Project Report

Just Another DB Group
github.com/1strayos/cs-470-database-mgmt-systems
Caleb German, Ethan McFarland, Andrew Bellas, Suban Burale

ABSTRACT:

Introduction:

The purpose of this project is to create a database that contains data on COVID-19 cases by zip code, hospital, and population demographics. This database is only a proof of concept and will not contain all demographic data in the United States but will demonstrate the usefulness and functionality of such a database. Some uses could be: retrieving number of infected by zip code and state, retrieving information about hospitals, test/equipment hospital has per zip code, and information such as median income, age of populations/ patients on a state down to a hospital level of detail.

Information Collection & Requirements:

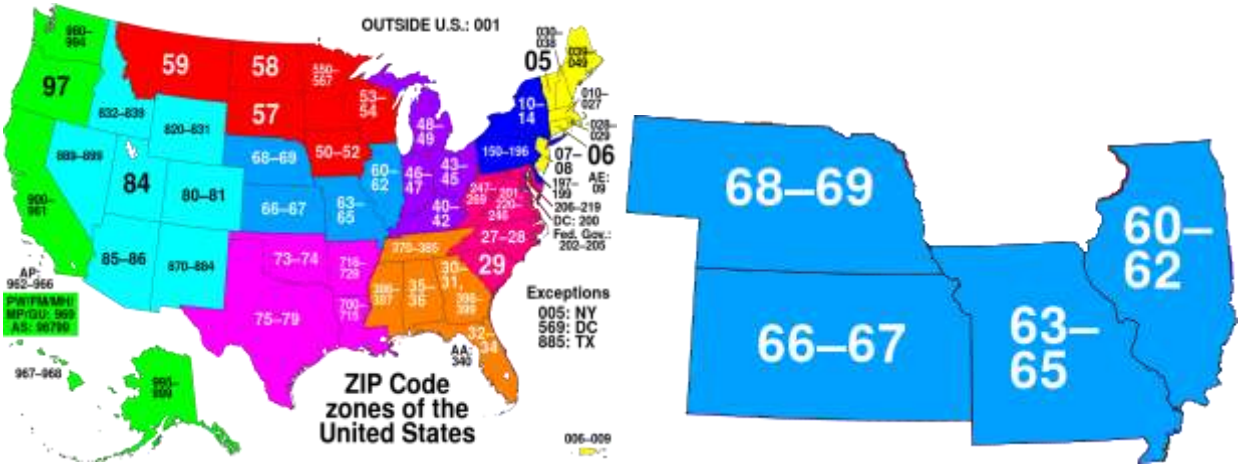
- Each zip code and hospital have a population subset of a total population, with information on median age and median income along with derived attributes such as: number of patients that have the virus, and a number of patients who do not have recovered from the virus.
- Each zip code has a unique code and a state abbreviation corresponding to the state that zip code is located in.
- Each hospital is located in a zip code and has a maximum occupancy, number of staff, name, and derived information such as number of healthy and sick patients.
- Each hospital contains patients that have a name, age, infection status, and are unique by their patent id.

- Each patient takes COVID tests to identify if they are infected or not. Test have a brand id, result, and are unique by test id.

Architecture:

We will be using a two-tier client-server architecture for our database. The client-side will be running an application program that accesses the server-side database. The server will house our RDBMS built using SQL hosted by Amazon RDS. The server will also process transactions and queries that are given to it by the client-side machines. The client side will be running a windows form application that interfaces with our RDBMS.

Scope & Ideas:
This database offers experts and citizens alike information on COVID-19 cases. It provides COVID-19 cases concerning geographic and population constraints. Patient's visits to hospitals will be collected, in which that data will be added to the zip codes. For our project we are only looking at the area codes:



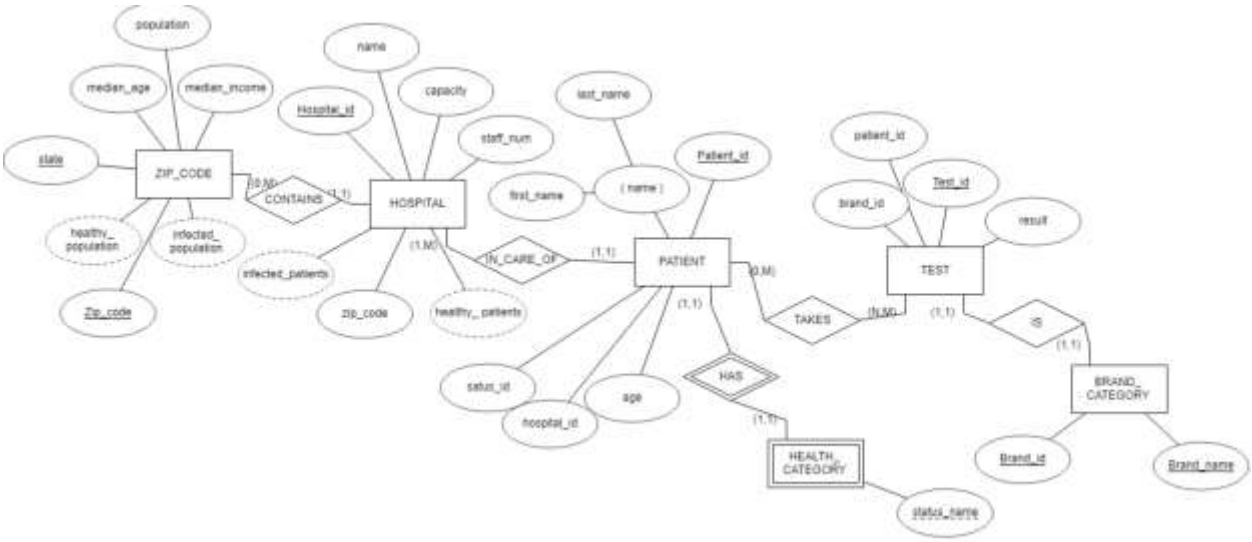
Platform:
For this project, we will be using the Community Edition of MySQL 8.0. Though MySQL comes with built-in command-line interfaces, the Workbench GUI provided by MySQL will serve as our primary interface, as it is far more user-friendly to build our RDBMS. The server running the MySQL instance will be accessible remotely via a windows form application and will have login credentials that will affect the way our RDMS will be accessed.

Roles:
Caleb: database designer, system analysts
Suban: database designer, system analysts
Ethan: Application programmer, DBMS system designer and implementor
Andrew: Application programmer, DBMS system designer and implementor

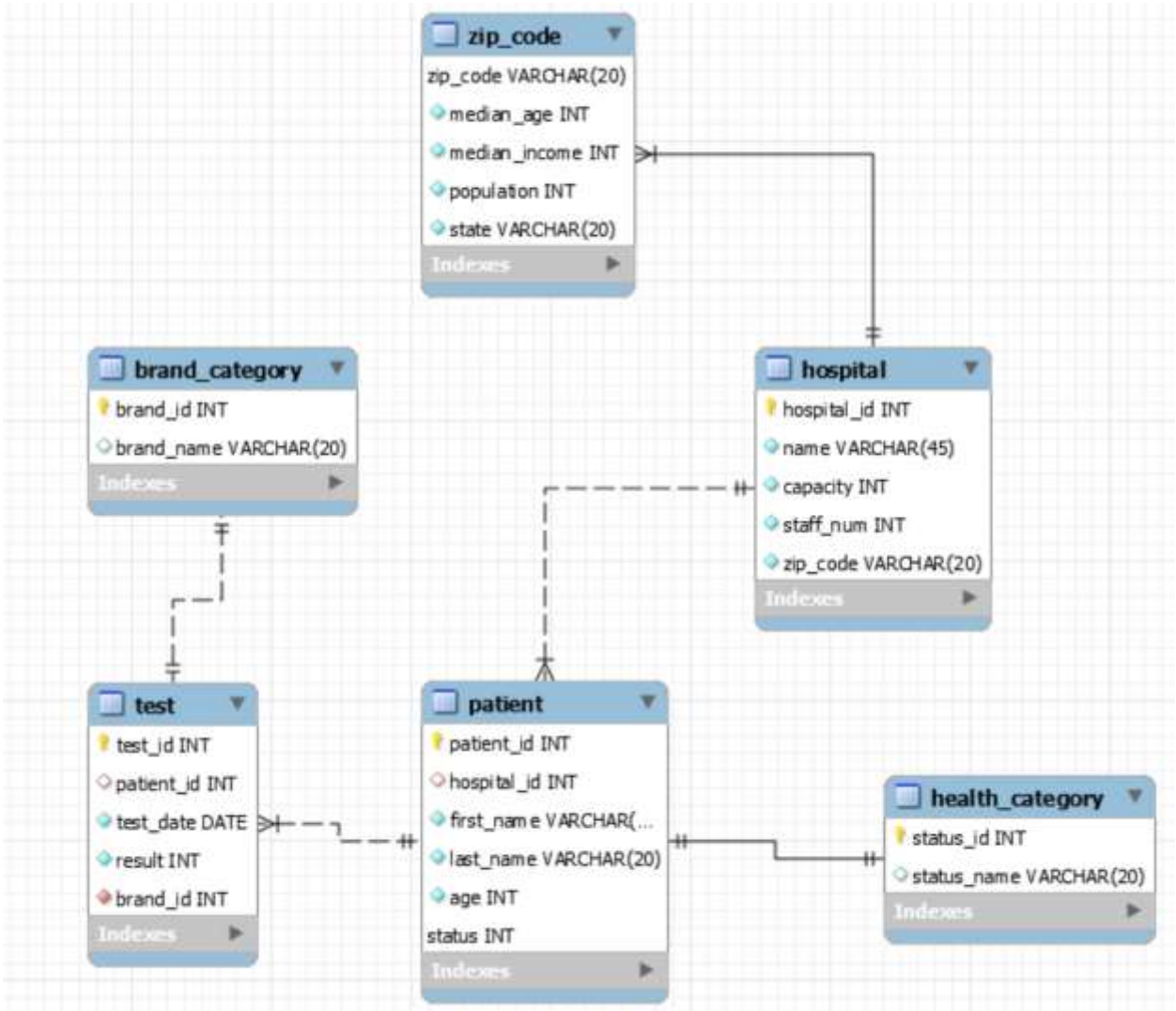
These roles are large and general but will help establish what area people on the team will be focusing on. In reality, each team member may reach across their assigned roles to help fellow team made and to better understand aspects of making and using a DBMS due to the small scale of our project. Most likely the Database Admin role will be split up between all of the team members due to this team's small size.

Entity Relation Diagram, Schema, Constraints, and Core Relational Algebra Queries

ER Diagram:


population
median_age
median_income
name
capacity
Hospital_id
staff_num
last_name
Patient_id
patient_id
Test_id
brand_id
result
state
ZIP_CODE
CONTAINS
HOSPITAL
first_name
name
healthy_ population
infected_ population
infected_patients
IN_CARE_OF
PATIENT
TEST
Zip_code
zip_code
healthy_patients
TAKES
IS
BRAND_ CATEGORY
status_id
age
HAS
hospital_id
HEALTH_ CATEGORY
Brand_id
Brand_name
status_name

Schema:


zip_code
zip_code VARCHAR(20)
median_age INT
median_income INT
population INT
state VARCHAR(20)
Indexes
brand_category
brand_id INT
brand_name VARCHAR(20)
Indexes
hospital
hospital_id INT
name VARCHAR(45)
capacity INT
staff_num INT
zip_code VARCHAR(20)
Indexes
test
test_id INT
patient_id INT
test_date DATE
result INT
brand_id INT
Indexes
patient
patient_id INT
hospital_id INT
first_name VARCHAR(...
last_name VARCHAR(20)
age INT
status INT
Indexes
health_category
status_id INT
status_name VARCHAR(20)
Indexes

Constraints:

- The number of patient's in a hospital cannot exceed the hospitals maximum occupancy
- A zip code must belong to a single state
- A hospital must be in a single zip code
- Test administered cannot be less than one each day
- Every patient must have a status and it must be a valid status
- Each test must have a brand(brand must be valid and patient id
- No two hospitals of the same name in the same zip code

Core Relational Algebra Queries:

- Join across all tables related to medical information:

  $$HOSPITAL \bowtie_{HOSPITAL.hospital\_id=PATIENT.hospital\_id}((HEALTH\_CATAGORY \bowtie_{PATIENT.status=HEALTH\_CATEGORY.status\_ID}((PATIENT \bowtie_{TEST.patient\_id=PATIENT.patient\_id} (BRAND\_CATEGORY \bowtie_{BRAND\_CATEGORY.brand\_id=TEST.brand\_id} (TEST))))$$

  This join gives a table with all medical information joined together. It may not be the most practical for queries optimization due to the lack of selects on the information but is general enough to extrapolate any medical information out in our database. This more a proof of concept than an actual query but can be built upon or torn down in levels of joins for aggregate information.

- The relational algebra to return the # of tests by brand.

  $$brand\_name\ F_{count\ TEST.brand\_id}(PATIENT \bowtie_{TEST.patient\_id=PATIENT.patient\_id} (BRAND\_CATEGORY \bowtie_{BRAND\_CATEGORY.brand\_id=TEST.brand\_id} (TEST)))$$

- The relational algebra for the hospital's names by zip codes(this more of example of a query that could be run given our data base)

  $$\Pi_{zip\_code,name}(Zip\_CODE \bowtie_{ZIP\_CODE.zip\_code= HOSPITAL.zip\_code}(HOSPITAL))$$