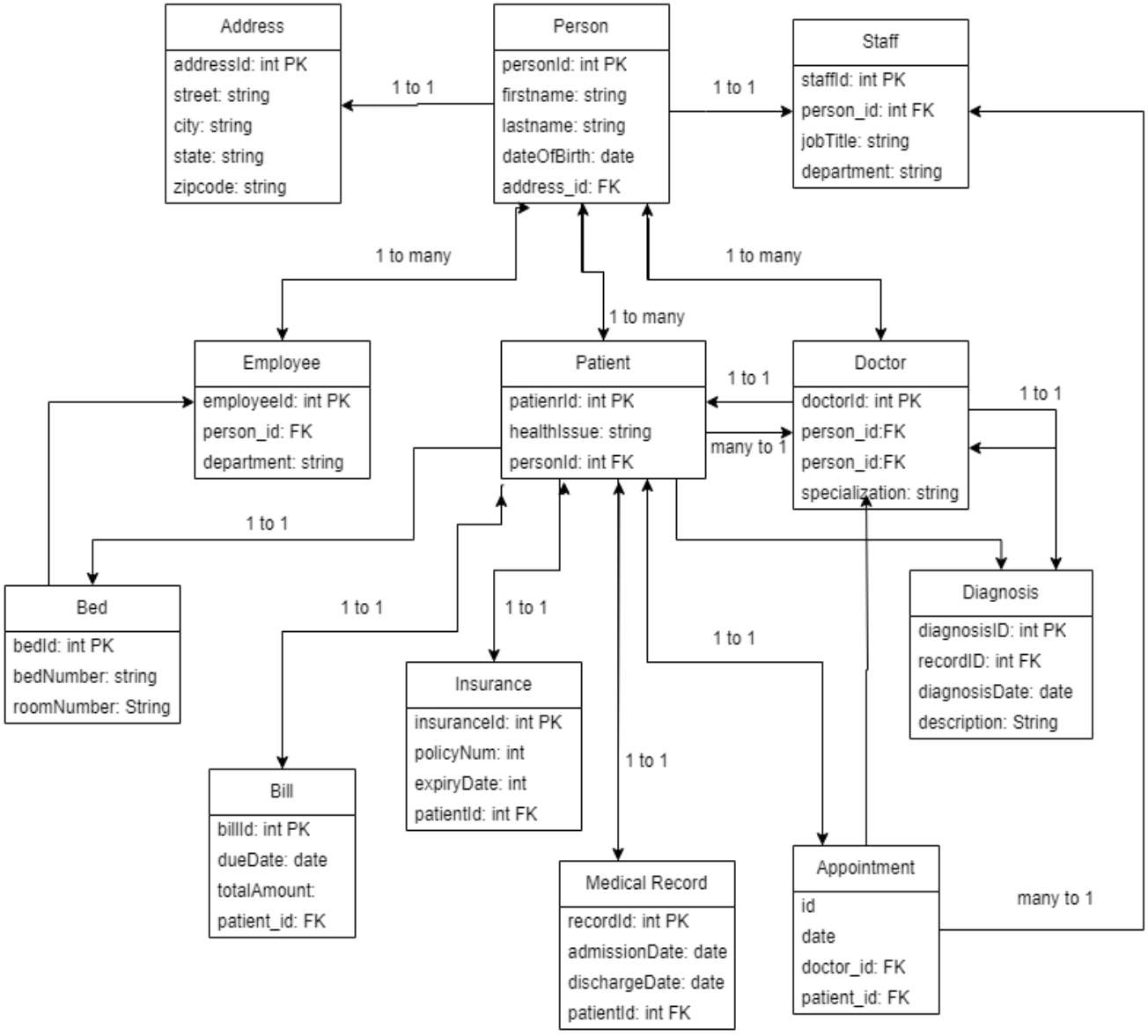

Views
PatientMedicalRecord
Doctor
AppointmentSlots

Triggers
CheckInsurance
InsuranceExpiryDate

Stored Procedures
RegisterPatient
RegisterPerson
RegisterDoctor

Index
patientId
doctorId

- The Normalization form being used in the HMS by me is 3NF. I considered using 3NF because it mostly eliminates the duplicacy of data in database and as the data replication is reduced, searching becomes much more faster and plus point is that comparatively it requires much lesser storage space.

1) Queries to register new user roles:

- INSERT INTO Address (street, city, state, zipCode) VALUES ('123 Elm St', 'Anytown', 'Anystate', '54321');
- INSERT INTO Person (firstName, lastName, dateOfBirth, addressId) VALUES ('John', 'Doe', '1990-01-01', @addressId);
- INSERT INTO Patient (patientId, healthIssue, personId) VALUES (@personId, 'Fever', @personId);
- INSERT INTO MedicalRecord (recordId, patientId, admissionDate, dischargeDate) VALUES (1, @personId, '2022-02-25', NULL);
- INSERT INTO Diagnosis (diagnosisId, recordId, diagnosisDate, description) VALUES (1, 1, '2022-02-25', 'Fever');
- INSERT INTO Staff (staffId, jobTitle, department, personId) VALUES (1, 'Nurse', 'Emergency', 1);
- INSERT INTO Doctor (doctorId, specialization, staffId) VALUES (1, 'General Physician', 1);
- INSERT INTO Appointment (appointmentId, patientId, doctorId, appointmentDate) VALUES (1, @personId, 1, '2022-02-25');
- INSERT INTO Bill (billId, totalAmount, patientId) VALUES (1, 100.00, @personId);

2) Query to add to the list of diagnosis of the patient tagged by date:

- INSERT INTO Diagnosis (diagnosisId, recordId, diagnosisDate, description) VALUES (2, 1, '2022-02-28', 'Diabetes');

3) Query to fetch required details of a particular patient:

- SELECT * FROM Person JOIN Patient ON Person.personId = Patient.personId
  JOIN MedicalRecord ON Patient.patientId = MedicalRecord.patientId
  WHERE Person.firstName = 'John' AND Person.lastName = 'Doe';

4) Query to prepare bill for the patient at the end of checkout:

- INSERT INTO Bill (billId, totalAmount, patientId) VALUES (2, 100.00, 1);

5) Query to fetch and show data from various related tables:

- SELECT * FROM Person JOIN Patient ON Person.personId = Patient.personId
  JOIN Appointment ON Patient.patientId = Appointment.patientId
  JOIN Doctor ON Appointment.doctorId = Doctor.doctorId
  WHERE Person.firstName = 'John' AND Person.lastName = 'Doe';

6) Read operations using views:

- CREATE VIEW PatientDetails AS
  SELECT p.patientId, p.healthIssue, p.personId,
  per.firstName, per.lastName, per.dateOfBirth, per.addressId
  FROM Patient p JOIN Person per ON p.personId = per.personId;

7) Read operations using indexing wherever required:

- CREATE INDEX idx_person_id ON Person (personId);

8) Bill generation using stored procedures:

- DELIMITER //

  CREATE PROCEDURE CalculateBill(IN patientId INT, OUT totalAmount DECIMAL(10,2))

  BEGIN

  SELECT SUM(totalAmount) INTO totalAmount

  FROM Bill WHERE patientId = patientId;

  END //

  DELIMITER ;

9) Triggers to indicate when patients medical insurance limit has expired:

- DELIMITER //

  CREATE TRIGGER CheckInsuranceLimit

  BEFORE INSERT ON Insurance

  FOR EACH ROW

  BEGIN

  IF NEW.expiry < CURDATE() THEN

  SIGNAL SQLSTATE '45000'

  SET MESSAGE_TEXT = 'Insurance limit has expired';

  END IF;

  END//

  DELIMITER ;