Using a Relational Database to Track Blood Sample Attributes and Storage Locations

Carter Reid

PROBLEM STATEMENT

Biological science technology is advancing rapidly. So rapidly, that researchers have a hard time keeping up. They store and preserve blood, tissue, and electronic data gathered on patients for future patients and discoveries. They preserve the samples for future researchers with future technologies. They preserve the data gathered for future researchers to mine through the vast wealth of genetic information, searching for correlations and genetic markers indicating potential targets for future treatments. Samples are physically stored in large industrial sized freezers should the need for further investigation occur. Currently, many genes have been identified and specific mutations have been identified for many of these genes, even if their functionality, and the effect of the mutations/polymorphisms remains unknown.

With so much data and samples stored, across multiple biorepositories, samples can get lost, and data can remain undisturbed due to organizational deficiencies. I propose to create a relational database model to solve this issue. The proposed model should be capable of tracking both patient genetic data, disease genetic data, sample location data, and additional identifiers such as draw time, patient DOB etc.

METHODS AND TOOLS

The database will be constructed and visualized through MySQL workbench. The proposed database will be hosted using Google Could SQL. Initial queries and database optimization will be performed using SQAlchemy in Jupyter notebooks. Due to HIPAA restrictions, actual patient data may be difficult to access, and impossible to publish without deidentifying multiple patient attributes. It is for this reason that I will most likely need to generate test data to populate the relational database. I will also include built in statistical analysis tools for the gene bank researcher, such as variance and correlation coefficients which will be implemented using pandas and numpy. If a visualization tool is deemed applicable during development, pyplot and seaborn will be used. Finally, if time permits, a user interface that would allow the researcher custom search queries will be implemented. This is slated last in development as the primary objective is to create the relational database, and the user interface merely helps facilitate the use of said database.

THE PROPOSED RELATIONAL DATABASE

Table Patient, contains: patient name, patient_ID (primary key)

Table Sample, contains: sample type, sample_ID (primary key), collection date, storage location

Table Disease, contains: disease_ID (primary key), disease markers.

A “has_a” table relating patients with samples.

A “has_a” table relating patients with diseases.

CHALLENGES AND OBSTICALS

Generating placeholder data has its advantages and disadvantages. There is the possibility of creator bias which could lead to a relational database that works on a hypothetical model but fails in a real-world application. To prepare and prevent this, I will be using some real world data gathered form the CDC and NIH, at least to the extent of tracking real attributes and disease names. Patients will be entirely made up as will their disease status.

Google Cloud SQL uses a credits system, which I currently only have $50 and would not like to purchase additional credits unless necessary.

LEARNING GOALS

The main objective besides creating a functional relational database is to learn and practice hosting and maintaining a relational database. If I create a database that appears stable, and is capable of cascading additions, insertions, and deletions, that would constitute a success. A better success would be if I created a stable database, that also could potentially be useful and could potentially be used by a researcher to perform customized queries.

MAJOR CHANGES TO PROPOSED IMPLEMENTATION

SQAlchemy was swapped out in favor of SQLite. This decision was based on ease of use in the initial implementation, and difficulties exporting the relational diagram from MySQL workbench. MySQL workbench was still used to create a relational diagram which the SQLite tables were created from. This decision to use SQLite instead of SQAlechemy created downstream consequences of which I was unable to connect the database to a cloud service due to technical limitations of SQLite. Further work on this project would need to include a change form SQLite to SQAlchemy for cloud integration. For data efficiency and reduction of duplication and redundancy, several tables were altered or added to make the relational database mostly third normal form. I say ‘mostly’ because there is some data in the Patient_mutations table that has redundancy, though that was initially implemented to make a single query more efficient.

Built in variance and correlation queries and calculations were not implemented due to time constraints, and the decision to cut this from the finalized product was based on them not being necessary for the learning goals. Instead, the user interface was implemented prior to these as development was done using SQLite and the user interface made testing easier. The goal being to create a relational database, where a user could track both patient and genetic data, disease data, and sample location data, and the UI supports this goal better than more in depth statistical analysis would. Because of this decision, numpy, pyplot and seaborn were not implemented or used. If development were to continue, I do not see any reason why these tools would be incompatible with the current implementation of the healthcare database.

THE FINALIZED RELATIONAL DATABASE

Patient table has patient disease and DOB added to it as well as previous patient ID and name. The decision to add DOB makes sense because each patient will only have one date of birth, and patient name and DOB is a common identification method in healthcare, which this database is intended to simulate. Samples table has patient ID included in it to make samples easier to relate to patients, sample type was dropped due to simplicity, though in a real setting sample type would be a realistic field to have, though it is not necessary in this case to meet the learning goals. Storage location was expanded to have a freezer ID and a row and column. Freezer ID, row, and column are under a unique constraint to ensure that samples cannot share the same location. Disease table was reworked to only have cancer type and disease ID, which is to be referenced by patients and patient mutations. Patient_mutations table was added instead of incorporating mutations into the disease table. Patient mutations table has mutation ID, which references mutations to connect mutation ID with mutation name, and disease ID which references the Diseases table to determine cancer type. The assumption being that a patient will present only one form of cancer at a time, which is not absolutely true in the real world, but has a fairly low probability of occurring. The patient mutations table has a unique constraint on all tuples, so that redundant data is not entered, though a patient with 5 mutations will take up 5 rows in the table. As more mutations are recorded, revising this to a list datatype may be a good idea.

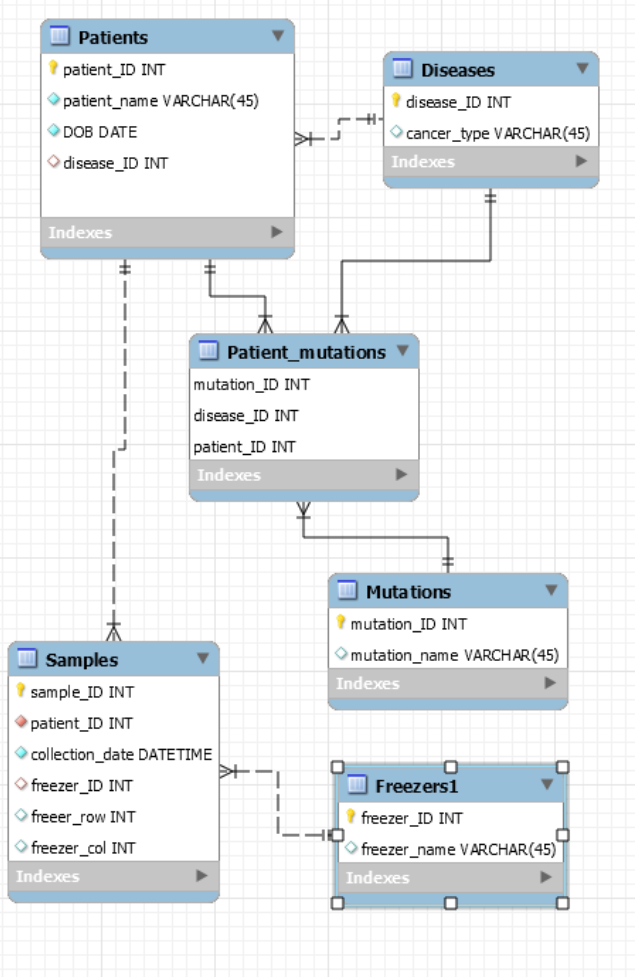

Figure 1: Schema of the finalized relational database

TRIAL DATA

The placeholder data was initially slated to have real gene mutations and diseases pulled from the CDC and NIH. This is still the case to some extent, though I merely picked the top 12 gene mutations associated with cancer cell differentiation. I also picked the top ten cancer's that affect both genders so that I wouldn't have to do any gender testing during the generation of the face data. In other words, I left out ovarian and prostate cancers. Breast cancer can affect both genders, so I left that one in. Patient names were generated from the website: www.listofrandomnames.com which only generates 50 names at a time, so I only have 50 patients in the mock database. The freezer name table consists of actual freezers from my previous workplace, where freezers were given names to make it easier for people to identify them as opposed to using their barcode number.

THE FINAL RESULT

The final product is a python application that allows you to load an existing database or create a new one with random information. The user can add patient data, add samples to a patient entry, edit patient data, change sample location, or remove patient data and all samples associated with the patient. If a user tries to add a mutation, disease, or freezer that is not currently in the database, the application will automatically add it. The database tracks freezer locations and prevents the user from storing multiple samples in the same freezer, row, and column. The user interface supports efficient patient lookup either by patient ID or name. If there are multiple patients with the same name, DOB will be requested to ensure the correct patient is selected. It is possible to have multiple patients with the same name and same date of birth, which is unlikely, and causes a couple extra steps to select the correct patient but does not cause an actual problem to the application.

```
+------------+--------------+---------------+---------+
|  patient_ID | patient Name  |  Date of Birth | Disease  |
|------------+--------------+---------------+---------|
|          49 | Lizbeth Buker |       10031924 | Melanoma |
+------------+--------------+---------------+---------+
Genetic Markers:  ['BRCA2', 'BRAF', 'RAS', 'p53', 'ERK', 'TRK']
Samples belinging to this patient:
+------------+------------+-----------------+-------+----------+-----------------+
|  sample_ID |  Patient_ID |  collection date |  row |  column | Freezer name    |
|------------+------------+-----------------+-------+----------+-----------------|
|        245 |          49 |         7021928 |    24 |        5 | Steve the Pirate |
|        246 |          49 |         7181984 |    24 |        6 | Steve the Pirate |
|        247 |          49 |         7051908 |    24 |        7 | Albus DoubleDoor |
|        248 |          49 |         4261905 |    24 |        8 | Steve the Pirate |
|        249 |          49 |         7231964 |    24 |        9 | Mr Freeze        |
+------------+------------+-----------------+-------+----------+-----------------+
Total Samples:  5
```

Figure 2: Sample patient report with genetic markers and sample locations

The idea being that a oncology researcher can look up a patient to see their disease and genetic markers and then efficiently find all patient samples if they wish to identify other novel genetic markers.

The application allows the user to select all samples belonging to a disease as well. The idea being that if a researcher reads about a newly identified gene, and want to see how many patients with melanoma in their database may also have this marker, they can select all melanoma samples and sequence and test

them to see how correlated the new marker is with the disease.

```
Find Samples by Disease
Please enter the name of the disease
Melanoma
The following samples are from patients with the selected disease:
+-------+----------+-----------------+-----------+-------+--------+-----------------+
|  p_ID | Disease  |  collection date |  Sample ID |  row |  column | Freezer name    |
|-------+----------+-----------------+-----------+-------+--------+-----------------|
|     4 | Melanoma |         9141925 |        20 |     2 |      0 | Steve the Pirate |
|     4 | Melanoma |         9021956 |        21 |     2 |      1 | Mr Freeze       |
|     4 | Melanoma |         4142013 |        22 |     2 |      2 | Mr Freeze       |
|     4 | Melanoma |         9221934 |        23 |     2 |      3 | Steve the Pirate |
|     4 | Melanoma |         6181902 |        24 |     2 |      4 | Albus DoubleDoor |
|    10 | Melanoma |         6051926 |        50 |     5 |      0 | Albus DoubleDoor |
|    10 | Melanoma |        11231989 |        51 |     5 |      1 | Steve the Pirate |
|    10 | Melanoma |         6101912 |        52 |     5 |      2 | Albus DoubleDoor |
|    10 | Melanoma |         4181979 |        53 |     5 |      3 | Albus DoubleDoor |
|    10 | Melanoma |        10241970 |        54 |     5 |      4 | Steve the Pirate |
|    14 | Melanoma |         6241990 |        70 |     7 |      0 | Steve the Pirate |
|    14 | Melanoma |         1221979 |        71 |     7 |      1 | Mr Freeze       |
|    14 | Melanoma |         9131999 |        72 |     7 |      2 | Steve the Pirate |
|    14 | Melanoma |        10251981 |        73 |     7 |      3 | Albus DoubleDoor |
|    14 | Melanoma |         6112004 |        74 |     7 |      4 | Mr Freeze       |
|    28 | Melanoma |         8131995 |       140 |    14 |      0 | Mr Freeze       |
|    28 | Melanoma |         2121920 |       141 |    14 |      1 | Mr Freeze       |
|    28 | Melanoma |         9021924 |       142 |    14 |      2 | Albus DoubleDoor |
|    28 | Melanoma |         1141951 |       143 |    14 |      3 | Steve the Pirate |
|    28 | Melanoma |        10171940 |       144 |    14 |      4 | Albus DoubleDoor |
|    32 | Melanoma |         1071934 |       160 |    16 |      0 | Mr Freeze       |
|    32 | Melanoma |         3021926 |       161 |    16 |      1 | Albus DoubleDoor |
|    32 | Melanoma |        11121966 |       162 |    16 |      2 | Mr Freeze       |
|    32 | Melanoma |         7131990 |       163 |    16 |      3 | Mr Freeze       |
|    32 | Melanoma |         1141934 |       164 |    16 |      4 | Steve the Pirate |
|    35 | Melanoma |         7112006 |       175 |    17 |      5 | Mr Freeze       |
|    35 | Melanoma |         8201990 |       176 |    17 |      6 | Steve the Pirate |
|    35 | Melanoma |        10031959 |       177 |    17 |      7 | Mr Freeze       |
|    35 | Melanoma |         4031905 |       178 |    17 |      8 | Albus DoubleDoor |
|    35 | Melanoma |         7212002 |       179 |    17 |      9 | Steve the Pirate |
|    48 | Melanoma |         7072001 |       240 |    24 |      0 | Albus DoubleDoor |
|    48 | Melanoma |         2111900 |       241 |    24 |      1 | Albus DoubleDoor |
|    48 | Melanoma |         1232013 |       242 |    24 |      2 | Mr Freeze       |
|    48 | Melanoma |        11261973 |       243 |    24 |      3 | Albus DoubleDoor |
|    48 | Melanoma |        11091968 |       244 |    24 |      4 | Mr Freeze       |
|    49 | Melanoma |         7021928 |       245 |    24 |      5 | Steve the Pirate |
|    49 | Melanoma |         7181984 |       246 |    24 |      6 | Steve the Pirate |
|    49 | Melanoma |         7051908 |       247 |    24 |      7 | Albus DoubleDoor |
|    49 | Melanoma |         4261905 |       248 |    24 |      8 | Steve the Pirate |
|    49 | Melanoma |         7231964 |       249 |    24 |      9 | Mr Freeze       |
+-------+----------+-----------------+-----------+-------+--------+-----------------+
```

Figure 3: Sample report for all patients with melanoma.

```
Matches found:  2 containing:  ['BRCA1', 'BRCA2', 'BRAF', 'MYC']
+--------------+--------------------+-----------------+-------------------+
|  patient_ID | patient Name       |  Date of Birth | Disease           |
|--------------+--------------------+-----------------+-------------------|
|            9 | Clemencia Bussell |         3161949 | Colorectal Cancer |
+--------------+--------------------+-----------------+-------------------+
Genetic Markers:  ['BRCA1', 'BRCA2', 'BRAF', 'BCL2', 'SWI/SNF', 'WNT', 'MYC', 'ERK']
Samples belinging to this patient:
+-------------+-------------+-----------------+-------+---------+-----------------+
|  sample_ID |  Patient_ID |  collection date |  row |  column | Freezer name    |
|-------------+-------------+-----------------+-------+---------+-----------------|
|          45 |           9 |         1201925 |     4 |       5 | Albus DoubleDoor |
|          46 |           9 |         3221902 |     4 |       6 | Steve the Pirate |
|          47 |           9 |         2021922 |     4 |       7 | Steve the Pirate |
|          48 |           9 |        10202016 |     4 |       8 | Steve the Pirate |
|          49 |           9 |        10191946 |     4 |       9 | Mr Freeze       |
+-------------+-------------+-----------------+-------+---------+-----------------+
Total Samples:  5

+--------------+--------------------+-----------------+-----------+
|  patient_ID | patient Name       |  Date of Birth | Disease   |
|--------------+--------------------+-----------------+-----------|
|            4 | Ardelia Highsmith |         8141943 | Melanoma  |
+--------------+--------------------+-----------------+-----------+
Genetic Markers:  ['BRCA1', 'BRCA2', 'BRAF', 'pRb', 'p53', 'SWI/SNF', 'MYC']
Samples belinging to this patient:
+-------------+-------------+-----------------+-------+---------+-----------------+
|  sample_ID |  Patient_ID |  collection date |  row |  column | Freezer name    |
|-------------+-------------+-----------------+-------+---------+-----------------|
|          20 |           4 |         9141925 |     2 |       0 | Steve the Pirate |
|          21 |           4 |         9021956 |     2 |       1 | Mr Freeze       |
|          22 |           4 |         4142013 |     2 |       2 | Mr Freeze       |
|          23 |           4 |         9221934 |     2 |       3 | Steve the Pirate |
|          24 |           4 |         6181902 |     2 |       4 | Albus DoubleDoor |
+-------------+-------------+-----------------+-------+---------+-----------------+
Total Samples:  5
```

Figure 4: Sample report for find patients and samples from genetic markers.

An additional query that researchers may wish to do is to find a set of patients and samples with a certain combination of genetic markers, such as: BRCA1, BRCA2, BRAF, MYC. See figure 4. Initially I intended this interaction to also produce other statistical values, which were not implemented due to time constraints. Instead, the query lists all patients that have the genetic markers listed. One potential problem is if the researcher only searches for a common genetic marker, then the output is sometimes large, as it prints a full patient report for all patients with the markers selected for. A solution to this is to have more specific searches.

NEXT STEPS

Next steps for this application include adding a query to select from genetic attributes and a disease, as I believe researchers would be interested in selecting melanoma patients with BRAF mutation, rather than first selecting patients with a specific disease and then selecting patients with a certain genetic profile. I believe a more integrated query would make more sense in the long run.

SQLite is a great a great in-process library for self-contained, serverless transactional SQL database engine. Online hosting and support for simultaneous users would be extremely beneficial for the validity of this hospital application. Most web hosting services do not support SQLite. It is supposedly possible to connect to a dedicated server via ssh and access a SQLite database, though that was not exactly practical during this implementation. A switch from SQLite to some other transactional SQL database engine such as MySQL or SQAlchemy would be required for online hosting.

Adding visualization would be interesting as well, though perhaps not with the randomly generated data currently populating this application. Correlation coefficients and histograms based on patient age are just two of many potential additions to the data mining aspect of this application. Adding in a way to select the most common genetic mutations from diseases would be useful as well.

Updating the user interface to a graphical user interface would be a major improvement. Some of the menus were clunky and awkward due to needing to enter a number. A graphical user interface GUI would

allow for much more efficient use, though the creation and implementation of a GUI is currently outside the present scope of the project.

LEARNING GOALS ACHIEVED

The primary learning goal for this assignment was to create a stable relational database that supported multiple tables related through various foreign keys. This has been achieved, in that there are six tables with six references between them. The application supports adding, updating, and removing data. Certain tables are protected from data removal, though this does not interfere with the learning goals. For example, if a disease is entered in the diseases table, then all patients with that disease are removed, the disease still exists in the disease table. All tables with key of patient ID cascade on deletion. The data base is stable – any sequence of adding data or removing or modifying data does not invalidate or disrupt the overall schema.

The secondary learning goal was to create a database system that could be useful to a researcher. I believe, as someone who has worked in a blood bank that yes, this could be of limited use to a blood bank administrator. The current application does not provide strong data mining and statistical tools to help researchers discover novel relationships between genetic markers and disease, but it does provide a method for tracking and accessing existing information.