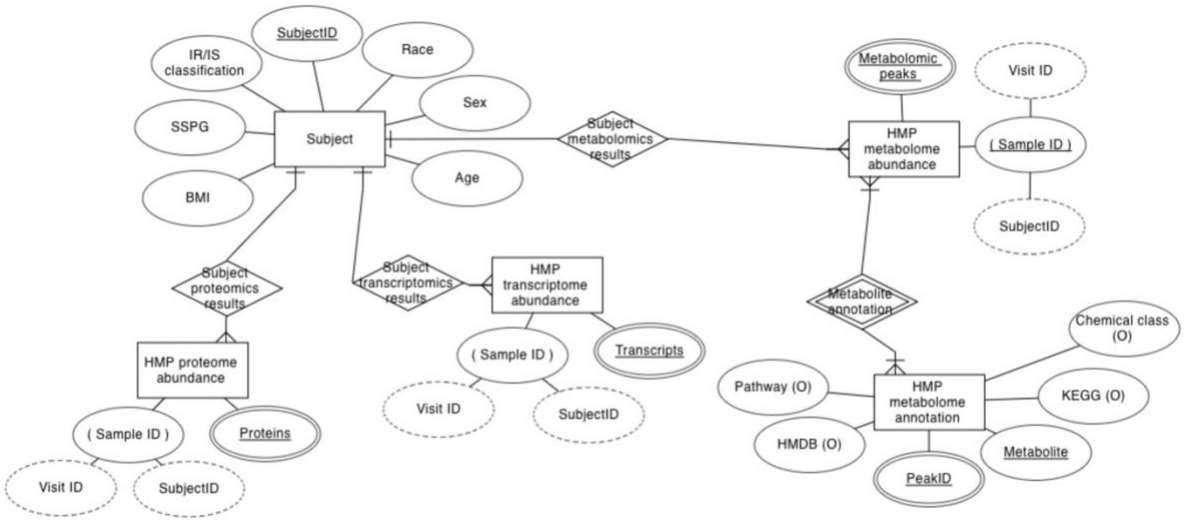

Figure 1. ER diagram of multi-omics data set

The ER diagram represents the five datasets generated from a large multi-omics aging study that collected transcript, protein, and metabolite measurements.

The subject entity has seven attributes pertaining to the information on the Age, Sex, Race, BMI, steady-state plasma glucose (SSPG) level, Insulin sensitivity status (IR/IS classification), and the unique SubjectID of 106 participants. The subject entity has three relationships where it is related, in a one-to-many relationship, to each omics measurements datasets.

Each Subject is linked to many SampleIDs of the omics dataset, where the SampleID is a composite attribute that contains the SubjectID and a VisitID (SubjectID - VisitID). Each omics measurement entity has a unique multivalued attribute that is a model assumption of the many distinct transcript/protein/metabolites' measurement. For example, the transcriptomics dataset has 875 transcript measurement columns which would be impractical to model.

The final entity is HMP metabolome annotation contains information on each metabolite analysed, some of the metabolites have multiple peaks generated from the Mass-spectrometer. Some of the metabolites have other IDs from other databases (KEGG and HMDB) and two description attributes of the chemical class and the biological pathway of the metabolite. This entity is related to the metabolomics dataset where each metabolite can be related to many measurement peaks.

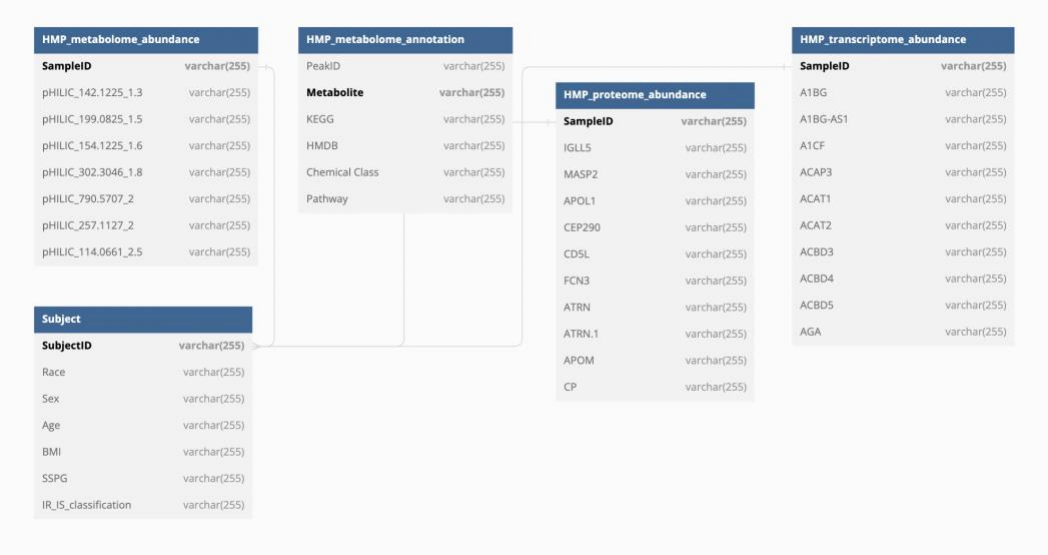

Figure 2. Database schema of multi-omics data set. In the abundance tables, only a small selection of the transcripts, proteins, and metabolites are shown.

High-level programme description

This is an Object-oriented python script that parses through csv and tsv data files to generate SQLite database tables and loads data into those created tables. The main class db_load_and_fetch contains four class functions and one static method. The first function handles the files that are inputted via the *argparse* command line interface option *database_files*, this stores the string names of the files into a list. *Input_handler* checks if the string file name ends with either '.csv' or '.tsv', if not then an error message is returned. Filenames with the correct extensions are appended the class variable self.file_list.

The createdb function essentially takes the first row in the data file and uses it to create an SQL table header. The using a context manager to generate a custom SQL create statement for every file inputted and setting the first column as the primary key. All columns have been set to Varchar (255). This creates a table named using the filename in 'Assignment.db'.

The loaddb function works similarly to created, where it uses a context manager to generate a SQL insert statement for every row in each datafile. There is a separate static method that parses through the metabolome annotations datafile. The function doesn't modify the format of the data when inserted into the database Due to peak identification uncertainties in mass-spectroscopy, several metabolites can be associated one peak, or a peak associated with multiple metabolites. The method parses and modifies the data to ensure that each metabolite has a unique row and the PeakIDs were aggregated into a multivalued column.

The querydb function takes a numeric value between 1 and 9 in the command line in this format: --querydb=1. This numeric value is the key in a python dictionary that is associated with a specific SQL Select statement. These nine statements retrieve specific information from the five tables created in the database. Numeric values outside the specific range returns an error message.