DBMS Minor Project

Submitted by:

Aaryan Vinod Kumar: 2024UCD2160
Dhriti: 2024UCD2171
Sanya Khajuria: 2024UCD3035

Index:

Problem Statement:

In most clinics today, the process of booking a doctor's appointment is still handled manually through a receptionist or clinic staff. Patients are required to call or physically visit the clinic during working hours to schedule their appointments. This system is not only time-consuming but also inconvenient, as patients cannot book appointments after hours, leading to delays in medical consultation.

Additionally, manual handling of records increases the chances of errors, double bookings, and difficulty in tracking patient history. To overcome these limitations, there is a need for an online appointment booking system where patients can easily schedule appointments with doctors at any time, including nights and weekends.

Doctors should be able to verify and confirm these appointments through the system, ensuring better time management and reducing dependency on manual staff. Such a solution will improve accessibility, reduce workload for clinic staff, and enhance overall patient experience.

DBMS Project: ER Model

(Clinic Management System)

Entities

Staff (Superclass)
- staff_id (Primary Key)
- name (VARCHAR)
- email (VARCHAR, UNIQUE)
- phone (NUMBER)
- address (VARCHAR)
- staff_type (ENUM: Doctor / Nurse)

Doctor (Subclass of Staff)
- doctor_id (Primary Key, also Foreign Key → staff)
- specialization (VARCHAR)
- pstart (DATETIME)

Nurse (Subclass of staff)
- nurse_id (Primary Key, also Foreign Key → staff)
- department (VARCHAR)
- shift_type (ENUM: Morning / Evening / Night)

Patient
- patient_id (Primary Key)
- name (VARCHAR)
- email (VARCHAR, UNIQUE)
- phone (VARCHAR)
- date_of_birth (DATE)
- gender (CHAR)
- address (VARCHAR)

Clinic
- clinic_id (Primary Key)
- name (VARCHAR)
- address (VARCHAR)
- contact_number (multivalued)

Clinic_Contact
- clinic_id (foreign key)
- contact_number (Discriminater/Partial Key)

Appointment
- appointment_id (Primary Key)
- patient_id (Foreign Key → Patient)
- doctor_id (Foreign Key → Doctor)
- clinic_id (Foreign Key → Clinic)
- appointment_datetime (DATETIME)
- status (ENUM: Booked / Completed / Cancelled)
- reason (VARCHAR)
- priority (ENUM: Low / Medium / High)

Payment
- payment_id (Primary Key)
- appointment_id (Foreign Key → Appointment, UNIQUE)
- amount (DECIMAL(10,2))
- payment_method (ENUM: Cash / Card / UPI / Insurance)
- payment_status (ENUM: Pending / Paid / Failed)
- payment_date (DATETIME)

Prescription
- prescription_id (Primary Key)
- appointment_id (Foreign Key → Appointment)
- diagnosis (TEXT)
- medicines (TEXT)
- advice (TEXT)
- prescription_date (DATETIME)

PrescriptionMedicine

- medicine_name(VARCHAR)
- dosage(VARCHAR)
- duration(VARCHAR)

Relationships

Staff→ Doctor / Nurse (Generalization/Specialization)

- staff is a superclass
- Doctor and Nurse are subclasses (specializations)
- Type: ISA (is-a) relationship
- Disjoint constraint: A person can be either Doctor OR Nurse (not both)
- Total participation: Every staff must be either a Doctor or Nurse

Patient → Appointment

- One Patient can book many Appointments
- Each Appointment belongs to exactly one Patient
- Cardinality: 1 : M

Doctor → Appointment

- One Doctor can handle many Appointments
- Each Appointment is handled by exactly one Doctor
- Cardinality: 1 : M

Clinic → Appointment

- One Clinic can host many Appointments
- Each Appointment takes place in one Clinic
- Cardinality: 1 : M

Appointment → Payment

- One Appointment has exactly one Payment record
- Each Payment belongs to exactly one Appointment
- Cardinality: 1 : 1

Appointment → Prescription

- One Appointment can have many Prescriptions
- Each Prescription belongs to exactly one Appointment
- Cardinality: 1 : M

Nurse → Doctor (Assists)

- One Nurse can assist many Doctors
- One Doctor can be assisted by many Nurses
- Cardinality: M : N
- Implementation: NurseDoctor junction table (nurse_id FK, doctor_id FK)

Prescription → PrescriptionMedicine

- One Prescription can include many medicines.
- Each PrescriptionMedicine belongs to exactly one Prescription.
- Cardinality: 1:M

Generalization/Specialization Design

Why Use Person Superclass?
- Common attributes between Doctor and Nurse reduce redundancy and support scalability.

Implementation Approaches

Approach 2 (Table Per Type) recommended: maintains normalization and avoids NULLs.

Key Design Decisions

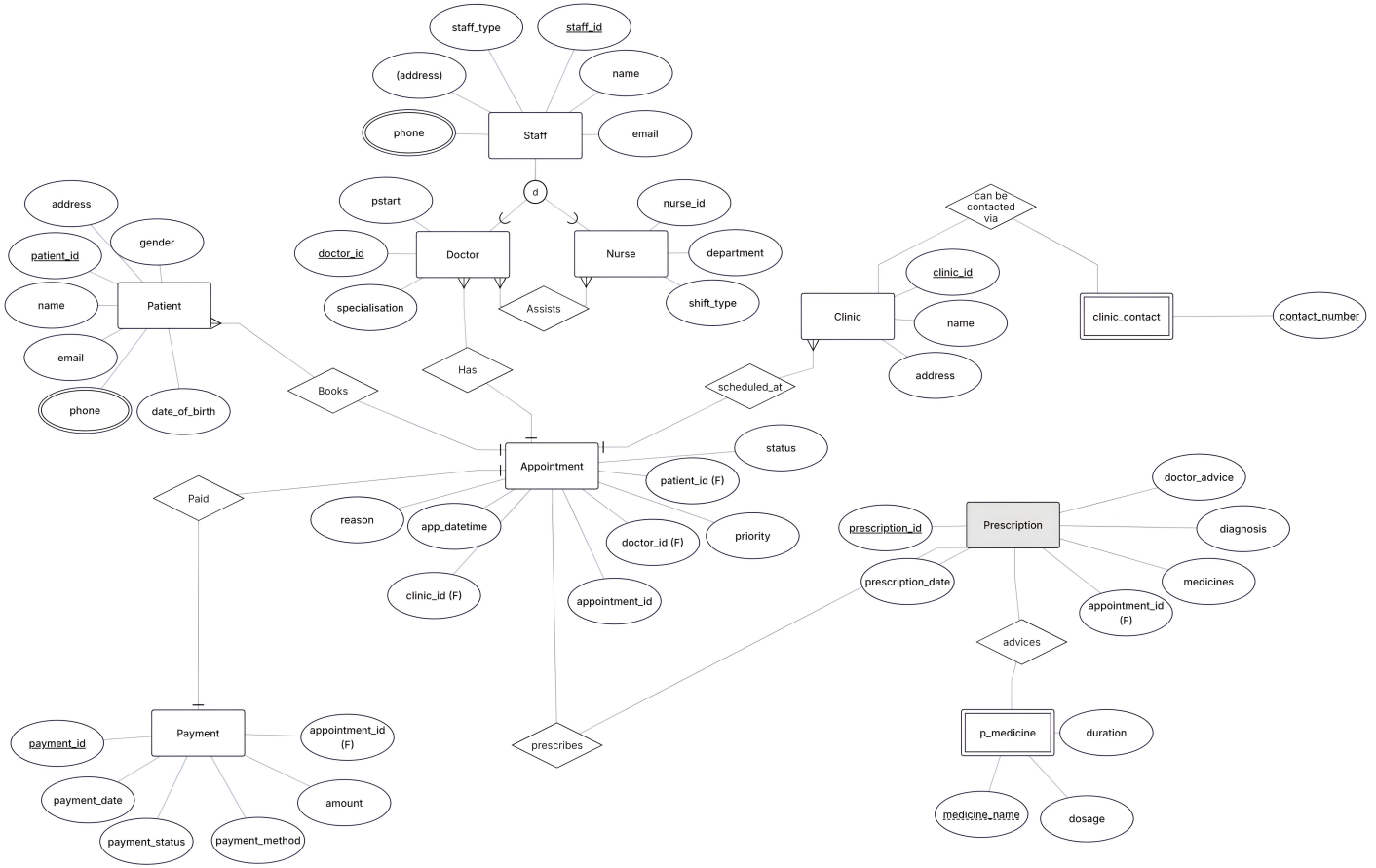
Data types, constraints, and normalization ensure integrity and efficiency.

ER Diagram Notation

ISA relationship denoted by disjoint (d) symbol. Rectangles for entities, diamonds for relationships, ellipse for attributes .

Additional Notes

Prescription only has appointment_id FK to avoid redundancy. Maintains 3NF normalization.


staff_type
staff_id
(address)
name
phone
Staff
email
d
pstart
nurse_id
doctor_id
Doctor
Nurse
department
specialisation
Assists
shift_type
address
gender
patient_id
name
Patient
email
phone
date_of_birth
Books
Has
can be contacted via
clinic_id
Clinic
name
address
clinic_contact
contact_number
scheduled_at
Appointment
status
patient_id (F)
Paid
reason
app_datetime
doctor_id (F)
priority
clinic_id (F)
appointment_id
prescription_id
Prescription
doctor_advice
diagnosis
prescription_date
medicines
appointment_id (F)
advices
Payment
payment_id
appointment_id (F)
payment_date
amount
payment_status
payment_method
prescribes
p_medicine
duration
medicine_name
dosage

DBMS Project: Relational Model

(Clinic Management System)

Problem Statement:

In most clinics today, the process of booking a doctor's appointment is still handled manually through a receptionist or clinic staff. Patients are required to call or physically visit the clinic during working hours to schedule their appointments. This system is not only time-consuming but also inconvenient, as patients cannot book appointments after hours, leading to delays in medical consultation. Additionally, manual handling of records increases the chances of errors, double bookings, and difficulty in tracking patient history.

To overcome these limitations, there is a need for an online appointment booking system where patients can easily schedule appointments with doctors at any time, including nights and weekends. Doctors should be able to verify and confirm these appointments through the system, ensuring better time management and reducing dependency on manual staff. Such a solution will improve accessibility, reduce workload for clinic staff, and enhance overall patient experience.

Relations (Tables)

1. Staff

Staff(staff_id, name, email, phone, address, staff_type)

- **Primary Key:** staff_id
- **Unique:** email
- **Constraints:**
    - staff_type ∈ {'Doctor', 'Nurse'}

2. Doctor

Doctor(doctor_id, specialization, pstart)

- **Primary Key:** doctor_id
- **Foreign Key:** doctor_id → Staff(staff_id) ON DELETE CASCADE
- **Note:** Inherits attributes from Staff

3. Nurse

Nurse(nurse_id, department, shift_type)

- **Primary Key:** nurse_id
- **Foreign Key:** nurse_id → Staff(staff_id) ON DELETE CASCADE
- **Constraints:**
    - shift_type ∈ {'Morning', 'Evening', 'Night'}
- **Note:** Inherits attributes from Staff

4. Patient

Patient(patient_id, name, email, phone, date_of_birth, gender, address)

- **Primary Key:** patient_id
- **Unique:** email

5. Clinic

Clinic(clinic_id, name, address)

- **Primary Key:** clinic_id

6. Clinic_Contact

Clinic_Contact(clinic_id, contact_number)

- **Primary Key:** (clinic_id, contact_number)
- **Foreign Key:** clinic_id → Clinic(clinic_id) ON DELETE CASCADE
- **Note:** Handles multivalued contact_number attribute

7. Appointment

Appointment(appointment_id, patient_id, doctor_id, clinic_id,

appointment_datetime, status, reason, priority)

- **Primary Key:** appointment_id
- **Foreign Keys:**
  - patient_id → Patient(patient_id)
  - doctor_id → Doctor(doctor_id)
  - clinic_id → Clinic(clinic_id)
- **Constraints:**
  - status ∈ {'Booked', 'Completed', 'Cancelled'}
  - priority ∈ {'Low', 'Medium', 'High'}

8. Payment

Payment(payment_id, appointment_id, amount, payment_method,

payment_status, payment_date)

- **Primary Key:** payment_id
- **Foreign Key:** appointment_id → Appointment(appointment_id)
- **Unique:** appointment_id (enforces 1:1 relationship)
- **Constraints:**
  - payment_method ∈ {'Cash', 'Card', 'UPI', 'Insurance'}
  - payment_status ∈ {'Pending', 'Paid', 'Failed'}

9. Prescription

Prescription(prescription_id, appointment_id, diagnosis, medicines,

advice, prescription_date)

- **Primary Key:** prescription_id
- **Foreign Key:** appointment_id → Appointment(appointment_id)

10. PrescriptionMedicine

PrescriptionMedicine(prescription_id, medicine_name, dosage, duration)

- **Primary Key:** (prescription_id, medicine_name)
- **Foreign Key:** prescription_id → Prescription(prescription_id) ON DELETE CASCADE
- **Note:** Weak entity dependent on Prescription

11. NurseDoctor (Junction Table)

NurseDoctor(nurse_id, doctor_id)

- **Primary Key:** (nurse_id, doctor_id)
- **Foreign Keys:**
  - nurse_id → Nurse(nurse_id) ON DELETE CASCADE
  - doctor_id → Doctor(doctor_id) ON DELETE CASCADE
- **Note:** Implements M:N "Assists" relationship

Relational Schema Summary

Total Relations: 11 tables

Key Implementation Notes:

1. **Generalization (Staff → Doctor/Nurse):** Implemented using "Table Per Type" approach
    - Staff table holds common attributes
    - Doctor and Nurse tables hold specialized attributes
    - Ensures disjoint constraint (a staff member is either Doctor OR Nurse)
2. **Multivalued Attribute:** Clinic contact numbers handled via separate Clinic_Contact table
3. **Weak Entity:** PrescriptionMedicine depends on Prescription
4. **1:1 Relationship:** Appointment-Payment enforced via UNIQUE constraint on appointment_id in Payment
5. **M:N Relationship:** Nurse-Doctor "Assists" relationship via NurseDoctor junction table
6. **Normalization:** All tables are in 3NF (Third Normal Form)

Functional Dependencies

Staff

- staff_id → name, email, phone, address, staff_type
- email → staff_id

Doctor

- doctor_id → specialization, pstart

Nurse

- nurse_id → department, shift_type

Patient

- patient_id → name, email, phone, date_of_birth, gender, address
- email → patient_id

Appointment

- appointment_id → patient_id, doctor_id, clinic_id, appointment_datetime, status, reason, priority

Payment

- payment_id → appointment_id, amount, payment_method, payment_status, payment_date
- appointment_id → payment_id (due to 1:1 relationship)

Prescription

- prescription_id → appointment_id, diagnosis, medicines, advice, prescription_date