CSCI 5408
DATA MANAGEMENT AND WAREHOUSING

Group - 6 Report

Banner ID:
B00963832 – Heli Desai
B00980674 – Ovaiz Ali
B00976563 – Haoyu Wang
B00974443 – Yukta Gurnani
B00981275 – Raisa Putri

Java Code Repository Link: https://git.cs.dal.ca/putri/csci5408-distributed-database

Contents

Figures

Tables

Background Research

The initial phase of our project was dedicated to background research within the healthcare domain. Our focus was on understanding the integration of Distributed Database Systems technologies in healthcare. The team conducted a comprehensive review of various aspects within the healthcare industry, gathering information from credible sources.

Below is a tabular representation of our research findings:

Table 1:Table depicting the background research findings.

Topic	URL	Summary of Findings
Hospital	Halifax Infirmary	Explored facilities and services offered, highlighting the significance of advanced healthcare infrastructure.
Patient	Better Health	Discussed the importance of patient records and paperwork in hospital settings.
Medical History	National Cancer Institute	Reviewed how medical histories are documented and their critical role in patient care.
Prescription	Student Doctor	Analysed prescription writing practices, emphasizing accuracy and patient safety.
Prescription	NCBI	Further insights into the standard prescription processes and their implications.
Hospital Staff	Better Health	Investigated the roles and responsibilities of hospital staff in patient care.
Medical Waste	WHO	Highlighted the management and disposal practices for medical waste to ensure environmental safety.
Patient Meals	Johns Hopkins Medicine	Explored the nutritional care and meal services provided to patients.
Insurance	My Aetna Health Plans	Reviewed insurance card benefits and its role in accessing healthcare services.
HIMSS	HIMSS	Comprehensive insights on healthcare information technology, crucial for distributed database systems.
Health Tech Insider	Health Tech Insider	Focuses on the latest in health and medical technology, relevant for database system innovations.
IoT Equipped Intelligent Distributed Framework for Smart Healthcare Systems	Arxiv	Discusses an IoT-based distributed framework for healthcare, emphasizing secure and fast data availability through DDMS and blockchain.
Distributed Database Strategies in A Healthcare Record Systems	*https://proceeding.unikal.ac.id › article › download*	Deploys Multi-Master Replication model, presenting theoretical research on distributed database, thereby enabling the development of large systems in dispersed environment and wide use scope.
Strategies to Access Patient Clinical Data	*https://www.scitepress.org › Papers*	Makes comparison between applying a common data model and using Semantic Web principles by evaluating them

from Distributed Databases		according to parameters relevant to data integration, such as cost, data quality, interoperability, extendibility, consistency, and efficiency.
What is Distributed Healthcare?	*https://orionhealth.com › global › what-is-distributed-h...*	Focuses on how distributed healthcare can be harnessed to give personalized healthcare
A Secure Healthcare System: From Design to Implementation	*https://www.sciencedirect.com › science › article › pii*	Presents the design and development of a novel EHR system incorporating two formal development methodologies as software engineering perspective, together with database development approach.

Summary of Findings

To get knowledge of healthcare domain, we visited several websites such as HIMSS and Health Tech Insider. Through this, we gained insights about functioning of existing healthcare systems. On gaining domain knowledge, we started working on gathering requirements. We had dived deep into the phase of requirement gathering by reviewing many research papers and blogs. With this, we learned that how different technologies such as IOT, blockchain can be harnessed to ensure secure and fast data availability through DDMS. We even came across by the concept of multi-master replication. This model involves having multiple master nodes that are all able to accept write commands, and these write operations are replicated to all other masters.[1] Multi-master setups are useful for achieving higher availability and better write performance across geographically distributed systems.[1]

Further when referring the “Strategies to Access Patient Clinical Data from Distributed Databases” research paper, we got to know that despite the recognised gain of data sharing, database owners remain reluctant to grant access to the contents of their databases because of privacy and security issues, and because of the lack of a common strategy for data sharing. The paper eventually discusses about the two main approaches to solve this issue, i.e. applying a common data model, or using Semantic Web principles. The paper concludes by making comparison between two approaches by evaluating them according to parameters relevant to data integration, such as cost, data quality, interoperability, extendibility, consistency, and efficiency.

To understand how the product will impact lives of patients in a better way, we started looking for various internet sources and found a blog by Orion health. The blog “What is Distributed Healthcare?” elaborates the benefits of providing decentralised care services. The blog urges for the requirement of common data platform to get dynamic view of patient’s medical record. Also, it would help in crossing the boundary between primary, community and hospital-based care.

On understanding the requirements, we focused on how database development and software engineering go together. For this we studied the “A Secure Healthcare System: From Design to Implementation” research paper. This paper demonstrates the steps of development process based on database ER-model and normalizes the final design to 3NF.

The background research conducted across various topics within the healthcare industry revealed several key insights:

Infrastructure and Services: Many such other hospitals, like the Halifax infirmary, represent the importance of a strong healthcare infrastructure providing all possible facilities required in comprehensive health care.

Patient Care and Safety: Proper management of patient records, prescriptions, Json, and medical history is core to ensuring high standards of patient care and safety.

Management in healthcare facilities is very critical and encompasses the handling of the workforce, in this case, efficient management of hospital staff. It also involves the way hospital operations that deal with waste disposal of medical bodies are managed and the way patients are served with their meals. Insurance: Understanding insurance is an important facet of the process of financial planning of healthcare. The offered benefits make the access to services easy and possible in accordance with the offered health care.This has found, thus, the significance required to integrate advanced database systems within the healthcare sector to manage the operations of healthcare in an efficient manner. Insights collected from all these sources, therefore, are going to be helpful in developing a database model which is fitting to the subtlety of the requirements within the healthcare sector.

Design Issues

This has to be addressed, if ever there is an absolute need, when handling historical data—most especially that the health industry would keep the records of the medical history and the date it corresponds to. The Medical History entity is included for the purpose. All others that require historical data must have a "date" attribute necessary to support this assurance that our Entity-Relationship Diagram (ERD) is complete, taking on historical data smoothly and avoiding any glitches.

Upon reviewing our initial conceptual model, the following design issues have been identified:

1. **Chasm Trap in Doctor Entity**
   The "Treated_By 1:M Patient" relationship under the Doctor entity will give rise to a chasm trap since for doctors, it is possible that they treat more than one patient at the same time, and the patients can also be treated by more than one doctor. This makes it vague and complicates things for queries involving interaction.
2. **Fan Trap in Surgery Entity**
   Surgery shows a potential fan-trap situation. That is, one patient will have undergone a lot of surgeries, and the same for many doctors. This situation brings some sort of complexity which may produce both unnecessary duplication and representation of uncertainty of surgical procedures and intervention staff.

Solutions Proposed
To address these issues and improve the initial conceptual model, the following solutions are proposed:

1. **Chasm Trap Resolution**
   A new entity, relationship, was added to clearly define doctor-patient interactions without ambiguities.
2. **Fan Trap Resolution**
   Create a separate json entity to represent the groups of doctors related to surgeries so that duplicacy of data is avoided, which ultimately results in consistency in output.

Entity-Relationship Diagram (ERD)

Our ERD was meticulously developed to include over 20 entities, reflecting the intricate relationships within the healthcare domain. Each entity was designed with consideration for real-world healthcare processes, ensuring a comprehensive model that supports the diverse needs of healthcare data management.

Initial ERD

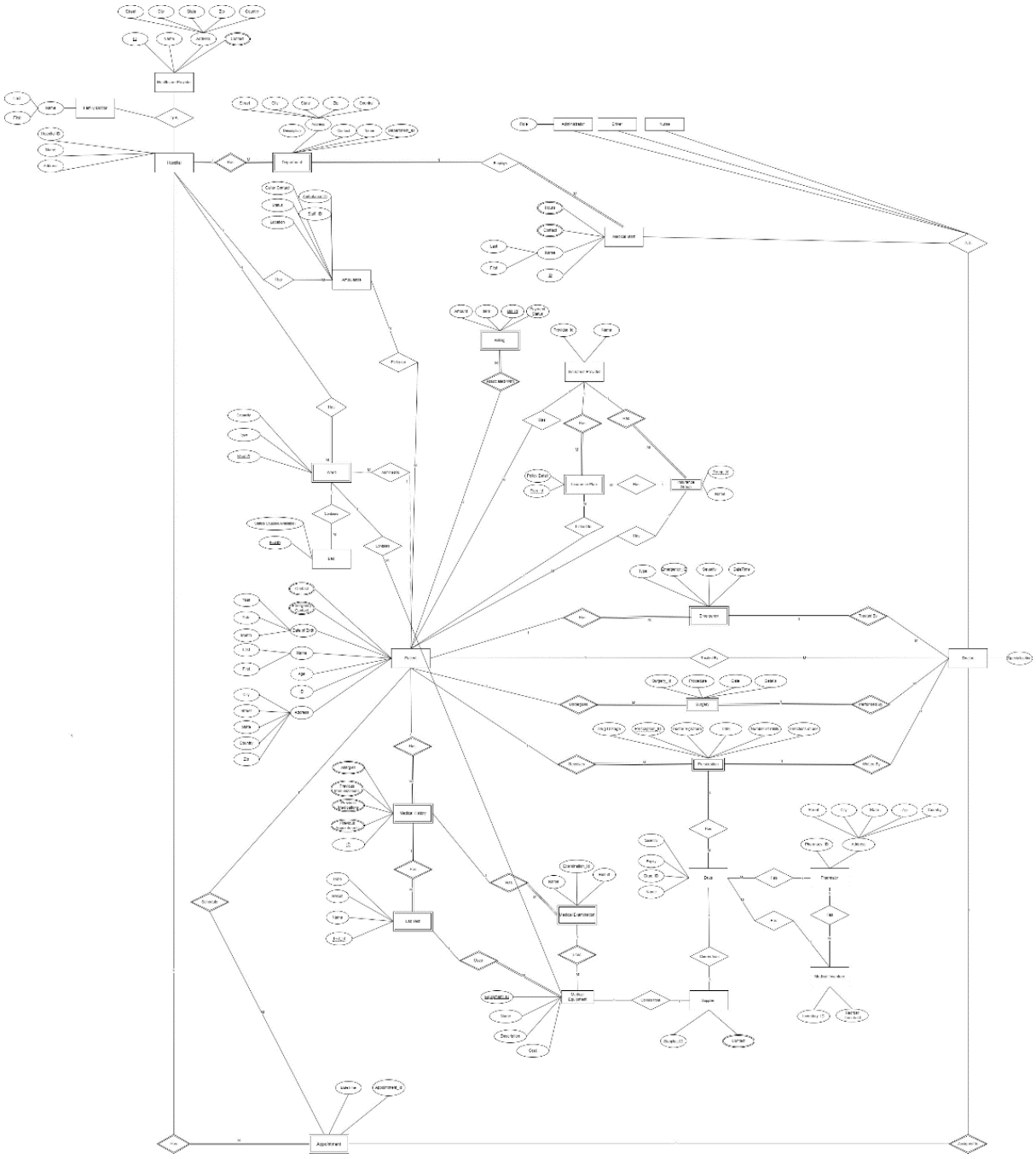

Figure 1: Figure representing the initial ERD

Final ERD After Resolving Design Issue.

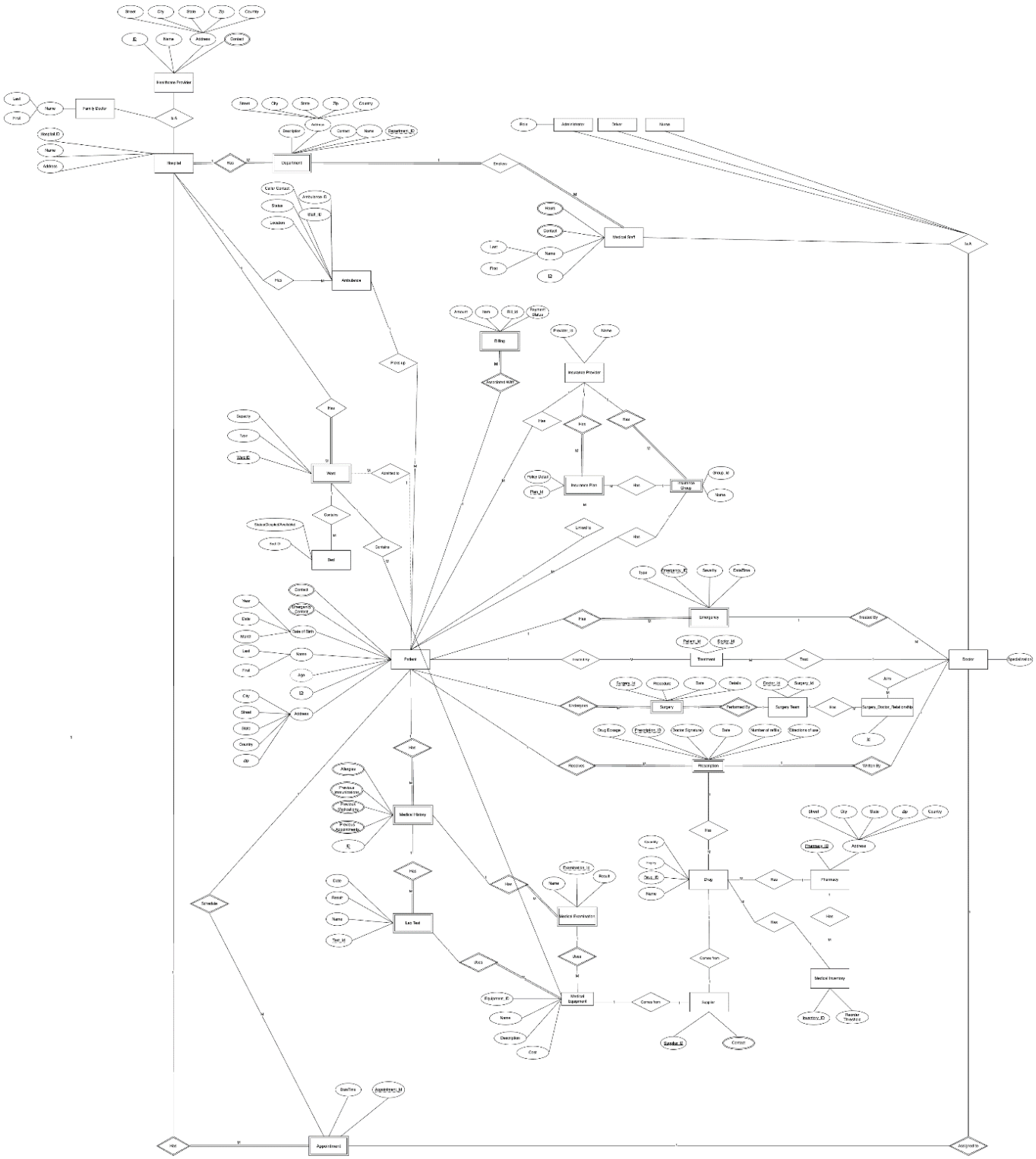

Figure 2: Final ERD

Logical Phase

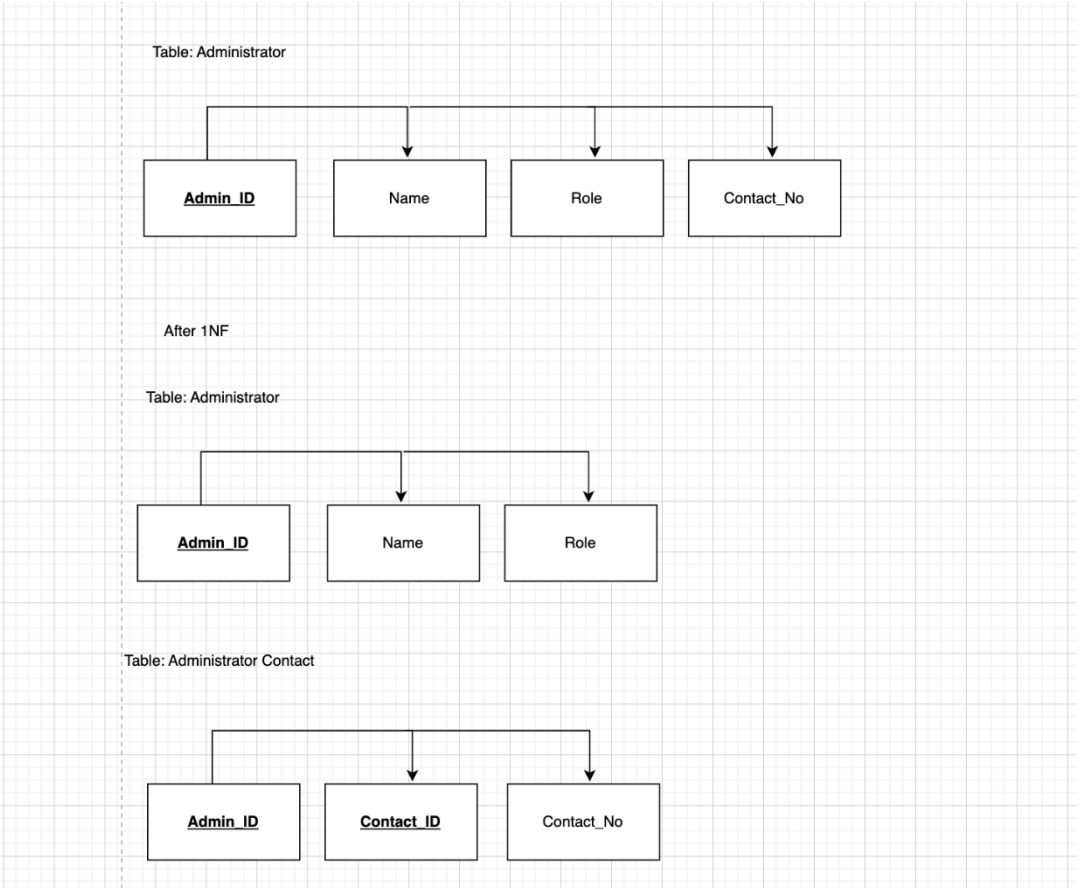

Figure 3: Adminstrator Table

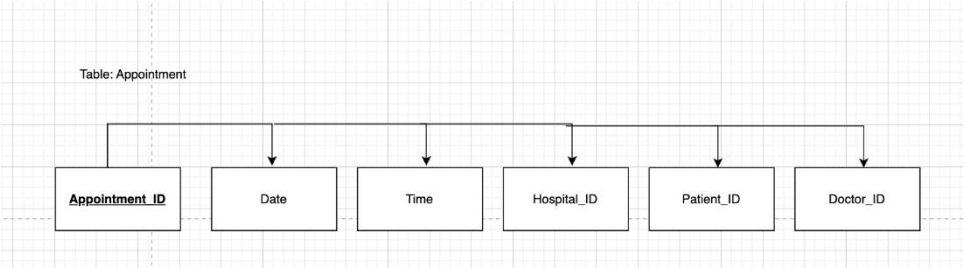

Figure 4: Appointment table

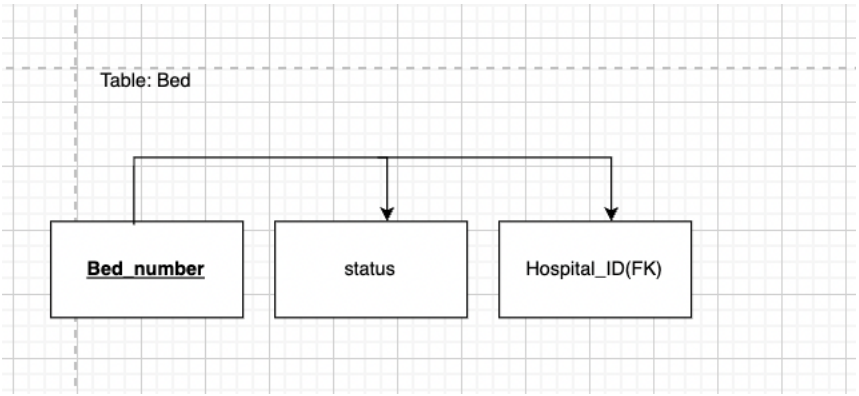

Figure 5: Bed table.

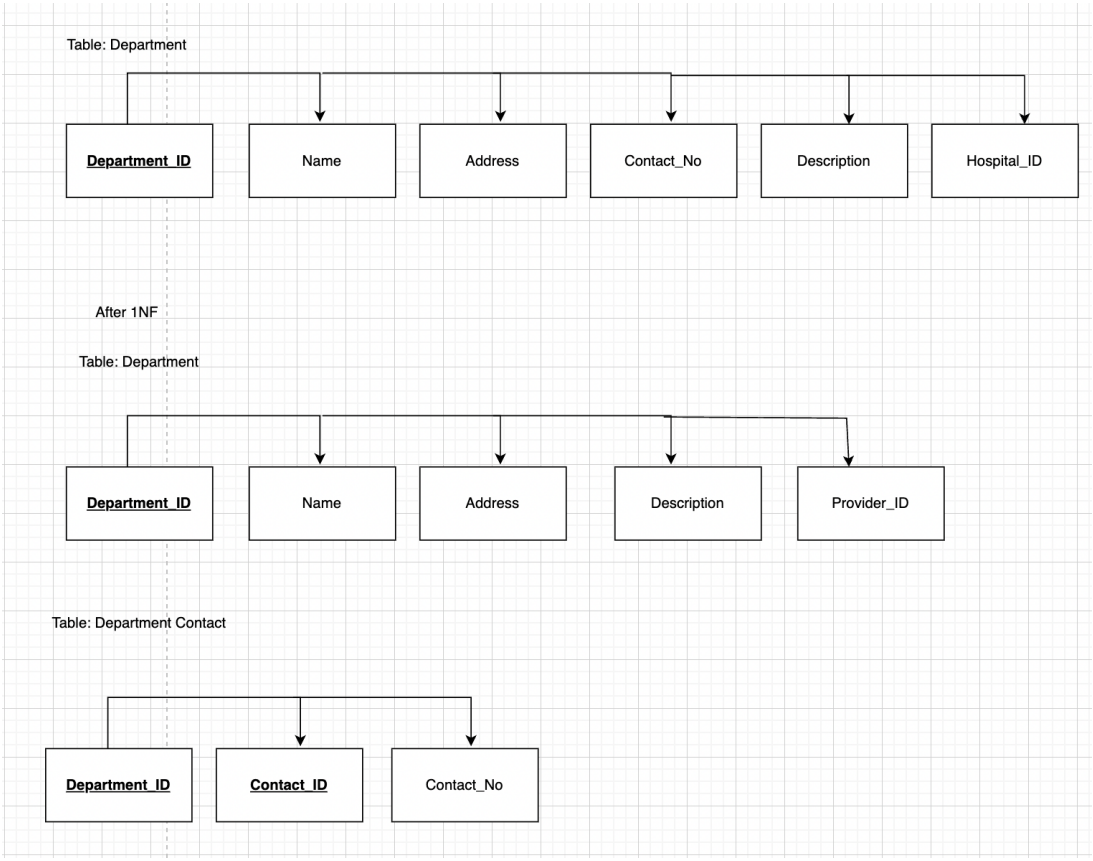

Figure 6: Department table.

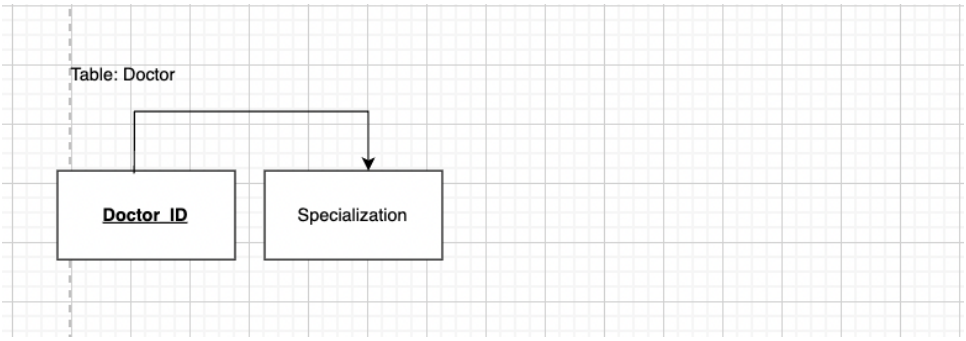

Figure 7: Doctor table.

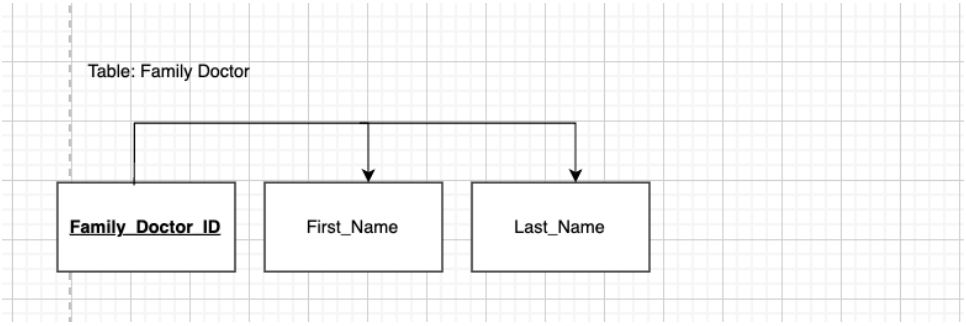

Figure 8: Family doctor table.

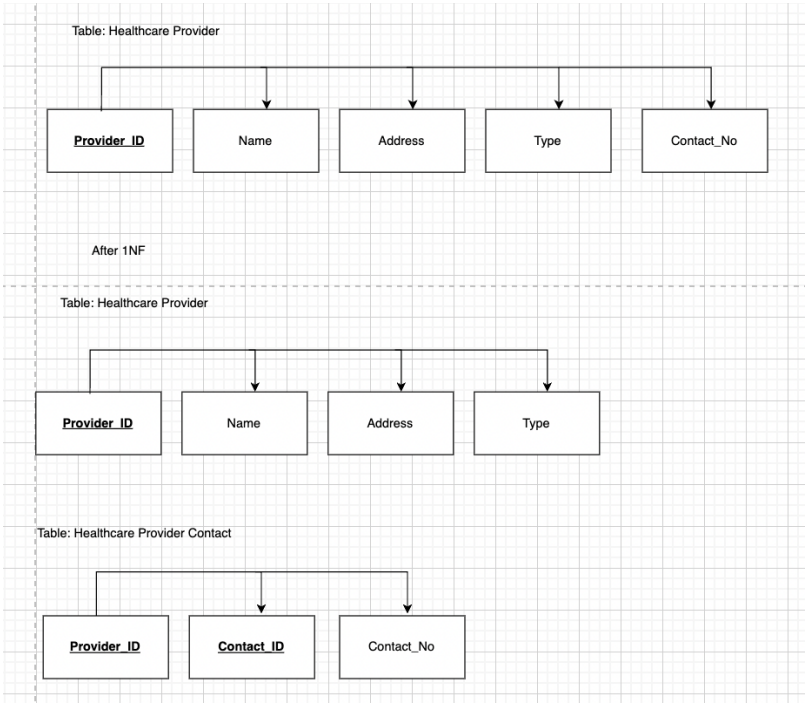

Figure 9: Healthcare Provider table.

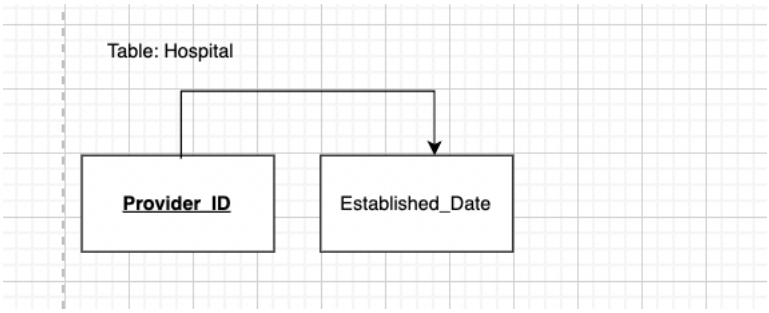

Figure 10: Hospital table.

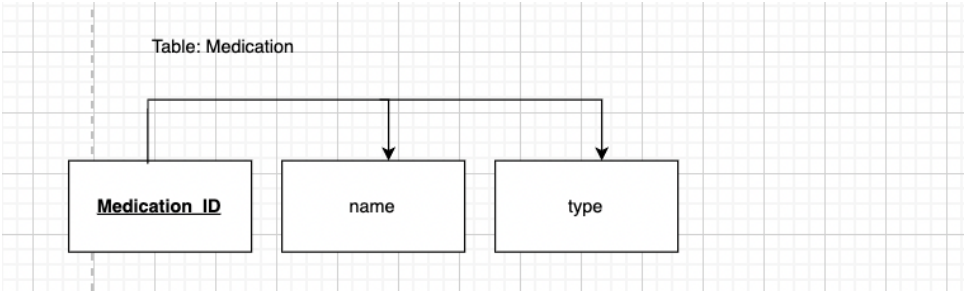

Figure 11: Medication table.

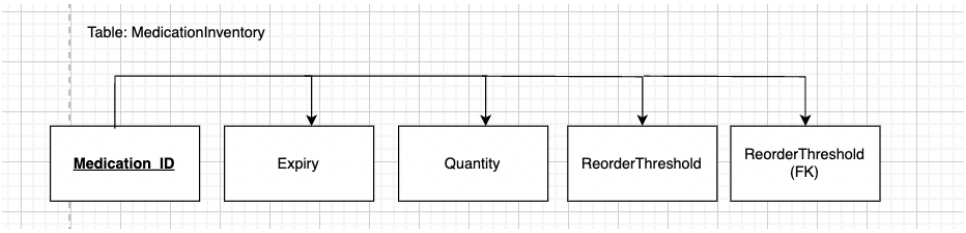

Figure 12: Medication Inventory table.

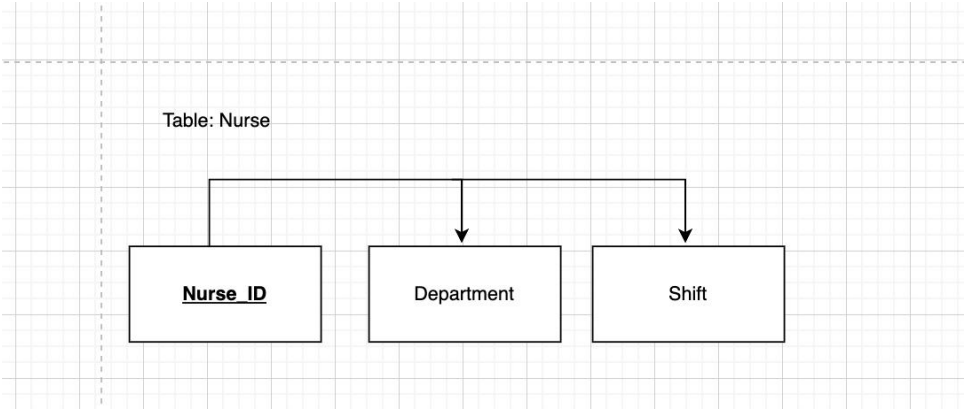

Figure 13: Nurse table.

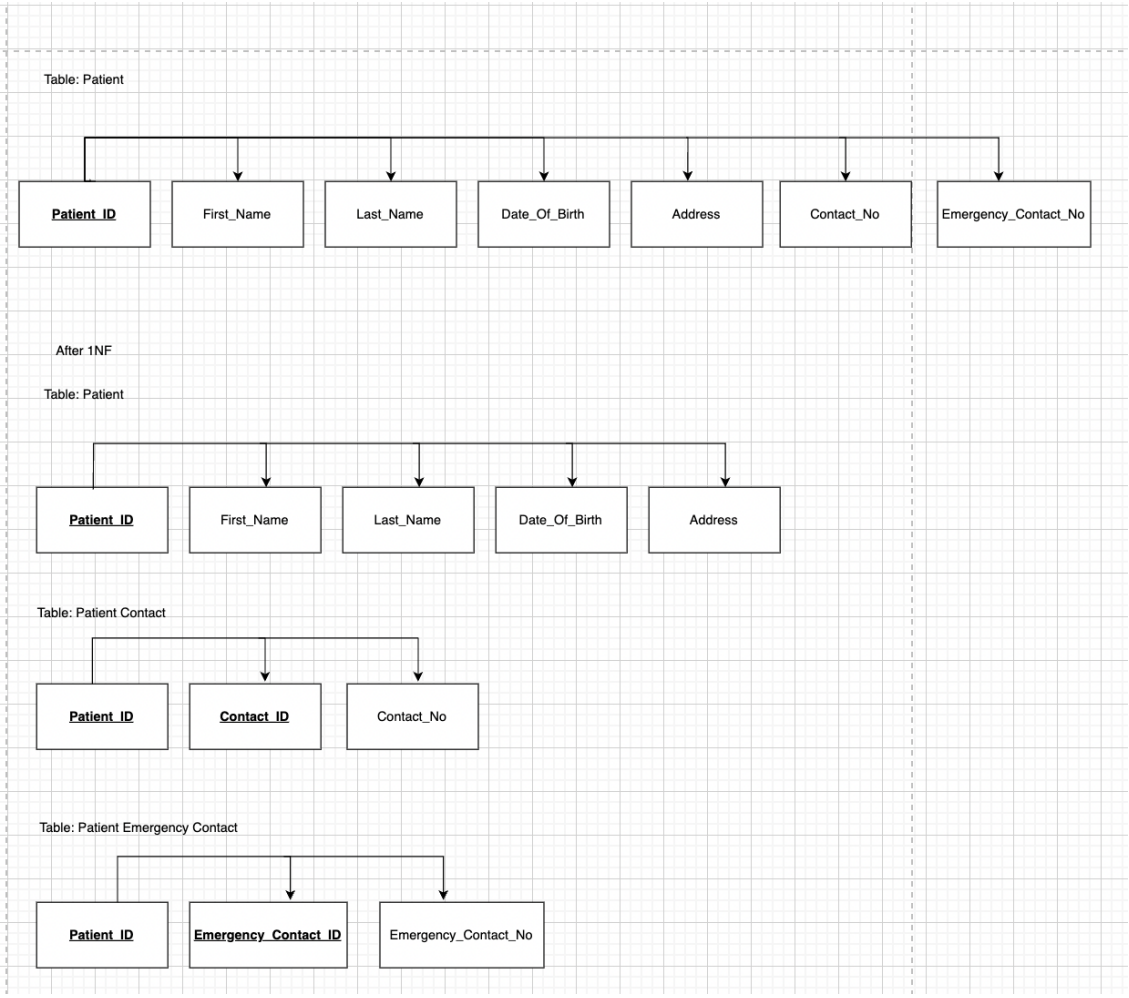

Figure 14: Patient table.

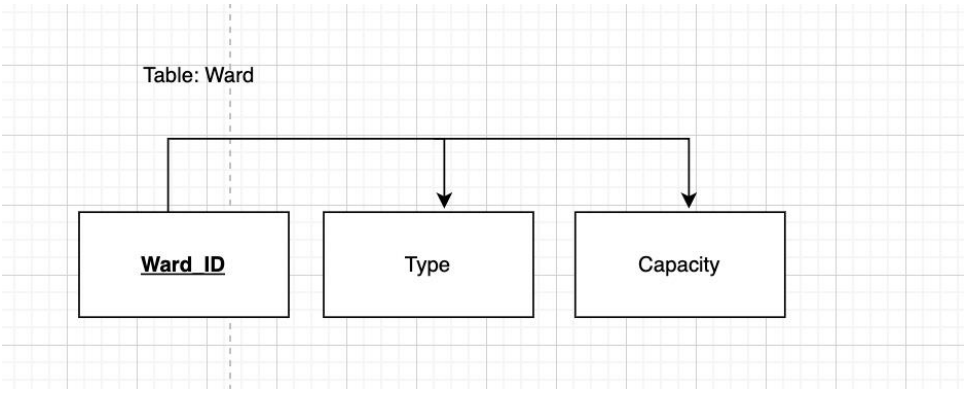

Figure 15: Ward table.

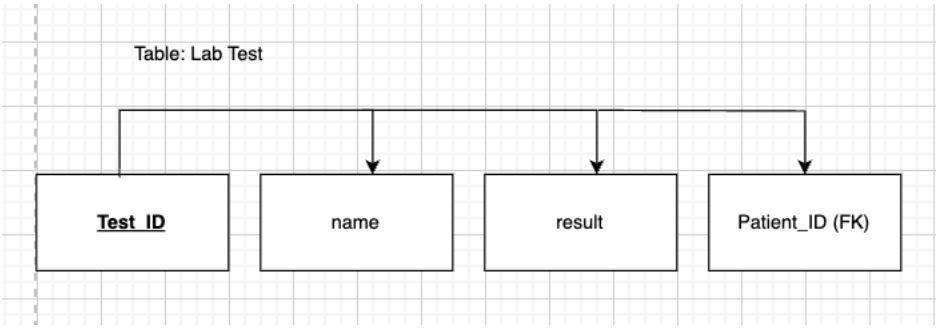

Figure 16: Lab test table.

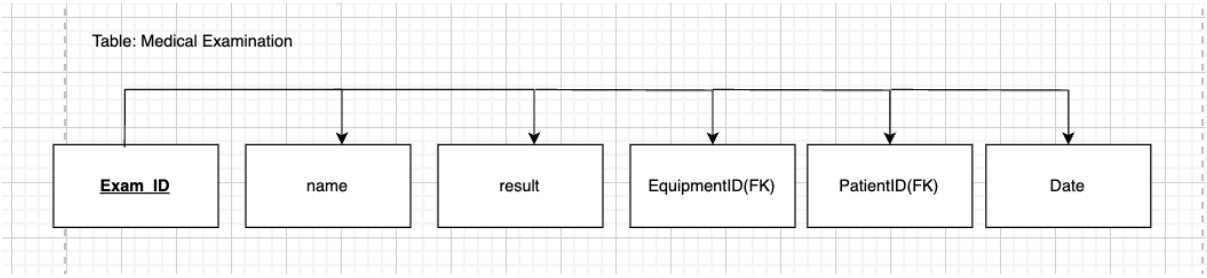

Figure 17: Medical examination table.

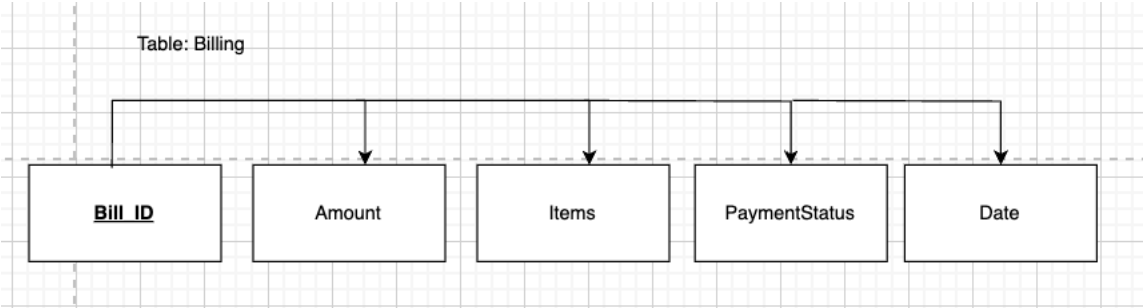

Figure 18: Billing table.

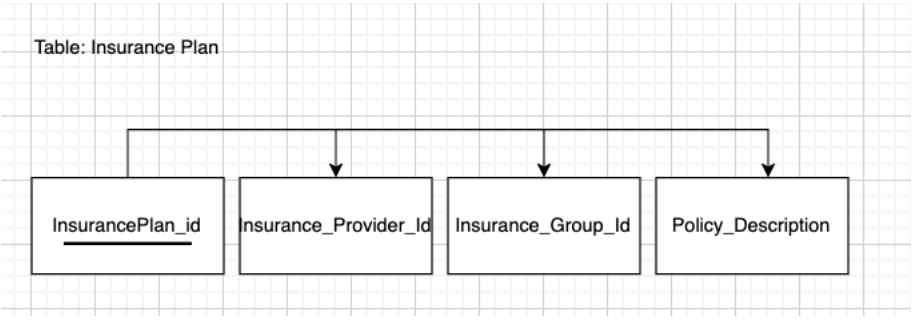

Figure 19: Insurance Plan table.

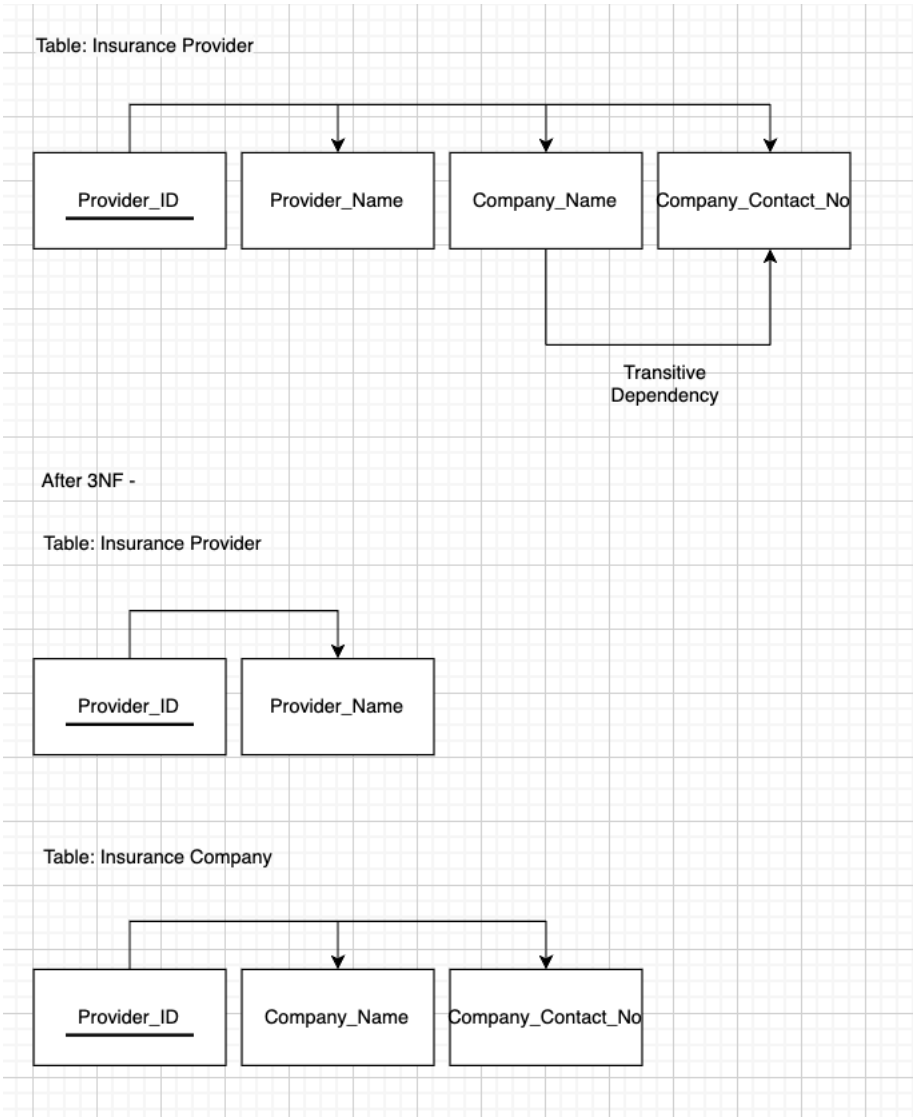

Figure 20: Insurance Provider table.

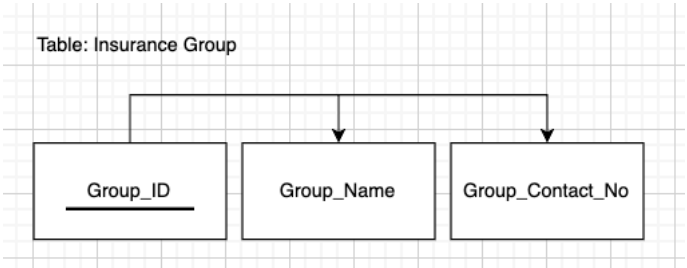

Figure 21: Insurance Group table.

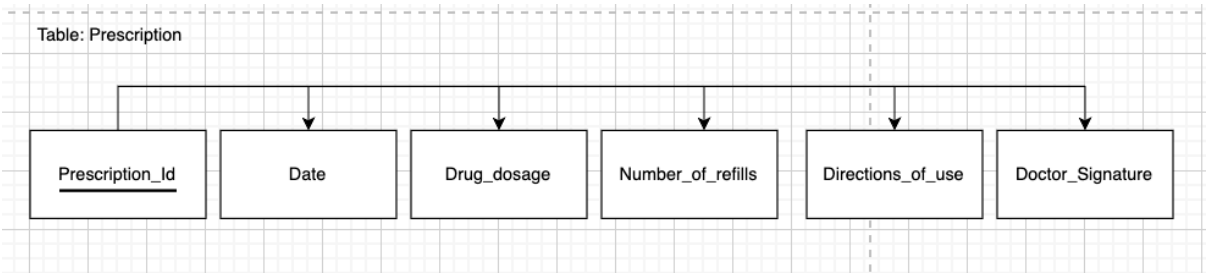

Figure 22: Prescription table.

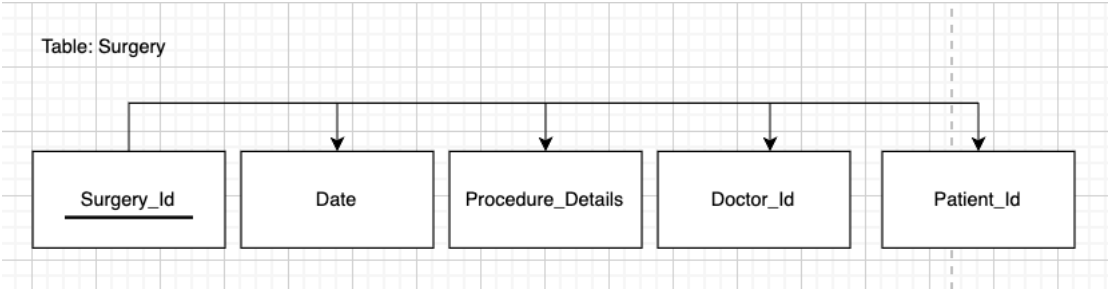

Figure 23: Surgery table.

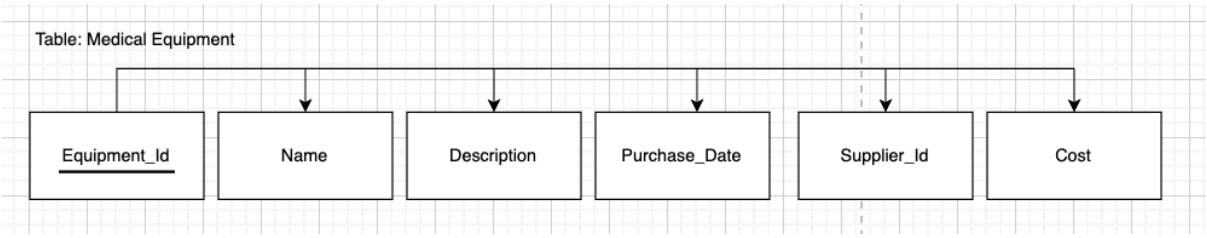

Figure 24: Medical Equipment table.

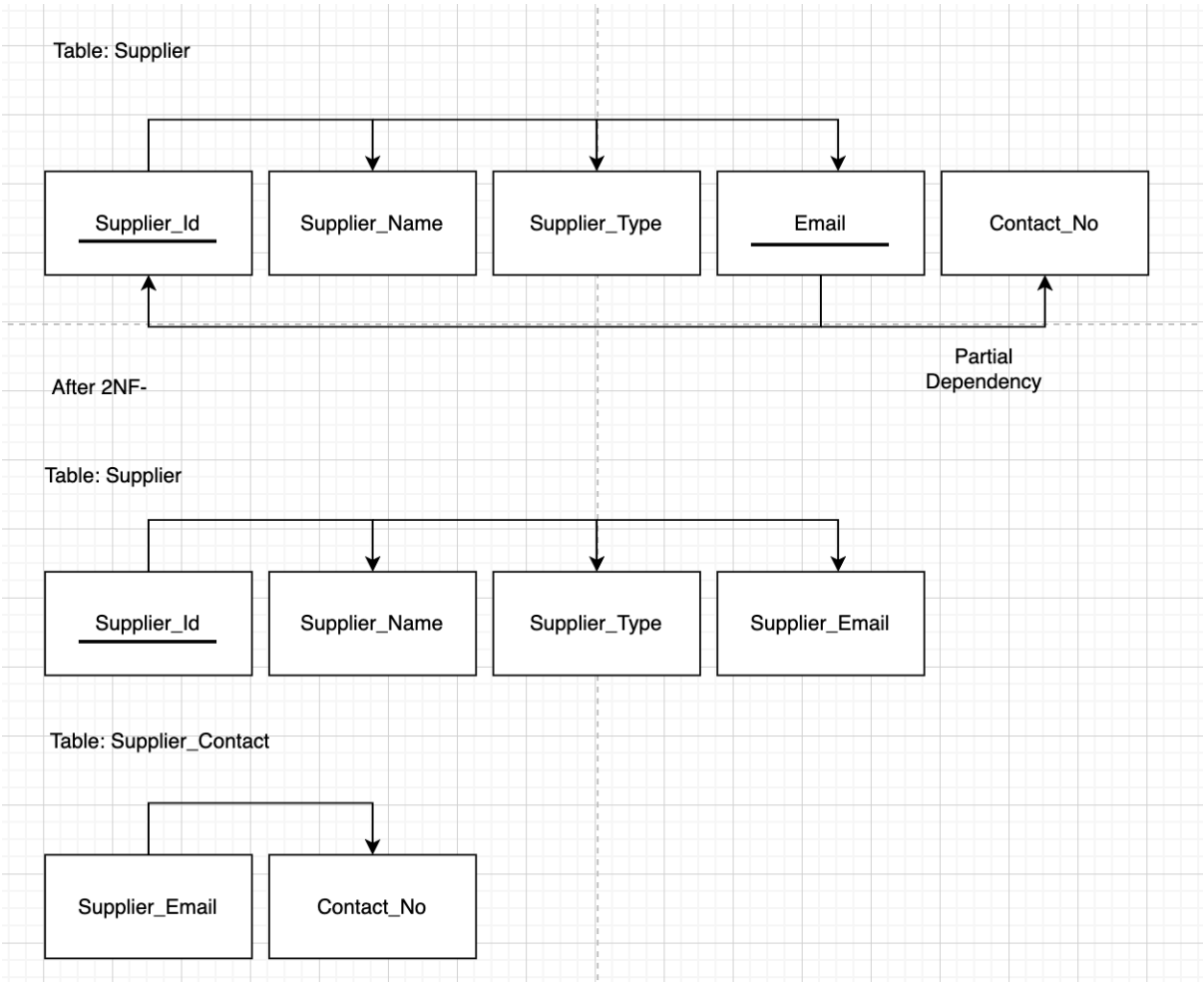

Figure 25: Supplier table.

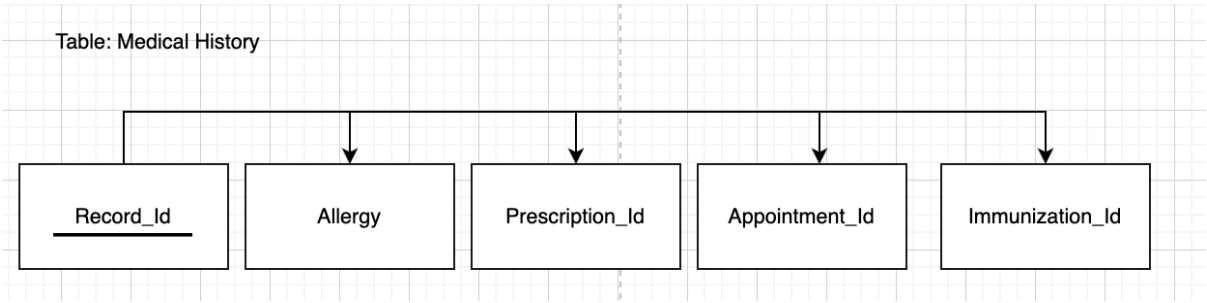

Figure 26: Medical History table.

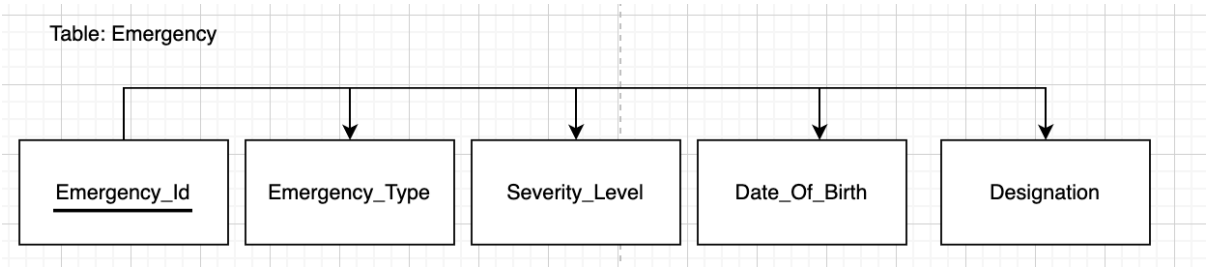

Figure 27: Emergency table.

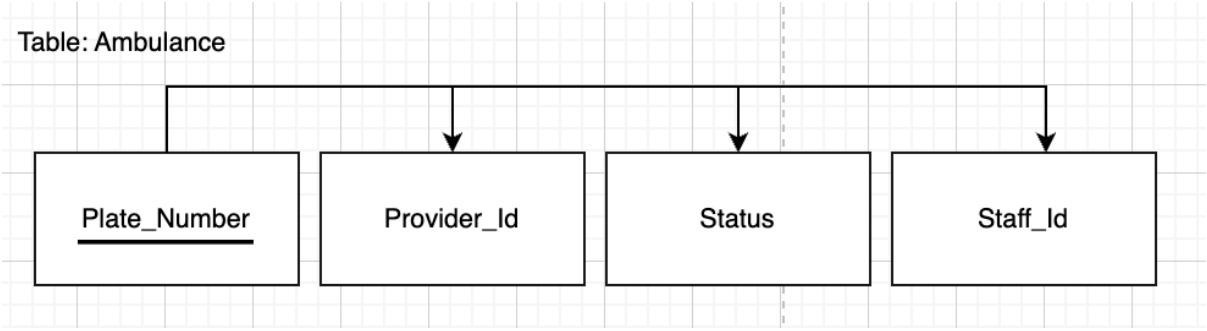

Figure 28: Ambulance table.

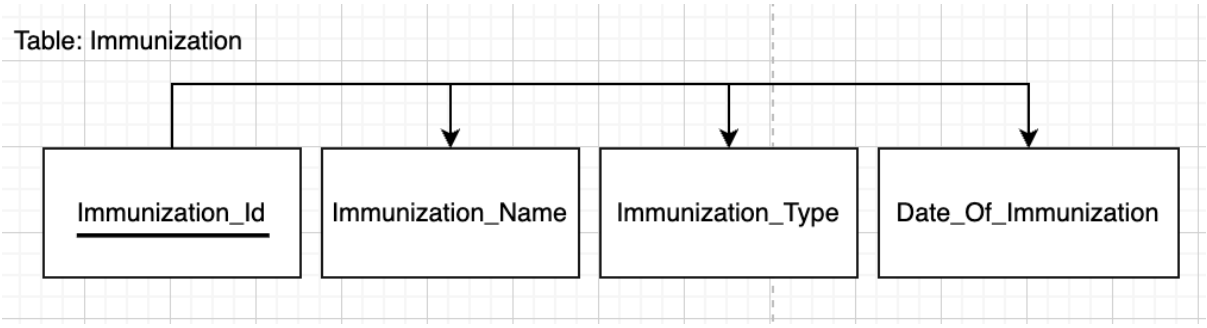

Figure 29: Immunization table.

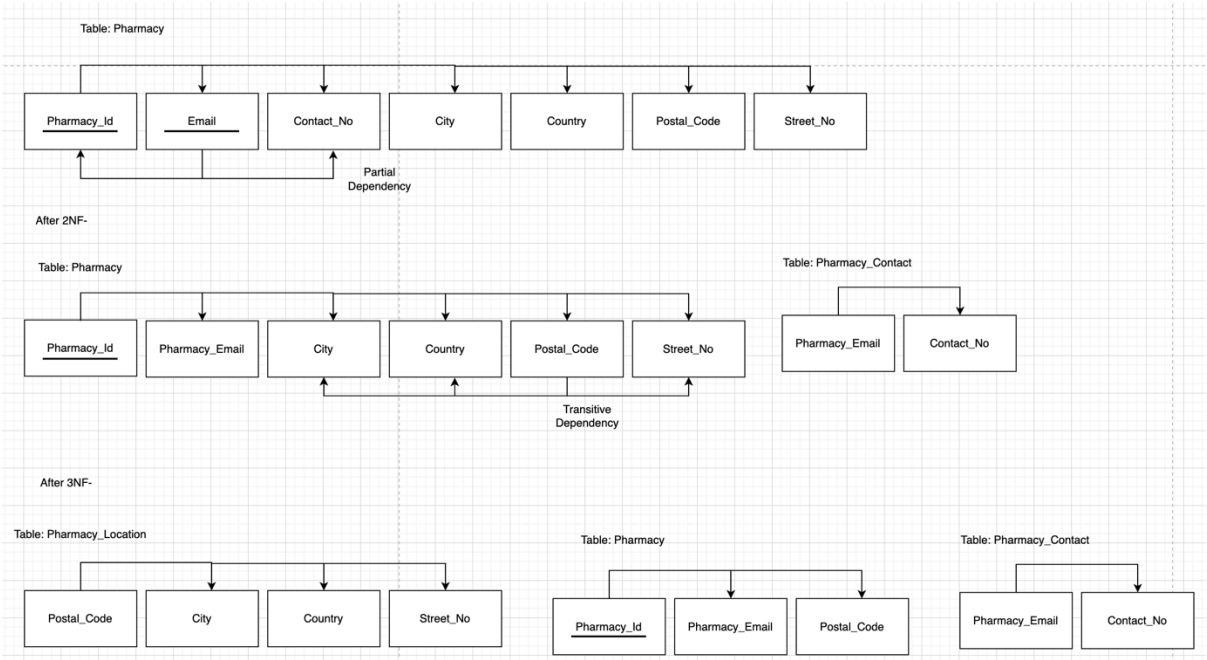

Figure 30: Pharmacy table.

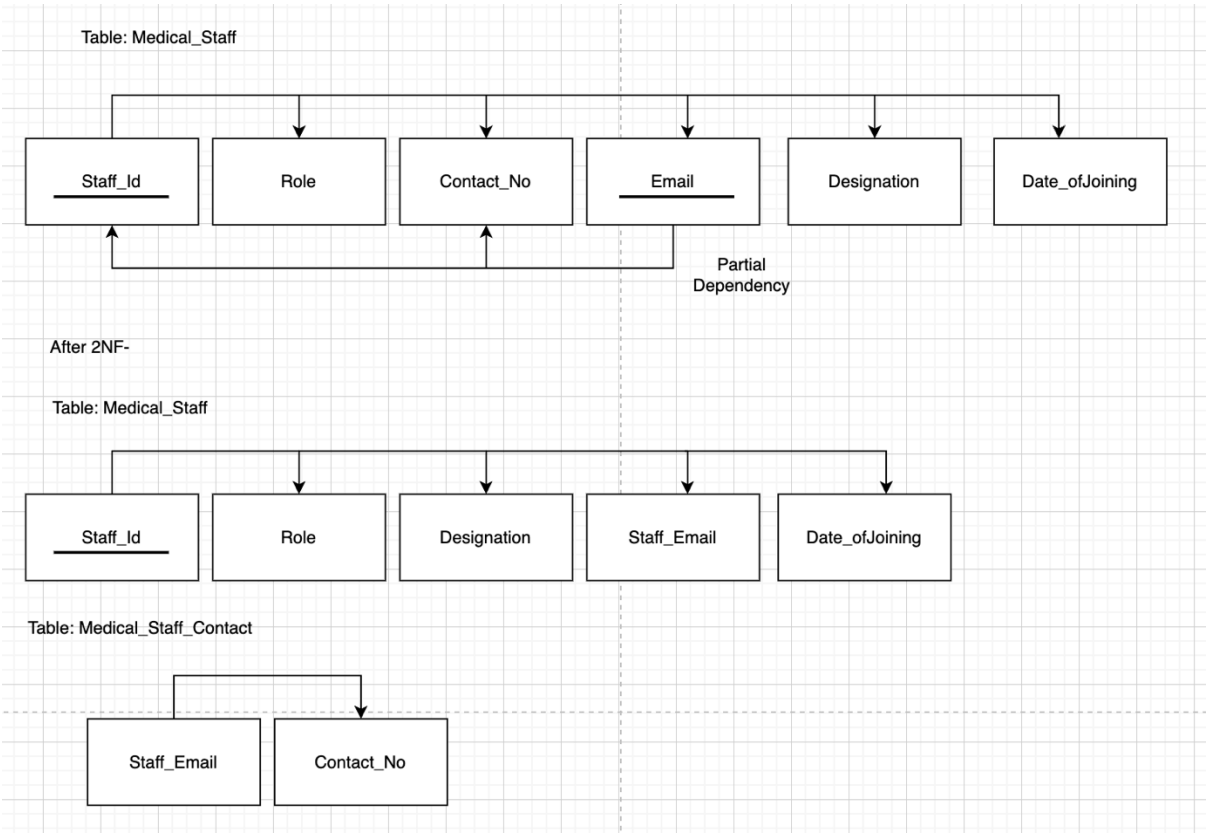

Figure 31: Medical Staff table.

1. First Normal Form:
   Tables such as administrator, department and healthcare provider were having contact number attribute which may contain more than one value. Hence, we had normalized tables to 1NF by creating separate tables for contact number.

2. Second Normal Form:
   While performing second normalization, we identified partial keys such as email within supplier and pharmacy tables.

3. Third Normal Form:
   At last, we normalized all tables into third normal form, by eradicating transitive dependencies found within few tables such as insurance provider and pharmacy.

DDL Statements

DDL Statements for VM1

1. *Create Database*

```
CREATE DATABASE VM1;
```

2. *Use Database*

```
USE VM1;
```

3. *Create Patient Information Table*

```
CREATE TABLE Patient (
  patient_id INT PRIMARY KEY,
  first_name VARCHAR(35),
  last_name VARCHAR(35),
  date_of_birth DATE,
  address VARCHAR(255)
);
```

4. *Create Patient Contact Information*

```
CREATE TABLE Patient_Contact (
  patient_id INT,
  contact_id INT AUTO_INCREMENT PRIMARY KEY,
  contact_no VARCHAR(15),
  FOREIGN KEY (patient_id) REFERENCES Patient(patient_id)
);
```

5. *Create Healthcare Provider Information*

```
CREATE TABLE Healthcare_Provider (
  healthcare_provider_id INT PRIMARY KEY,
  name VARCHAR(35),
  address VARCHAR(255),
  type VARCHAR(10)
);
```

6. *Create Healthcare Provider Contact Information*

```
CREATE TABLE Healthcare_Provider_Contact (
  healthcare_provider_id INT,
  contact_id INT AUTO_INCREMENT PRIMARY KEY,
  contact_no VARCHAR(15),
  FOREIGN KEY (healthcare_provider_id) REFERENCES
Healthcare_Provider(healthcare_provider_id)
);
```

7. *Create Hospital Information*

```
CREATE TABLE Hospital (
  hospital_id INT PRIMARY KEY,
  healthcare_provider_id INT,
  FOREIGN KEY (healthcare_provider_id) REFERENCES
Healthcare_Provider(healthcare_provider_id)
);
```

8. *Create Hospital Information*

```
CREATE TABLE Family_Doctor (
  family_doctor_id INT PRIMARY KEY,
  healthcare_provider_id INT,
  FOREIGN KEY (healthcare_provider_id) REFERENCES
Healthcare_Provider(healthcare_provider_id)
);
```

9. *Create Doctor Information*

```
CREATE TABLE Doctor (
  doctor_id INT PRIMARY KEY,
  specialization VARCHAR(50)
);
```

10. *Create Nurse Information*

```
CREATE TABLE Nurse (
  nurse_id INT PRIMARY KEY,
  name VARCHAR(50),
  shift VARCHAR(50),
  contact_details VARCHAR(100)
);
```

11. *Create Administrator Information*

```
CREATE TABLE Administrator (
  admin_id INT PRIMARY KEY,
  name VARCHAR(50),
  role VARCHAR(50),
  contact_details VARCHAR(100)
);
```

12. *Create Appointment Information*

```
CREATE TABLE Appointment (
  appointment_id INT PRIMARY KEY,
  date DATE,
  time TIME,
  patient_id INT,
  doctor_id INT,
  FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
  FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id));
```

13. *Create Department Information*

```
CREATE TABLE Department (
  department_id INT PRIMARY KEY,
  hospital_id INT,
  description TEXT,
  FOREIGN KEY (hospital_id) REFERENCES Hospital(hospital_id)
);
```

14. *Create Ward Information*

```
CREATE TABLE Ward (
  ward_id INT PRIMARY KEY,
  hospital_id INT,
  type VARCHAR(50),
  capacity INT,
  FOREIGN KEY (hospital_id) REFERENCES Hospital(hospital_id)
);
```

15. *Create Nurse Ward Assignment*

```
CREATE TABLE NurseWard (
  nurse_id INT,
  ward_id INT,
  assignment_time TIMESTAMP,
  PRIMARY KEY (nurse_id, ward_id),
  FOREIGN KEY (nurse_id) REFERENCES Nurse(nurse_id),
  FOREIGN KEY (ward_id) REFERENCES Ward(ward_id)
);
```

16. *Create Bed Information*

```
CREATE TABLE Bed (
  Bed_number INT PRIMARY KEY,
  Status VARCHAR(20),
  Hospital_id INT,
  FOREIGN KEY (Hospital_id) REFERENCES Hospital(Hospital_id)
);
```

17. *Create Lab Test Information*

```
CREATE TABLE Lab_Test (
  Test_id INT PRIMARY KEY,
  Name VARCHAR(255),
  Result VARCHAR(255),
  TestDate DATE
);
```

18. *Create Medical Examination Information*

```
CREATE TABLE Medical_Examination (
  Exam_id INT PRIMARY KEY,
  Name VARCHAR(255),
  Result VARCHAR(255),
  Equipment_ID INT,
```

```
  Date DATE,
  PatientID INT,
  FOREIGN KEY (PatientID) REFERENCES Patient(patient_id)
);
```

19. *Create Billing Information*

```
CREATE TABLE Billing (
  Bill_id INT PRIMARY KEY,
  Amount DECIMAL(10, 2),
  Items VARCHAR(255),
  Payment_Status VARCHAR(20),
  Date DATE
);
```

20. *Create Surgery Information*

```
CREATE TABLE Surgery (
  Surgery_ID INT PRIMARY KEY,
  Date DATE,
  Procedure_Details LONGTEXT,
  Doctor_Id INT,
  Patient_Id INT,
  FOREIGN KEY (Doctor_Id) REFERENCES Doctor(doctor_id),
  FOREIGN KEY (Patient_Id) REFERENCES Patient(patient_id)
);
```

21. *Create Surgery Information*

```
CREATE TABLE Prescription (
  Prescription_ID INT PRIMARY KEY,
  Date DATE,
  Drug_dosage VARCHAR(255),
  Number_of_refills INT,
  Directions_of_use VARCHAR(255),
  Doctor_signature BLOB
);
```

22. *Create Immunization Information*

```
CREATE TABLE Immunization (
  Immunization_Id INT PRIMARY KEY,
  Immunization_Name VARCHAR(100),
  Immunization_Type VARCHAR(50),
  Date_Of_Immunization DATE);
```

23. *Create Medical History Information*

```
CREATE TABLE Medical_Histry (
  Record_Id INT PRIMARY KEY,
  Allergy VARCHAR(255),
  Prescription_Id INT,
  Appointment_Id INT,
  Immunization_Id INT,
  FOREIGN KEY (Prescription_Id) REFERENCES
Prescription(Prescription_ID),
  FOREIGN KEY (Appointment_Id) REFERENCES
Appointment(appointment_id),
  FOREIGN KEY (Immunization_Id) REFERENCES
Immunization(Immunization_Id)
);
```

24. *Create Medical Staff Information*

```
CREATE TABLE Medical_Staff (
  Staff_Id INT PRIMARY KEY,
  Role VARCHAR(50),
  Staff_Email VARCHAR(255) UNIQUE,
  Designation VARCHAR(100)
);
```

25. *Create Medical Staff Contact Information*

```
CREATE TABLE Medical_Staff_Contact (
  Staff_Email VARCHAR(255) PRIMARY KEY,
  Contact_No VARCHAR(15),
  Date_Of_Joining DATE,
  FOREIGN KEY (Staff_Email) REFERENCES Medical_Staff(Staff_Email)
);
```

26. *Create Medical Emergency Information*

```
CREATE TABLE Emergency (
  Emergency_Id INT PRIMARY KEY,
  Emergency_Type VARCHAR(50),
  Severity_Level VARCHAR(50),
  Date_Of_Birth DATE,
  Patient_Id INT,
  Doctor_Id INT,
  FOREIGN KEY (Patient_Id) REFERENCES Patient(patient_id),
  FOREIGN KEY (Doctor_Id) REFERENCES Doctor(doctor_id)
);
```

27. *Create Ambulance Information*

```
CREATE TABLE Ambulance (
  Plate_Number VARCHAR(20) PRIMARY KEY,
  Status VARCHAR(50),
  Provider_Id INT,
  FOREIGN KEY (Provider_Id) REFERENCES
Healthcare_Provider(healthcare_provider_id)
```

DDL Statements for VM2

1. *Create Database*

```
CREATE DATABASE VM2;
```

2. *Use Database*

```
USE VM2;
```

3. *Create Supplier Information*

```
CREATE TABLE Supplier (
  Supplier_Id INT PRIMARY KEY,
  Supplier_Name VARCHAR(255),
  Supplier_Email VARCHAR(255) UNIQUE,
  Supplier_Type VARCHAR(50)
);
```

4. *Create Supplier Information*

```
CREATE TABLE Supplier_Contact (
  Contact_id INT AUTO_INCREMENT PRIMARY KEY,
  Supplier_Email VARCHAR(255),
  Contact_no VARCHAR(15),
  FOREIGN KEY (Supplier_Email) REFERENCES Supplier(Supplier_Email)
);
```

5. *Create Pharmacy Information*

```
CREATE TABLE Pharmacy (
  Pharmacy_Id INT PRIMARY KEY,
  Postal_Code VARCHAR(20) UNIQUE,
  Pharmacy_Email VARCHAR(255) UNIQUE
);
```

6. *Create Pharmacy Information*

```
CREATE TABLE Pharmacy_Contact (
  Pharmacy_Email VARCHAR(255) PRIMARY KEY,
  Contact_No VARCHAR(15),
  FOREIGN KEY (Pharmacy_Email) REFERENCES Pharmacy(Pharmacy_Email)
);
```

7. *Create Medication Inventory Information*

```
CREATE TABLE Medication_Inventory (
  medication_id INT,
  Expiry DATE,
  Quantity INT,
  Reorder_Threshold INT,
  Pharmacy_Id INT,
  PRIMARY KEY (medication_id, Pharmacy_Id),
  FOREIGN KEY (medication_id) REFERENCES Medication(medication_id),
  FOREIGN KEY (Pharmacy_Id) REFERENCES Pharmacy(Pharmacy_Id)
);
```

8. *Create Medication Equipment Information*

```
CREATE TABLE Medical_Equipment (
  Equipment_Id INT PRIMARY KEY,
  Name VARCHAR(255),
  Description LONGTEXT,
  Purchase_Date DATE,
  Supplier_ID INT,
  Cost DECIMAL(10,2),
  FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID)
);
```

9. *Create Insurance Provider Information*

```
CREATE TABLE Insurance_Provider (
  Provider_ID INT PRIMARY KEY,
  Provider_Name VARCHAR(255)
);
```

10. Create Insurance Provider Information

```
CREATE TABLE Insurance_Company (
  Provider_ID INT,
  Company_Name VARCHAR(255),
  Company_Contact_Number VARCHAR(15),
  FOREIGN KEY (Provider_ID) REFERENCES
Insurance_Provider(Provider_ID)
);
```

11. Create Insurance Group Information

```
CREATE TABLE Insurance_Group (
  Group_ID INT PRIMARY KEY,
  Group_Name VARCHAR(255),
  Group_Contact_Number VARCHAR(15)
);
```

12. Create Insurance Plan Information

```
CREATE TABLE Insurance_Plan (
  InsurancePlan_id INT PRIMARY KEY,
  Insurance_Provider_ID INT,
  Insurance_Group_ID INT,
  Policy_Details VARCHAR(255),
  FOREIGN KEY (Insurance_Provider_ID) REFERENCES
Insurance_Provider(Provider_ID),
  FOREIGN KEY (Insurance_Group_ID) REFERENCES
Insurance_Group(Group_ID)
);
```

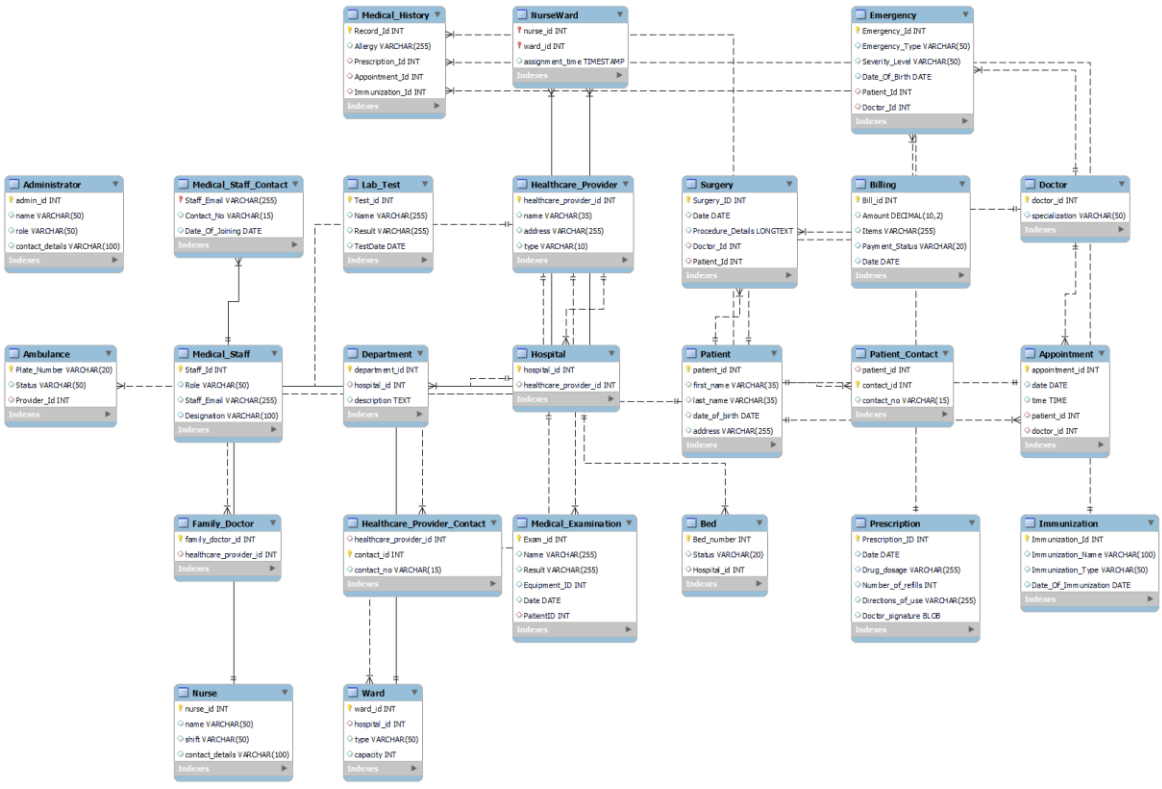

Figure 32: Data Model for VM1.

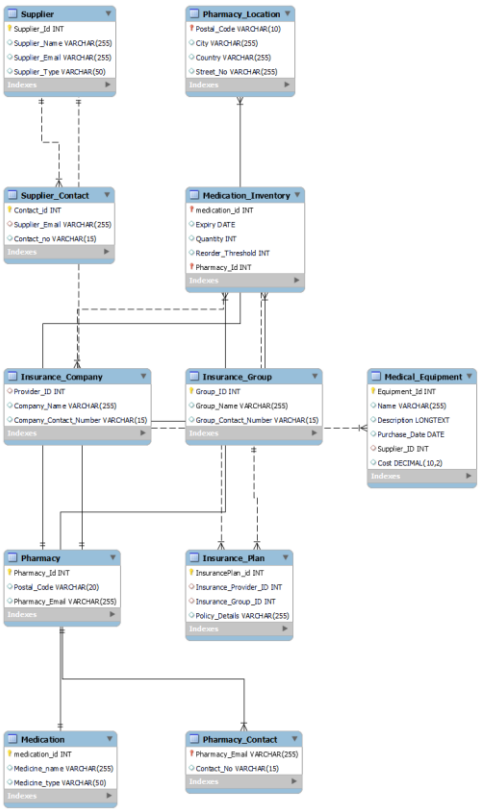

Figure 33: Data Model for VM2

Distributed Database

Fragmentation Decision

We used database-level fragmentation. We divided the tables into two VMs based on the nature of the information stored.

VM1: Table of patients, healthcare services and direct correlation

1. According to the background of our search and based on our traditional experience, we believe that patient information, healthcare providers, treatment appointments and healthcare personnel are more highly correlated, and this information are closely related. Putting them into the same database can reduce the number of cross-database queries and ensure the integrity and unity of the data.

2. The query frequency of these data is higher than that of other tables. Putting high-frequency data into the same table can simplify the query logic and reduce latency to improve performance.

3. Some of the foreign keys in these tables depend on each other, so we put the tables with the same dependencies together.

VM1 tables

- Patient
- Patient_Contact
- Healthcare_Provider
- Healthcare_Provider_Contact
- Hospital
- Family_Doctor
- Doctor
- Nurse
- Administrator
- Appointment
- Department
- Ward
- NurseWard
- Bed
- Lab_Test
- Medical_Examination
- Billing
- Surgery
- Prescription
- Immunization
- Medical_History
- Medical_Staff
- Medical_Staff_Contact
- Emergency
- Ambulance

VM2: Medicines, supply chain and insurance information

1. Drug inventory, medical device, and supplier information, etc., may not be updated as frequently as patient information, so we keep them in the same table.
2. This information often interacts with other third parties, e.g. insurance companies, drug suppliers etc. Keeping them in the same table makes it easier to handle related events.

VM2 tables

- Supplier
- Supplier_Contact
- Pharmacy
- Pharmacy_Contact
- Pharmacy_Location
- Medication
- Medication_Inventory
- Medical_Equipment
- Insurance_Provider
- Insurance_Company
- Insurance_Group
- Insurance_Plan

Process of Distributed Database Creation

We created two VM instances on GCP namely vmsql1 and vmsql2.

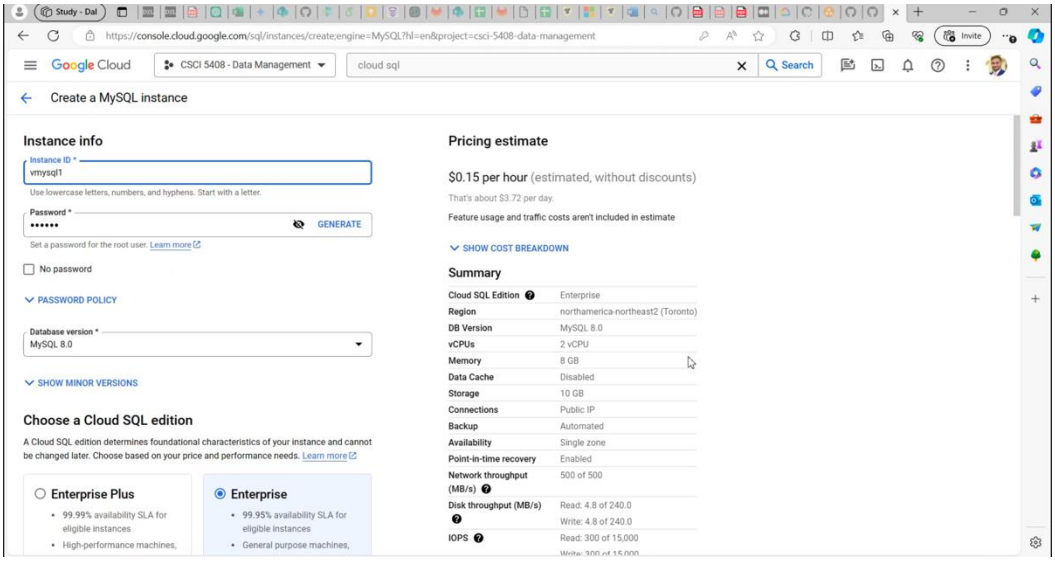

Figure 34: VMSQL1 instance creation.

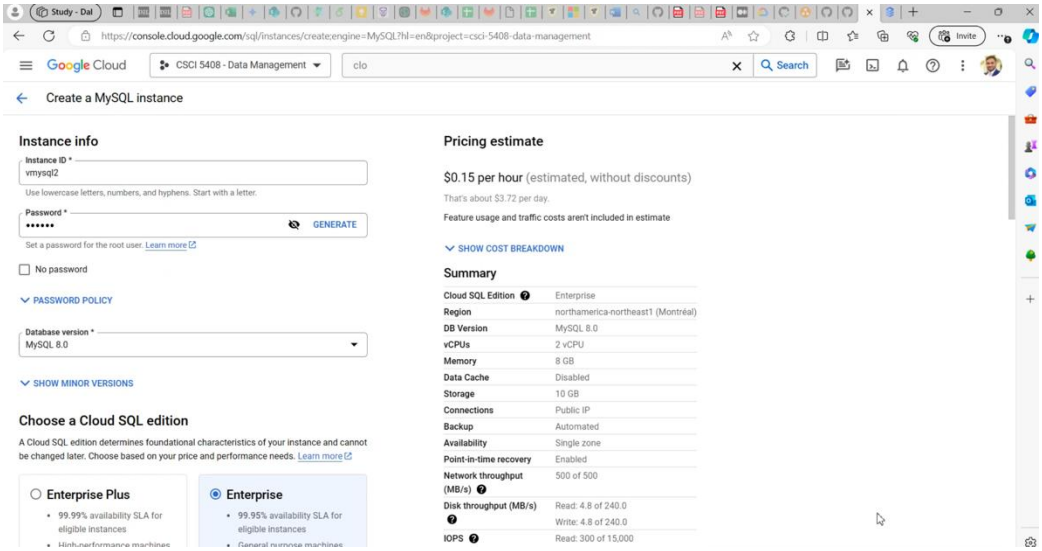

Figure 35: VMSQL2 Instance creation.

After generation of VM instances, connection was made with local database using the IP address and password.

For implementing, the distributed database feature we had created VM instance 1 on northamerica-northeast2 (Toronto) and VM instance on northamerica-northeast1 (Montreal) regions.

We have adopted homogenous classification as the VMs run on same MySQL version.

Structure & Placement of GDC

Table 2: GDC structure

VM	IP Address	Database	Tables
$VM1	34.130.170.253:3306	$VM1$Patient	Patient_Contact, Healthcare_Provider, Healthcare_Provider_Contact, Hospital, Family_Doctor, Doctor, Nurse, Administrator, Appointment, Department, Ward, NurseWard, Bed, Lab_Test, Medical_Examination , Billing, Surgery , Medical_History , Medical_Staff, Medical_Staff_Contact , Prescription, Emergency, Immunization, Ambulance
$VM2	34.118.154.112:3306	$VM2$Supplier	Supplier_Contact, Pharmacy, Pharmacy_Contact, Pharmacy_Location , Medication, Medication_Inventory, Medical_Equipment , Insurance_Provider , Insurance_Company, Insurance_Group, Insurance_Plan

The GDC needs to be built in a way that is efficient. safe and legal. [2]

1. Therefore, based on the rationale for the split table above. We set up the GDC to provide, efficient query management and ensure data consistency. For easy routing of information, the GDC structure consists of VM, IP address, Database, and tables fields. As per the scope of our project, these fields were providing useful information thus ensuring efficient retrieval of information.
2. We set up the databases in Montreal and Toronto, which are in the central and eastern parts of Canada, respectively, to cover our query requests in Canada over a wide range. This geographical distribution helps to reduce data access latency and improve application performance.
3. Keeping the Canadian data within Canada improves security.

4. By deploying VMs in different geographic locations, we can load balance the system and improve fault tolerance. This ensures that the GDC remains highly available when a VM encounters a failure.
5. We have placed GDC at a separate location for faster query execution. As synchronization would have been required if we had placed GDC within VM locations. In this way, latency will also be reduced. Placing GDC separately would make updating, replication, and maintenance easier. [3]

Novelty

We have implemented java program that takes SQL query as user input and fetches for the table name within the GDC.

Once the table is found within the VMs, given SQL query is executed.

The program identifies following SQL commands-

1) SELECT
2) INSERT
3) UPDATE
4) DELETE

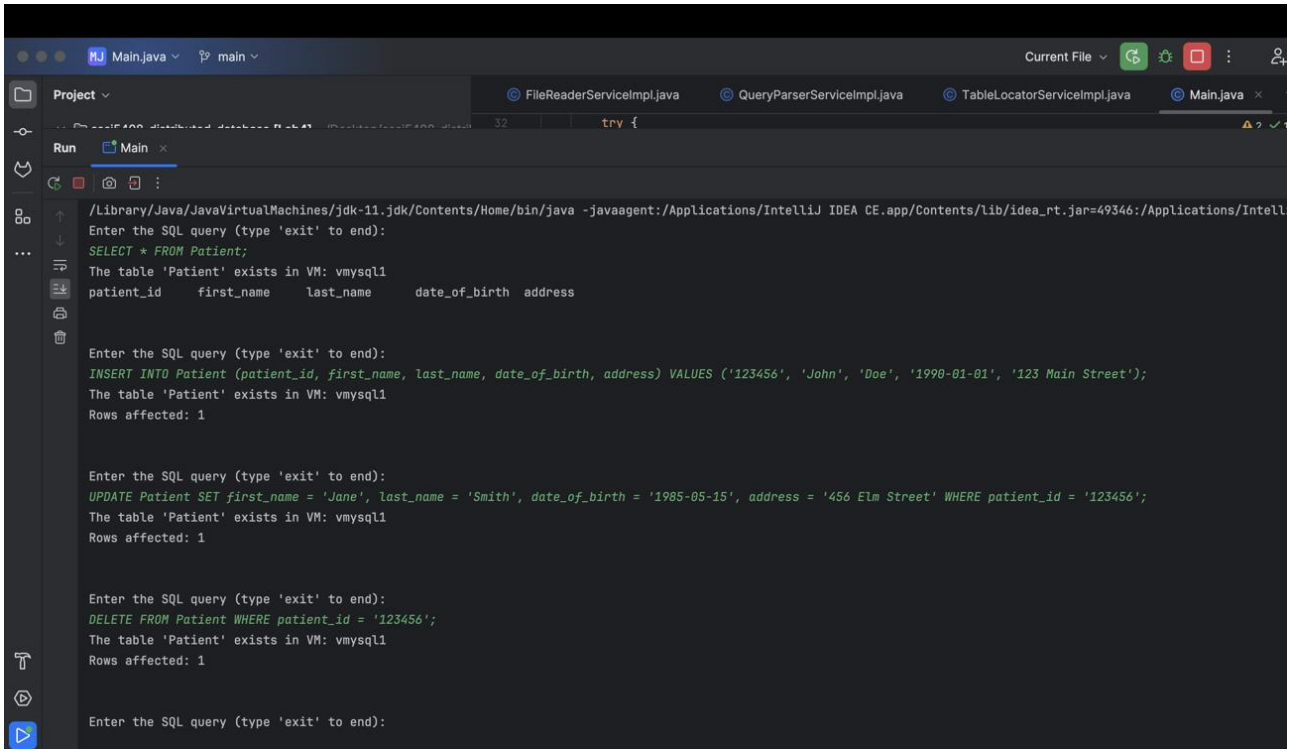

Figure 36: Test Case 1: - Testing execution of SQL queries within tables of VM1

1. *Patient table after insert query*

```
INSERT INTO Patient (patient_id, first_name, last_name, date_of_birth,
address)
VALUES ('123456', 'John', 'Doe', '1990-01-01', '123 Main Street');
```

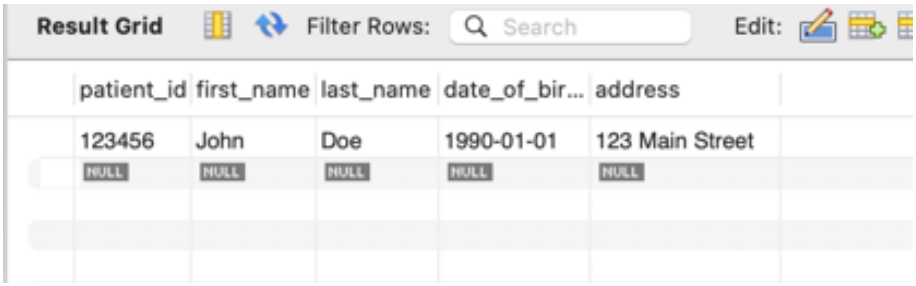

Figure 37: Patient table after insert query

2. *Patient table after update query*

```
UPDATE Patient
SET first_name = 'Jane',
    last_name = 'Smith',
    date_of_birth = '1985-05-15',
    address = '456 Elm Street'
WHERE patient_id = '123456';
```

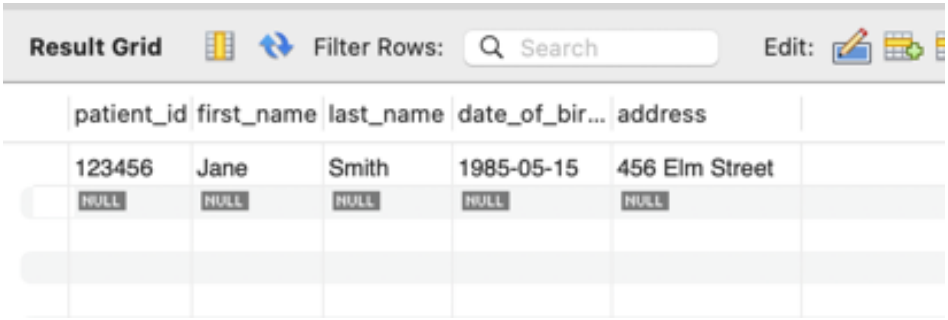

Figure 38: Patient table after update query.

3. *Patient table after delete query*

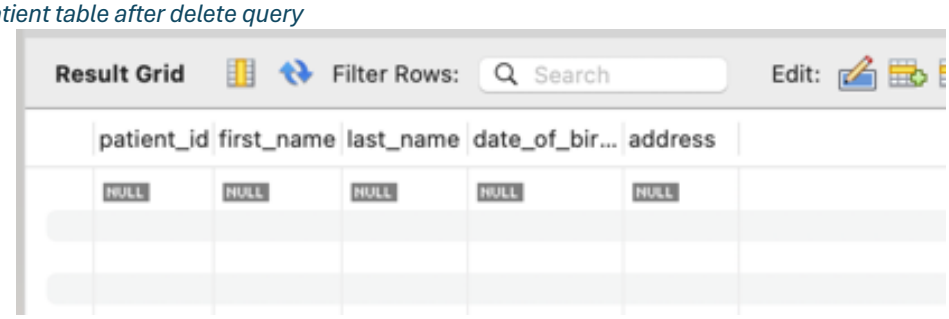

Figure 39: Patient table after delete query

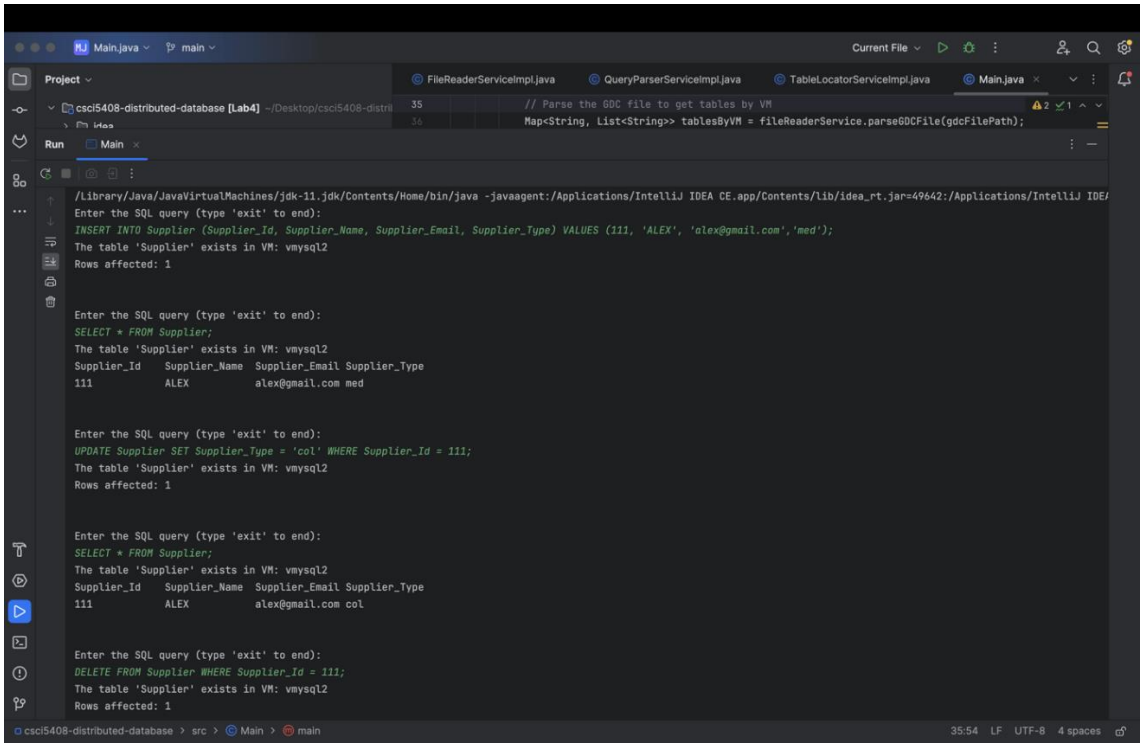

Figure 40: Test case 2 - Testing execution of SQL queries within tables of VM2

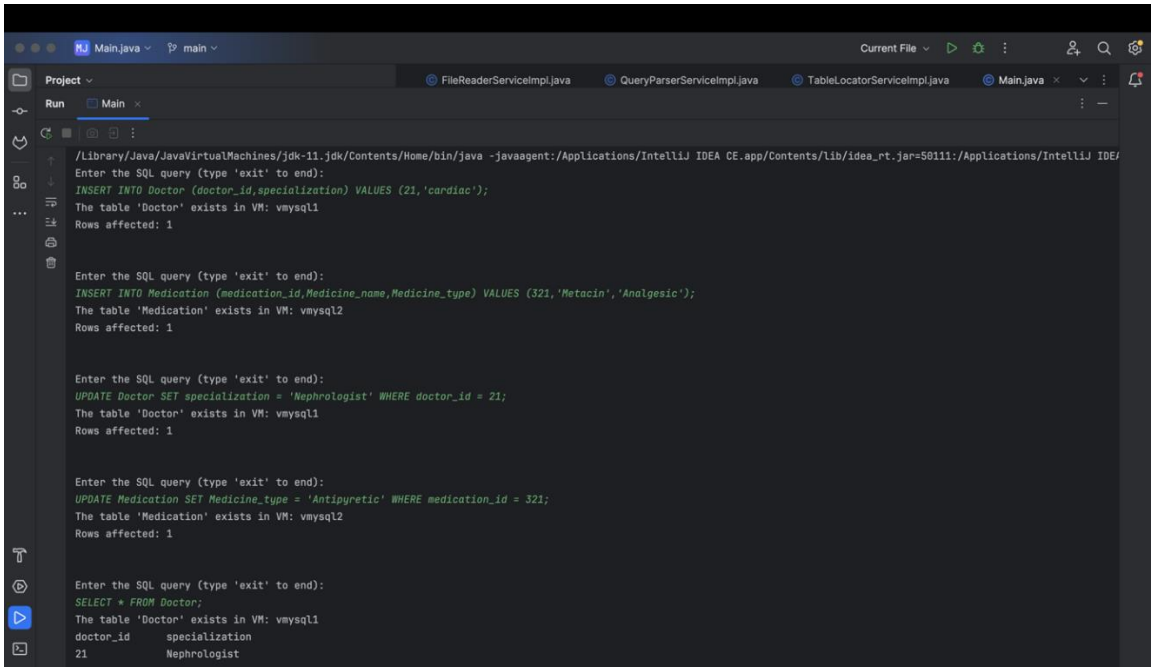

Figure 41: Test case 2 - Testing execution of SQL queries within tables of VM2

Figure 42: Test case 4 - Testing execution of SQL queries within tables of both VMs

Minutes of the meeting

Table 3: minutes of the meeting from meeting 1

Category	**Details**
Date/Time	Monday, February 19, 2024, 1:00 PM
Duration	45 minutes
Location	Online (MS Teams)
Attendees	All team members
Agenda	• Presentation of background research on the healthcare system. • Discussion on entities and relationships for ERD • Decision on healthcare industry scope.
Action Items	• All members to add research, entities, and attributes to shared OneDrive document. • Raisa to update on head TA's guidance. • All members to add to ERD shared by Haoyu. • Schedule next meeting for collaborative ERD design problem-solving.
Discussions	• Reviewed each member's research on the healthcare system. • Discussed scope: general healthcare system vs. hospital specific. • Clarification sought from head TA Bharat.
Outcomes	• Majority voted for healthcare system as the project scope. • Agreed to further refine entities and relationships in the next meeting. • Next meeting scheduled for Wednesday, February 21, 2024, 1:30 PM

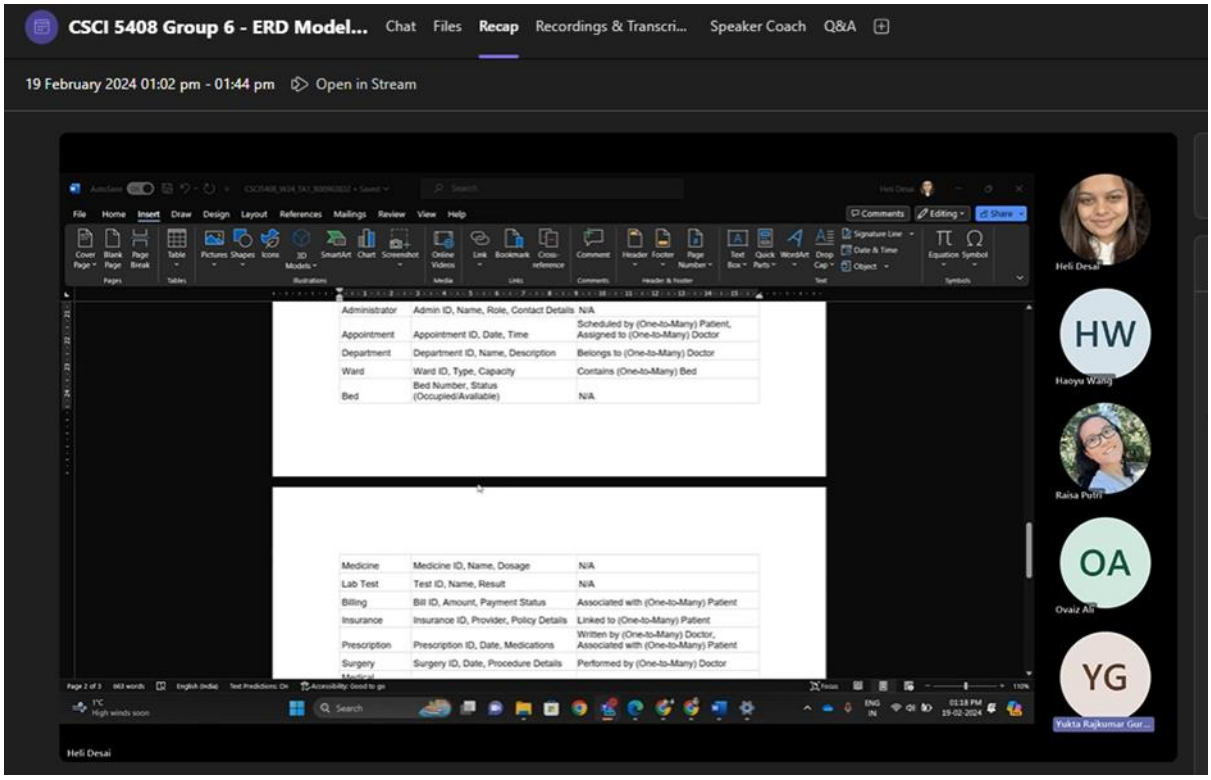

Figure 43: Screenshot for Meeting 1

Table 4: minutes of the meeting from meeting 2

Category	**Details**
Date/Time	Wednesday, February 21, 2024, 1:30 PM
Duration	2 hours, 5 minutes, 27 seconds
Location	Online (MS Teams)
Attendees	All team members
Agenda	• Consolidate ER Diagram (ERD). • Discuss and solve design issues. • Assign tasks for research and documentation.
Action Items	• Haoyu Wang and Raisa Putri to consolidate ERD. • Yukta Rajkumar Gurnani to conduct background research and summary. • Ovaiz Ali to address design issues. • Heli Desai to consolidate minutes into report.
Discussions	• Detailed review of entities, attributes, and relationships. • Collaborative effort on Draw.io for ERD adjustments. • Evaluation of each entity's necessity and associated relationships.
Outcomes	• Unified ERD with agreed-upon entities, attributes, and relationships. • Clear division of work for subsequent tasks. • Identified and planned resolution for design issues.

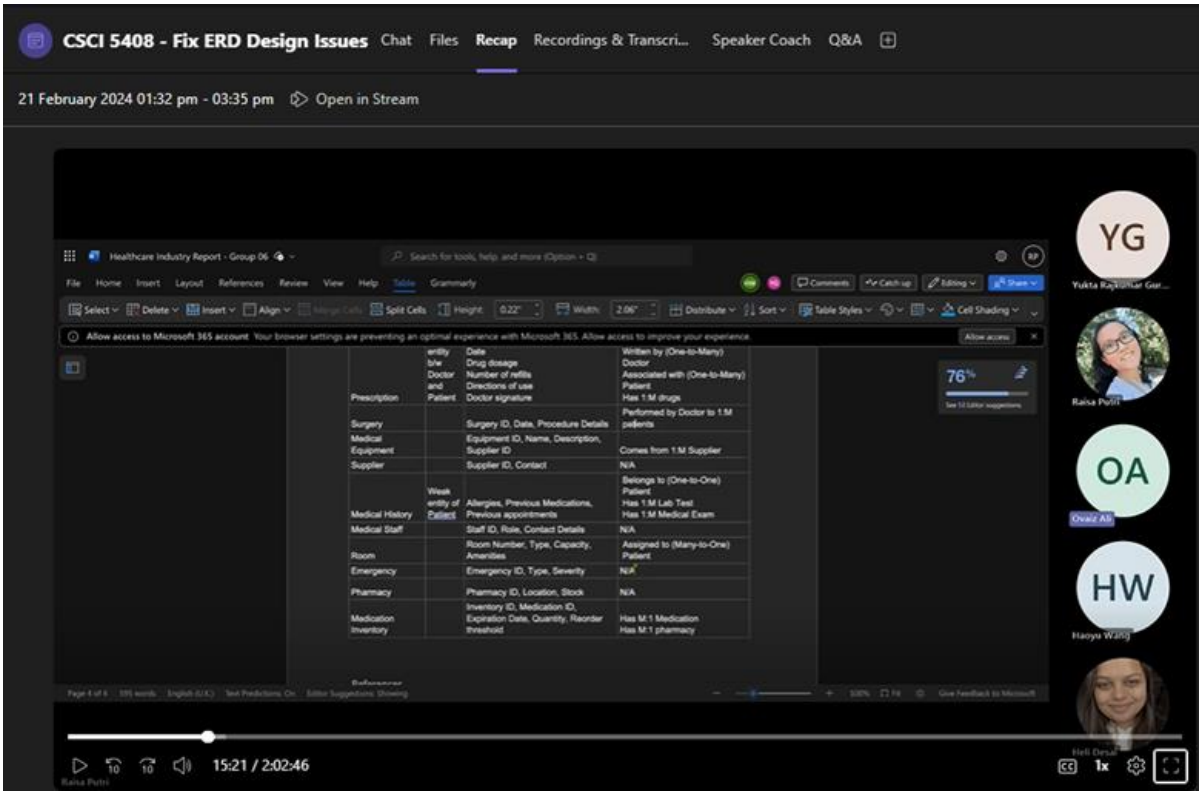

Figure 44: Screenshots of Meeting 2

Table 5: minutes of the meeting for meeting 3

Category	**Details**
Date/Time	Tuesday, February 27, 2024, 2:00 PM
Duration	1 hour
Location	Online (MS Teams)
Attendees	All team members
Agenda	• Finalize ERD. • Refine documentation for sprint 1 submission.
Discussions	• Detailed review of entities, attributes, and relationships after the design issues are resolved. • Collaborative effort on reviewing and refining the documentation
Outcomes	• Finalized submission

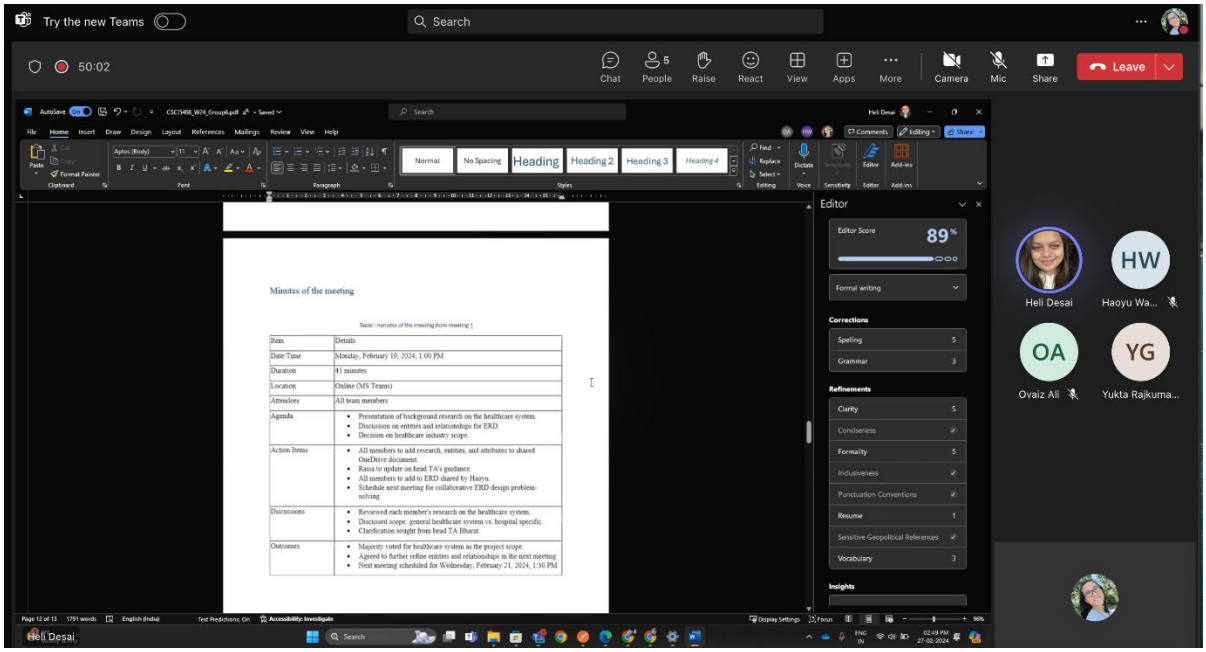

Figure 45: Screenshot of meeting 3

Table 6: Minutes of the meeting 4

Category	**Details**
Date	Thursday, March 7, 2024
Time	2:20 PM
Duration	40 minutes
Location	Online (MS Teams) & In-person at Goldberg
Attendees	Everyone in the team
Topic	Sprint 2 Planning
Action Items	Create Table Schemas (2NF & 3NF), dependency diagram: • Entities 1-5 -> Assigned to Raisa • Entities 6-10 -> Assigned to Haoyu • Entities 11-15 -> Assigned to Heli • Entities 16-20, 26 and 27 -> Assigned to Yukta • Entities 21-25 -> Assigned to Ovaiz
Discussions	Splitting tasks for Sprint 2. Each team member assigned entities for creating table schemas in 2NF and 3NF.
Next Steps	• Meet again on Sunday to finalize tables with foreign keys • Catch up with Head TA Bharat

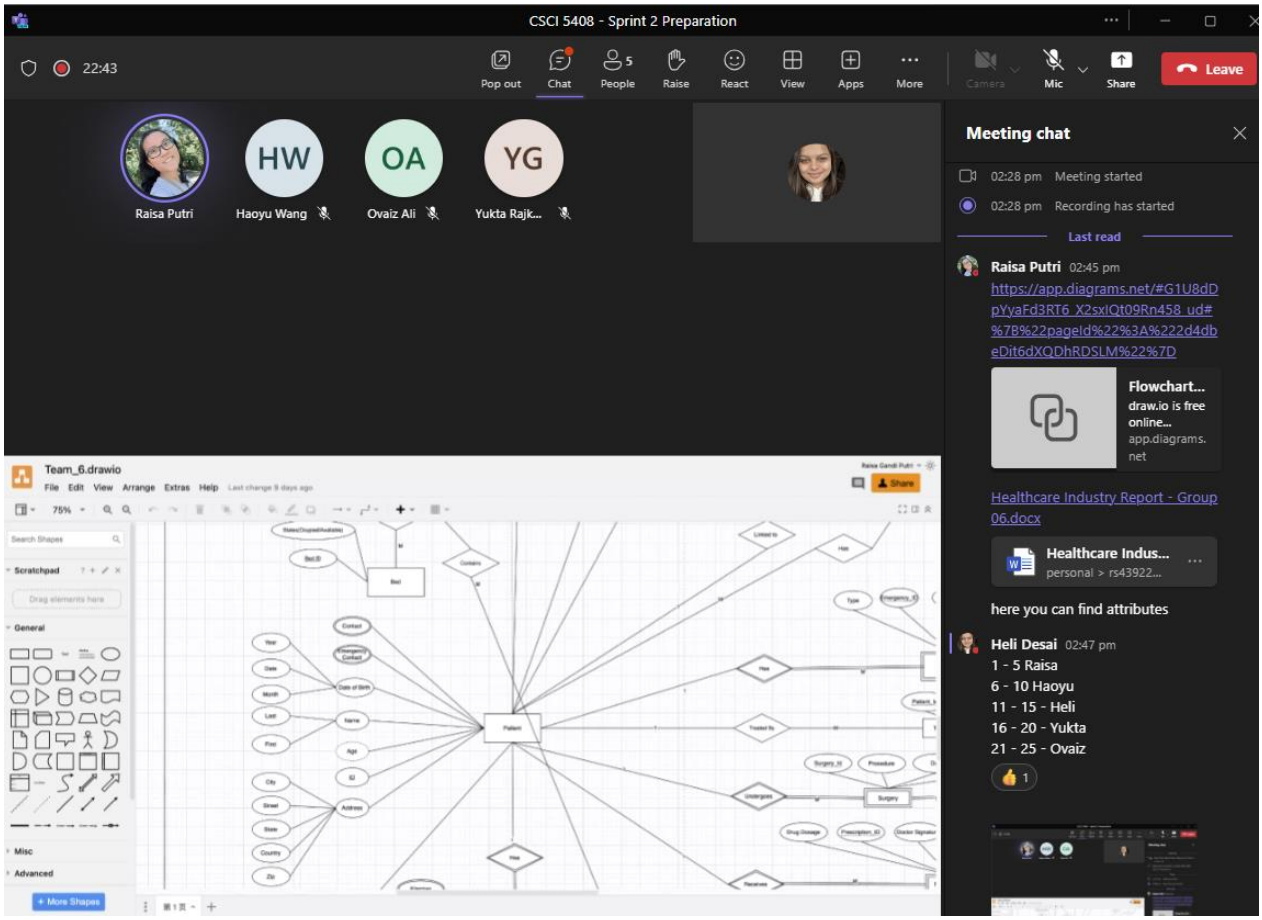

Figure 46: Screenshot of meeting 4

Table 7: Minutes of the Meeting 5

Category	**Details**
Meeting Date/Time	Sunday, March 10th, 2024 at 3:30 PM
Duration	1 hour
Location	Online (MS Teams)
Attendees	Ovaiz Ali, Raisa Putri, Haoyu Wang, Heli Desai, Yukta Rajkumar Gurnani
Topic	Finalizing Normalized Tables and Next Steps
Discussions	Finalized team's normalized tables.
Action Items	Write SQL DDL statements for each entity and Create entity dependency diagram for each table.
Next Steps	Meet on Monday after communications class to gather DDL statements and proceed with fragmentation

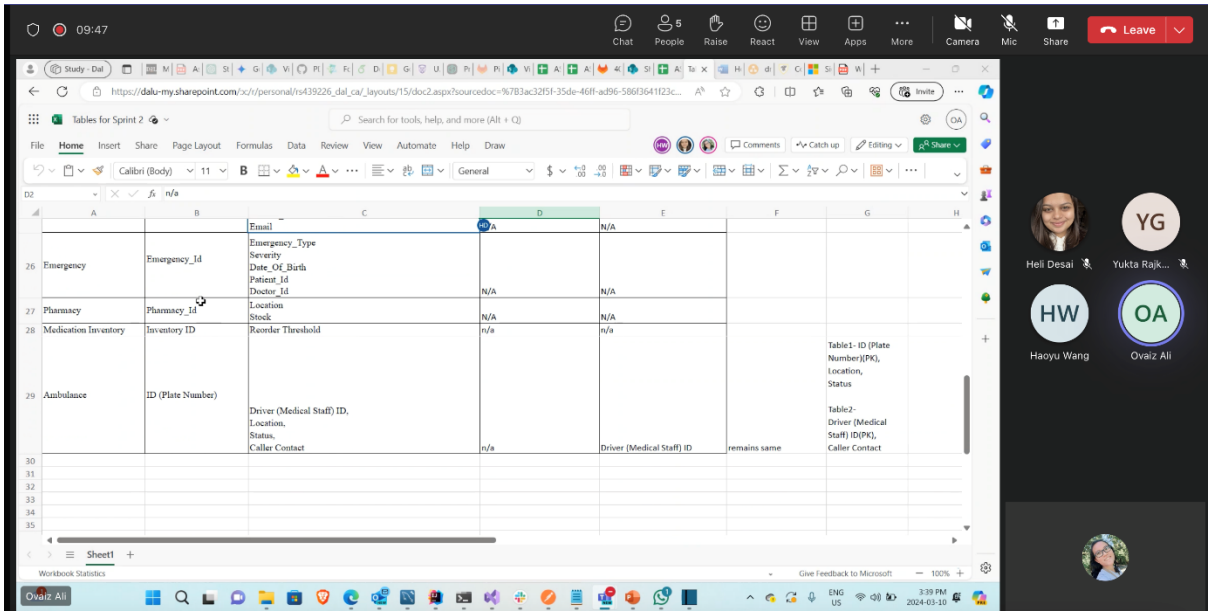

Figure 47: Screenshot for meeting 5

Table 8: Minutes of meeting 6

Category	**Details**
Meeting Date/Time	Wednesday, March 13, 2024 at 11:00 AM
Duration	1 hour 40 mins
Location	Online (MS Teams)
Attendees	Ovaiz Ali, Raisa Putri, Haoyu Wang, Heli Desai, Yukta Rajkumar Gurnani
Topic	Database Fragmentation & Meeting Preparation
Discussions	**Database Fragmentation** **- Strategies** Discussed VM (Vertical Fragmentation) and GDC (Global Data Distribution Committee) **- Entity Review** Reviewed each entity and determined fragmentation methods **- Concerns** Discussed how to present approach and concerns related to database fragmentation with Professor Dey
Action Items	Refine Fragmentation Strategies: Finalize fragmentation plans for all entities.
Next Steps	After the meeting with professor Finalize the approach

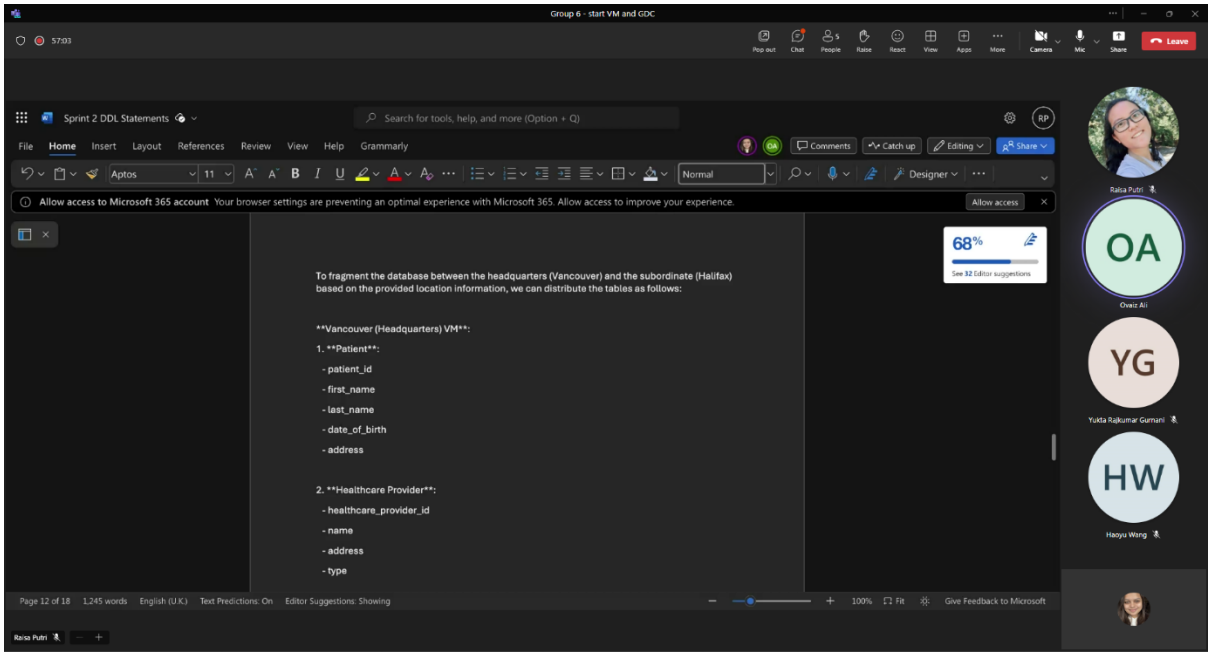

Figure 48: Screenshot of Meeting 6

Table 9: Minutes of meeting with Professor Saurabh Dey

Category	Details
Meeting Purpose	Discussion and clarification session on project progress and database implementation strategies.
Date & Time	Wednesday, March 13, 2024 at 3:00 PM
Duration	33 minutes 28 seconds
Attendees	Ovaiz Ali, Raisa Putri, Haoyu Wang, Heli Desai, Yukta Rajkumar Gurnani
Discussion Points	• Database Fragmentation and Cataloging: The team has successfully fragmented the databases, dedicating specific tables to the headquarters and others to a secondary VM, and is currently focusing on data cataloging. • Query Parsing and GDC Utilization: Clarified the need for query parsing, indicating that full query processing is not required. Instead, the Global Directory Catalog (GDC) should be used to determine where to execute queries, whether in VM1 or VM2. • Dynamic Connection Handling in Java: The Java program should not hardcode the connection but use variable connections to determine the target VM via GDC, enhancing the distributed database's functionality. • Distributed Database Implementation: Emphasis on the proper implementation of a distributed database system, highlighting that mere table distribution across instances does not constitute a distributed database.
Action Items	• Continue refining the Java program for dynamic database connection handling based on GDC. • Ensure that the GDC contains accurate and sufficient information to guide the distribution and access of data across VMs.
Next Steps	• Implement the feedback and clarifications provided by Saurabh Dey regarding the use of GDC and the structure of the distributed database.

	• Focus on adding the necessary information to the GDC to facilitate proper data access and management.

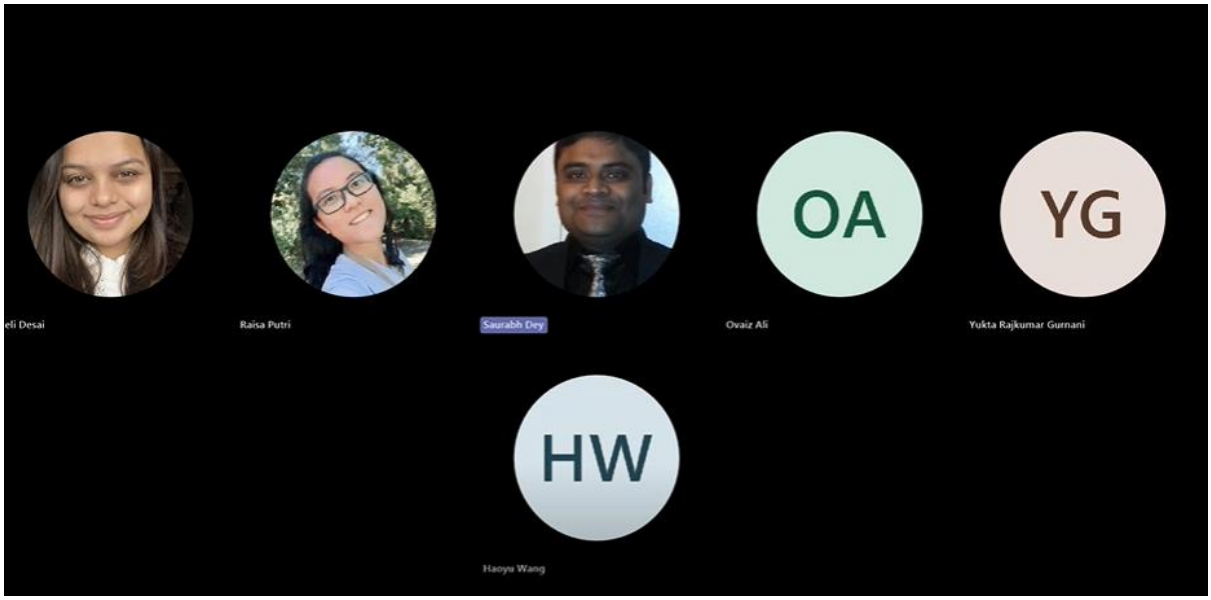

Figure 49: Screenshot of meeting with Professor Dey

Table 10: Minutes of meeting with Head TA

Category	**Details**
Topic	Project Discussion and Clarification Session
Date and Time	Wednesday, March 13, 2024 at 4:00 PM
Duration	7 mins
Location	Online (MS Teams)
Attendees	Ovaiz Ali, Bharat Shankaranarayanan, Yukta Rajkumar Gurnani, Raisa Putri, Haoyu Wang
Overview	Discussed progress on Excel file creation, task distribution, VM development, and Java program.

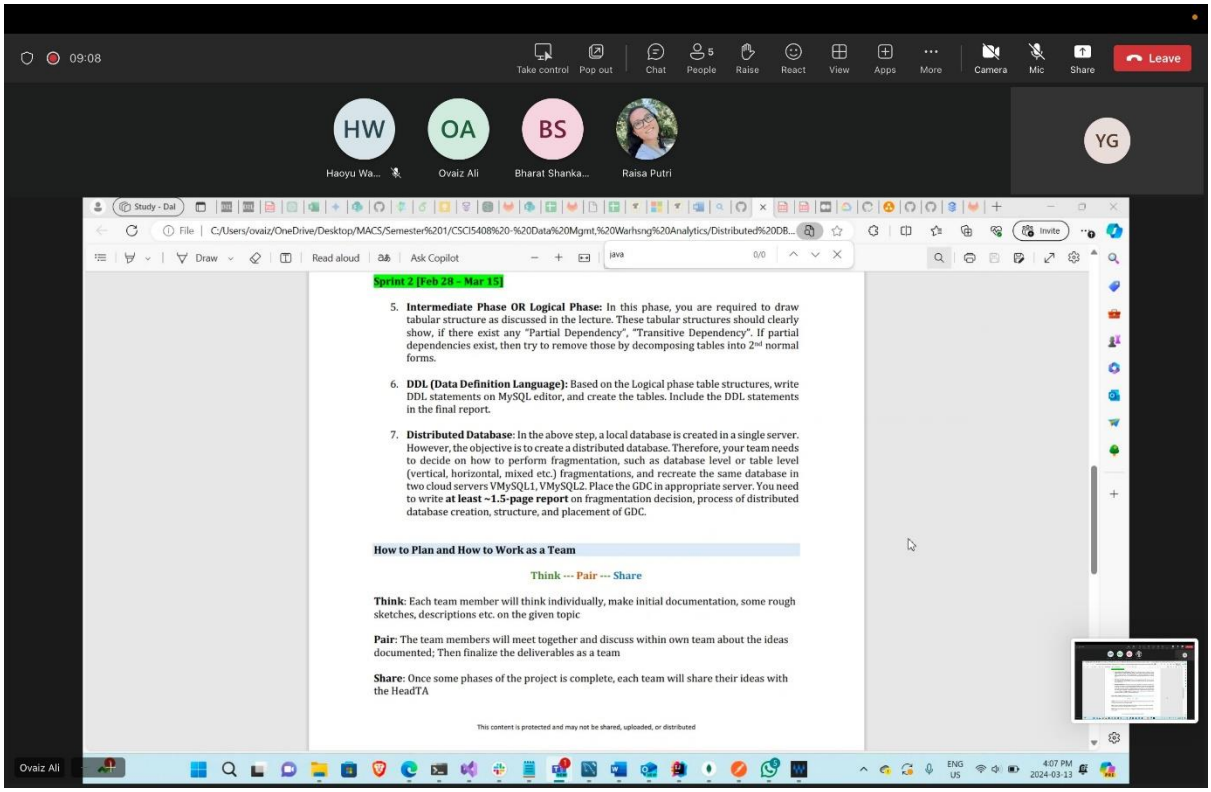

Figure 50: Screenshot for meeting with TA (a)

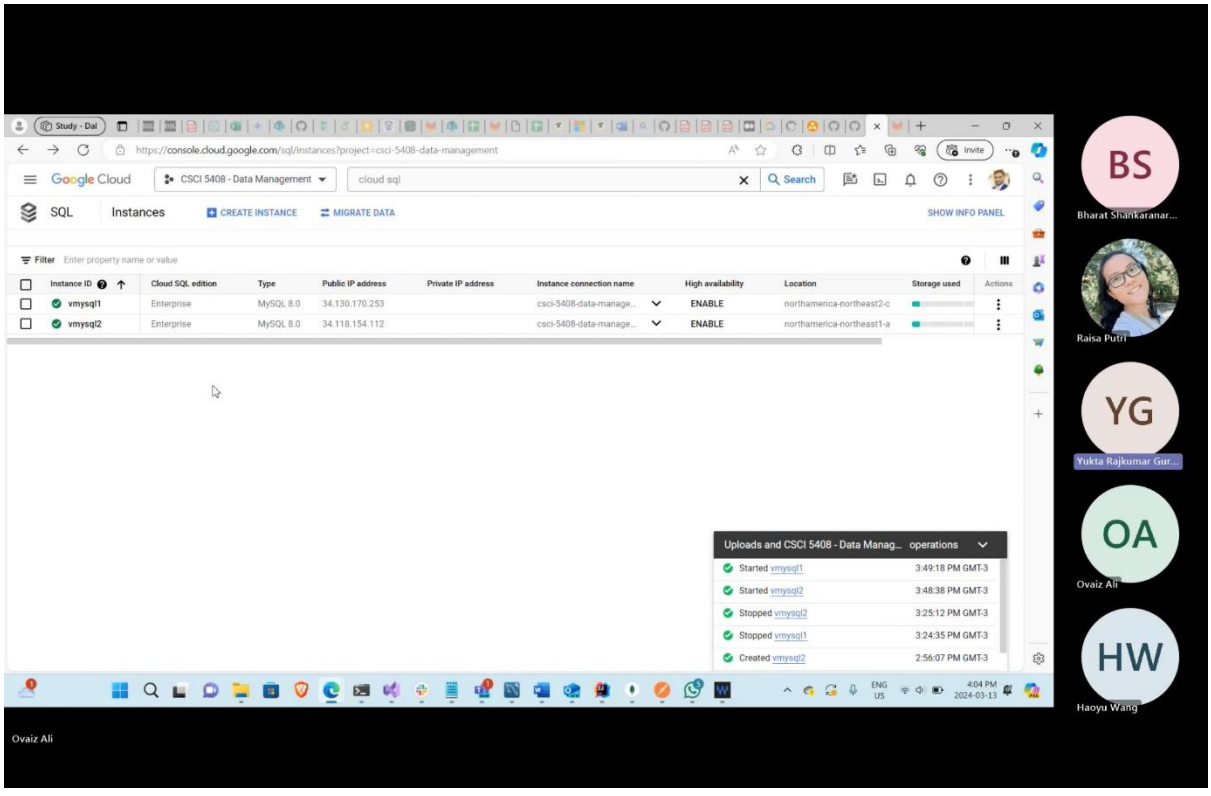

Figure 51: Screenshot for meeting with TA (b)

Table 11: Minutes of the meeting to finalise PPT and report

Category	Details
Topic	PPT creation for Meeting with TA and Report
Date and Time	Tuesday, March 20, 2024 at 10:00 AM
Duration	1 hr
Location	Online (MS Teams)
Attendees	Ovaiz Ali, Heli Desai, Yukta Rajkumar Gurnani, Raisa Putri, Haoyu Wang
Overview	prepare slides for TA Q&A Meeting
Action	Heli to Combine Sprint 1 and Sprint 2 Report

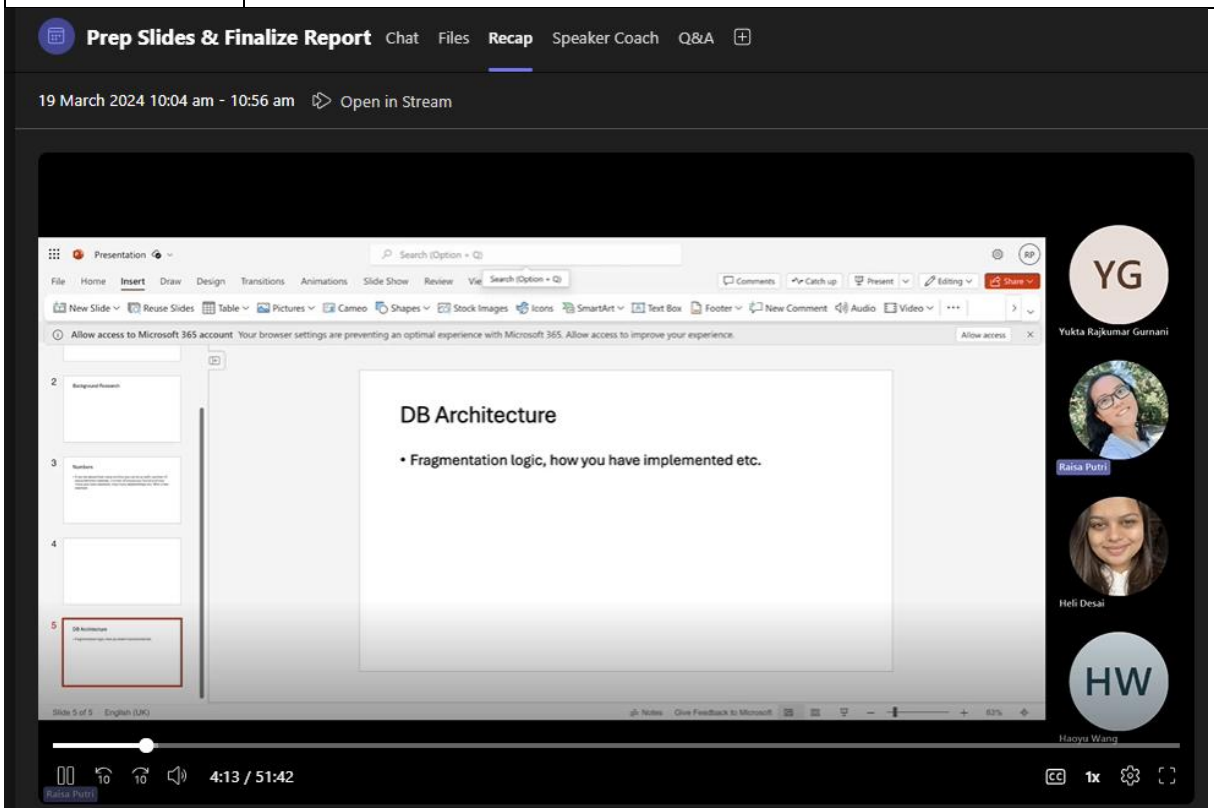

Figure 52: Meetings for Preparation of Slides and Finalising report

References

[1] “What is Redis replication with multi-master?” *Dragonfly* [Online]. Available: https://www.dragonflydb.io/faq/redis-replication-multi-master. [Accessed: Feb. 26, 2024].

[2] “Data Catalog overview,” *Google Cloud* [Online]. Available: https://cloud.google.com/data-catalog/docs/concepts/overview. [Accessed: Feb. 26, 2024].

[3] “Data Catalog documentation,” *Google Cloud* [Online]. Available: https://cloud.google.com/data-catalog/docs. [Accessed: Feb. 26, 2024].