Hospital Database

Prepared by
Moaz Maher Elsayed
Eslam Ashraf Negm
Fady Youssery Nageb
Yousef Mohamad Hamada
Amr Mohamad Ali
Abdallah Yahya Elsayed
Yahya Mahmoud Zakaria

GitHub
https://github.com/Moaz-Maher/Hospital-GUI

I. Business Requirements

A. Patient

- We have a set of patients each patient has unique id, first name, last name, gender, date of birth, address, unique email.
- Each patient is can be examined by multiple doctors and we store examine date and the diagnosis.
- Each patient can perform any number of operations.
- each patient can stay in a room and we store entry and leaving date.

B. Doctor

- Each doctor has unique id, first name, last name, degree which is one of the following (Bachelor – Master - Doctoral), graduation year, and specialization that he works in hospital.
- He can work only in one specialization.
- Doctor may manage one specialization (specialization match to his specialization).
- Doctor may supervise many nurses
- Doctor can perform any number of operations (operation is performed on one patient but can be performed with many doctors and nurses)
- Doctor also has a set of appointments marked with day (Sunday-Monday.. etc) and shift number from 1 to 6 and the clinic he will work at (must match his specialization) this shift. (doctor may work with other doctors at the same clinic at the same time)
- Doctor can examine any number of patients.

C. Specialization

- There are set of specialization in our hospital and it has unique name and the date that we launch this specialization in our hospital.
- Each specialization may have many clinics associated with it, but each clinic must belongs to one specialization.
- Specialization may be managed by one doctor (of the same specialization).
- Specialization may have many number of doctors work in it.

D. Clinic

- We have clinic unique id, name and floor number (0 to 10).
- Each clinic must be associated with one specialization.

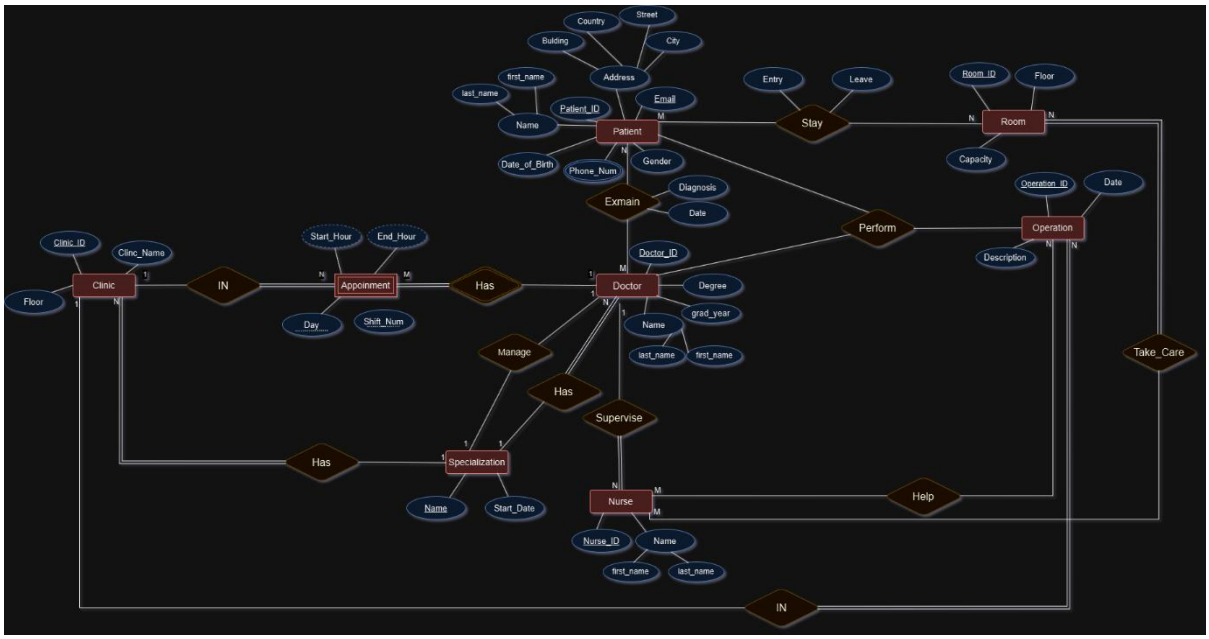
E. Nurse

- We have set of nurses that has first name, last name, unique id.
- Each nurse may participate in operations many times.
- Each nurse may take care of many rooms.

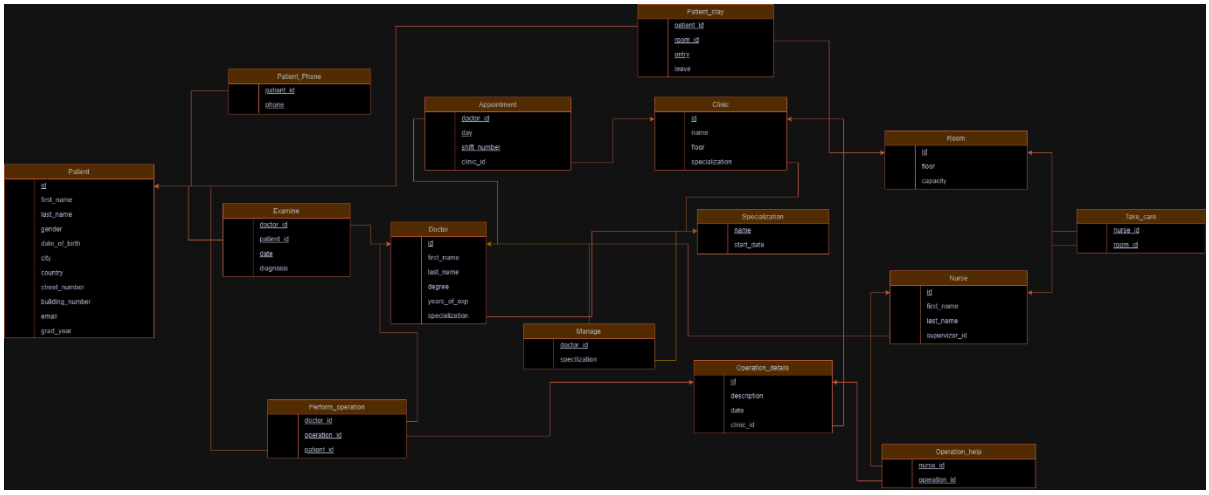
F. Operations

- Operation can be performed by more than one doctor one and one patient only.
- We need to store operation description and date of the operation and clinic that the operation was done in.

II. ERD



III. Mapping ERD to Logical Schema



Constraints:

A. Primary Key Constraints

- **Doctor**: id is the primary key.
- **Patient**: id is the primary key.
- **Examine**: Composite primary key on doctor_id, patient_id, date.
- **Patient_Phone**: Composite primary key on patient_id, phone.
- **Specialization**: name is the primary key.
- **Appointment**: Composite primary key on doctor_id, day, shift_number.
- **Clinic**: id is the primary key.
- **Operation_details**: id is the primary key.
- **Perform_operation**: Composite primary key on doctor_id, operation_id, patient_id.
- **Nurse**: id is the primary key.
- **Operation_help**: Composite primary key on nurse_id, operation_id.
- **Room**: id is the primary key.
- **Patient_stay**: Composite primary key on patient_id, room_id, entry.
- **Take_care**: Composite primary key on nurse_id, room_id.
- **Manage**: doctor_id is primary key

B. Foreign Key Constraints

- **Doctor**:
  - specialization references Specialization(name).
- **Examine**:
  - doctor_id references Doctor(id).
  - patient_id references Patient(id).
- **Patient_Phone**:
  - patient_id references Patient(id).

- **Specialization**:
  - manager_id references Doctor(id).
- **Appointment**:
  - doctor_id references Doctor(id) with ON DELETE CASCADE.
  - clinic_id references Clinic(id).
- **Clinic**:
  - specialization references Specialization(name) with ON UPDATE CASCADE and ON DELETE CASCADE.
- **Operation_details**:
  - clinic_id references Clinic(id).
- **Perform_operation**:
  - doctor_id references Doctor(id).
  - operation_id references Operation_details(id).
  - patient_id references Patient(id).
- **Nurse**:
  - supervizor_id references Doctor(id).
- **Operation_help**:
  - nurse_id references Nurse(id).
  - operation_id references Operation_details(id).
- **Patient_stay**:
  - patient_id references Patient(id).
  - room_id references Room(id).
- **Take_care**:
  - nurse_id references Nurse(id) with ON DELETE CASCADE.
  - room_id references Room(id) with ON DELETE CASCADE.
- **Manage**:
  - Doctor_id references Doctor(id).

C. Check Constraints

1. **Doctor:**
   degree which is one of the following (Bachelor – Master - Doctoral)
   grad_year between 1000 and 3000

- **Patient**:
  - gender must be either 'Male' or 'Female'.
- **Appointment**:
  - shift_number must be between 1 and 6.
  - day must be one of 'Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', or 'Saturday'.

- **Clinic**:
  - floor must be between 0 and 10.
- **Room**:
  - floor must be between 0 and 10.

D. Unique Constraints

- **Patient**:
  - email is unique.
- **Manage**:
  - specialization is unique.

E. Triggers

- **trg_EnsureDoctorClinicSpecialization**:
  - Ensures that a doctor is assigned to a clinic that matches their specialization. This is an AFTER INSERT, UPDATE trigger on the Appointment table.
- **trg_CheckRoomCapacity**:
  - Ensures that no patient is assigned to a room that has reached its maximum capacity. This is an AFTER INSERT, UPDATE trigger on the Patient_stay table.
- **trg_EnsureDoctorSpecializationMatch**:
  - To enforce the business rule that a doctor can only manage a specialization matching their own specialization.

F. Identity Columns

- **Clinic**:
  - id is an identity column (auto-incremented).
- **Operation_details**:
  - id is an identity column (auto-incremented).
- **Room**:
  - id is an identity column (auto-incremented).

Default:

- Room: capacity default is 0

IV. Normalization

Lets check functional dependencies for each table:

Doctor(id, first_name, last_name, degree, grad_year, specialization)

FDs:
id → first_name, last_name, degree, grad_year, specialization

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since id is the sole candidate key, and all non-prime attributes are fully dependent on id.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF since id is the sole determinant for all attributes.

Patient(id, first_name, last_name, gender, date_of_birth, city, country, street_number, building_number, email)

FDs:
id → first_name, last_name, gender, date_of_birth, city, country, street_number, building_number, email

email → id (email is unique)

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since id is the sole candidate key, and all non-prime attributes are fully dependent on id.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Examine(doctor_id, patient_id, date, diagnosis)

FDs:
(doctor_id, patient_id, date) → diagnosis

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since the composite key fully determines diagnosis.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Patient_Phone(patient_id, phone)

FDs:
(patient_id, phone) → No additional FDs (Composite primary key)

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since there are no partial dependencies.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Specialization(name, start_date)

FDs:
name → start_date

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since name is the sole candidate key.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Appointment(doctor_id, day, shift_number, clinic_id)

FDs:
(doctor_id, day, shift_number) → clinic_id

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since the composite key fully determines clinic_id.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Clinic(id, name, floor, specialization)

FDs:
id → name, floor, specialization

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since id is the sole candidate key.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Operation_details(id, description, date, clinic_id)

FDs:

id → description, date, clinic_id

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since id is the sole candidate key.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Perform_operation(doctor_id, operation_id, patient_id)

FDs:
(doctor_id, operation_id, patient_id) → No additional FDs

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since there are no partial dependencies.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Nurse(id, first_name, last_name, supervizor_id)

FDs:
id → first_name, last_name, supervizor_id

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since id is the sole candidate key.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Room(id, floor, capacity)

FDs:
id → floor, capacity

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since id is the sole candidate key.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Patient_stay(patient_id, room_id, entry, leave)

FDs:
(patient_id, room_id, entry) → leave

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since the composite key fully determines leave.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

Take_care(nurse_id, room_id)

FDs:
(nurse_id, room_id) → No additional FDs

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since there are no partial dependencies.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

We can see that database are normalized to BCNF.

Manage(doctor_id,specialization)

FDs:
Doctor_id → specialization
specialization → doctor_id

1NF: Satisfies 1NF as all attributes are atomic.
2NF: Satisfies 2NF since there are no partial dependencies.
3NF: Satisfies 3NF since there are no transitive dependencies.
BCNF: Satisfies BCNF.

We can see that database are normalized to BCNF.

GitHub
https://github.com/Moaz-Maher/Hospital-GUI

Thanks 😊