CSC 540 - Database Management Systems

Project Report 1

WolfHospital Management System

Team Members:

Mohd Sharique Khan (mkhan8@ncsu.edu)

Richa Dua (rdua2@ncsu.edu)

Siddu Madhure Jayanna (smadhur@ncsu.edu)

Viviniya Alexis Lawrence (palexis@ncsu.edu)

1. Assumptions and Problem Statement

Assumptions:

1. Registration Staff would be the admin of the system.
2. A nurse is responsible for all the beds in a ward.
3. While accepting the payment for the patient's treatment, only one transaction method is used i.e. the payment can't be split in multiple transactions. If the insurance company pays for the treatment, they pay in full.
4. Registration Fee is dependent on factors like arrival time of the patient into the hospital, emergency treatment.
5. No separate registration fee is charged to the patient. It is included in the consultation fee.
6. A medical record is maintained for each patient. A medical record can be associated with multiple billing records pertaining to each visit of the patient.
7. No nurse is assigned to an empty ward.
8. The system also allows patients to access their billing and medical records.

Problem Statement:

The WolfHospital Management is created for use by the hospital management staff which contains information about the staff, the wards in the hospital, patients visiting the hospital, and their medical and billing records.
The registration staff checks-in patients into the hospital and assigns them wards. Each ward has a responsible nurse that takes care of the patients allotted to that ward. Patients are treated by doctors and nurses. The patients are suggested medications and tests to be carried out by specialist doctors. The patients are billed according to their rendered services at the hospital such as doctor consultation, prescribed medications and tests, along with the charge of their stay at the hospital. The medical records of the patients are then saved in the database.

A few advantages of using a DBMS for the above mentioned problem statement would be:

- Minimized Data Inconsistency: Data Inconsistency exists when a different version of the same data appears at different places leading to chances of Updation Anomaly, thus, designing proper DBMS reduces the probability of data inconsistency.
- Improved Data Access: Designing a proper DBMS allows us to answer queries about the data in the database effectively and quickly. A DBMS facilitates faster and efficient reporting of the data and eases searching through the database .
- Privacy: Implementing databases in consideration of the data-views for each user role allows us to maintain privacy. The users can only access the data entities that are available for them. A DBMS ensures privacy using different access constraints on the data for different users.

2. User Classes:

The system supports four user classes that are;

Registration/Billing Staff: are the major class of users who will create and maintain the data of staff, patients and wards of the Hospital. They are also responsible for creating billing records and checking in patients upon verifying bed availability. They will also be responsible for handling patient check-out and generating required reports.

Doctors: can get information of all the patients they are responsible for and also their entire medical history for a given period of time. They can also view and update their patient's medical records with medication, tests and diagnosis results. Also, they can view their own personnel data and other staff's information at the Hospital to provide test recommendations.

Patients: can view their own personnel data and also their medical and billing records. They can also generate their medical history report.

Nurses: can view their own personnel data. They can view the ward and its status that they are responsible for. They can also view and update their patient's medical records with medication, tests and diagnosis results.

3. Main Entities:

The main entities for WolfHospital Management System are as mentioned below:

Staff - Name, DOB, Gender, Job Title, ProfTitle, Department, Phone, Address

Patient - SSN, Name, DOB, Gender, Phone, Address, Treatment Status (processing treatment plan, in ward, completing treatment)

Check-In - BedNo, WardNumber, StartDate, EndDate

Medical-Record - Patient, Staff, StartDate, EndDate

Billing-Record - Patient, PayeeSSN, ConsultationFee, BillingAddr, PayMethod, CardNo, VisitDate

4. Usage Situation:

Mr. Todd Baker is a 57-year-old male with a chronic heart condition. For the past four years, he has been under the care of Dr. Gregory House, who is a renowned Heart Specialist and cardiothoracic surgeon at Wolf Hospital at North Carolina.

On January 26th, Todd was found unconscious in his backyard and was rushed to the ER at the Wolf Hospital. It was found Todd suffered a major cardiac arrest and Dr. House was paged in after the ER staff took measures to stabilize his conditions. Meanwhile a medical record was created to track the prescribed medications, tests and its results. The billing staff checked bed availability, created a billing record, assigned a bed and completed patient check-in process.

Dr. House retrieves todd's medical history available with the hospital to know the medicines prescribed thus far and his test results. He suggests a few more tests, which are updated to his medical records with their diagnosis, and identifies Todd needs a by-pass surgery. The registration staff queries for staff list of nurses and doctors (to find a anesthetist) and books them as per Dr.House's suggestion.

The surgery goes well, and Todd's billing record is updated with the total fee including accommodation fees, consultation fees, medication and test fees which are settled on checkout.

5. Application Program Interfaces:

Information processing:

createStaff(name, age, gender, jobTitle, professionalTitle, department, phoneNo, address)
return staffID or NULL(error)

createPatient(SSN, name, DOB, gender, phoneNo, address, status)
return patientID or NULL(error)

createWard(capacity, charges)
return wardNO or NULL(error)

updateStaff *(*staffID*, <attribute(s)>*)
return success or failure (error)

- staffID and atleast one attribute(name, age, gender, jobTitle, professionalTitle, department, phoneNo, address) mandatory

updatePatient *(*patientID*, <attribute(s)>*)
return success or failure (error)

- patientID and atleast one attribute(SSN, name, DOB, gender, phoneNo, address, status) mandatory

updateWard(wardNO, <*attribute(s)*>)
return success or failure (error)

- wardNO and atleast one attribute(capacity, charges, staffID) mandatory

deleteStaff *(*staffID*)*
return success or failure (error)

deletePatient *(patientID)*
return success or failure (error)

deleteWard(wardNO)
return success or failure (error)

checkBedAvailability(wardNO/capacityType)
return bedNO or NULL (no bed available)

- Availability check can be done for ward or capacityType (1-bed, 2-bed, and 4-bed ward)
- If both are not provided, returns any available bed.

assignBed(wardNO, BID, patientID)
return success or failure

- wardNO, patientID, and bedNO are mandatory

reserveBed(wardNO,bedNO)
return success or failure

releaseBed(wardNO,bedNO)
return success or failure

Maintaining medical records for each patient:

createMedicalRecord(patientID, startDate, staffID)
return a MedicalRecordId or NULL (error)

updateMedicalRecord(MedicalRecordId, <attributes>)
return success or failure

- The attributes can be medications prescribed, tests, diagnosisResults, endDate

createCheckIn(bedNo, patientID, startDate, endDate)

returns success or failure

updateCheckIn(patientID, startDate, <attributes>
returns success or failure

- Attributes can be endDate, bedNO

Maintaining billing accounts:

createBillingRecord(patientID,consultationFee,payeeSSN,billingAddress,paymentMethod,cardNumber,visitDate, needCheckIn)
returns billingRecordId or null (Error)

- needCheckIn is used to check for bed availability

updateBillingRecord(billingRecordId,<attribute1>,<attribute2>)
returns success or failure

- attribute1 can be consultationFee, payeeSSN, billingAddress, paymentMethod, cardNumber, visitDate, medicationPrescribed, testName
- if attribute1 is medicationPrescribed, testName, an attribute2 which is cost should be provided.

getMedicalRecordForPatient(patientID, startPeriod, endPeriod)
returns a list of medicalRecords or NULL (error)

- if startPeriod and endPeriod is null, entire record list is returned

Reports:

getBedUsage()
returns a list of beds or NULL (error)

- the Bed Object would have the associated Ward number and AvailabilityStatus.

getPatientCount(isUnique)
returns count or 0 (error)

- isUnique is optional, when set to true returns distinct count of patients visited per month
- The count would be returned based on Billing Record visit date.

getWardUsage()
returns percentage or 0 (error)

getPatientListFor(staffID)
returns List of Patients or NULL (error)

getStaffList(role)
returns List of Staff or NULL(error)

6. Data Views

Registration/Billing Staff:

The registration staff are the admins of the system. They have access to all the entities and their attributes in the system like Staff (Name, DOB, Gender, Job Title, ProfTitle, Department, Phone, Address), Patient (SSN, Name, DOB, Gender, Phone, Address, Treatment Status), Medical Records (Medications, Tests), Ward Details (WardNumber, BedID, Staff), Billing Records (Fee, VisitDate, PaymentMethod, PaySSN, CardNumber, BillingAddress.

Doctors:

The doctors can view staff details like Name, DOB, Gender, Job Title, ProfTitle, Department, Phone, Address for all the staff members. He can also view information about his own patients like SSN, Name, DOB, Gender, Phone, Address, Treatment Status, medical records of his own patients (medication name of the prescribed medications for his patients, test name, specialist doctor information, and diagnosis results of the prescribed tests of his patients), ward details (bed number and ward number) of his assigned patients.

Nurses:

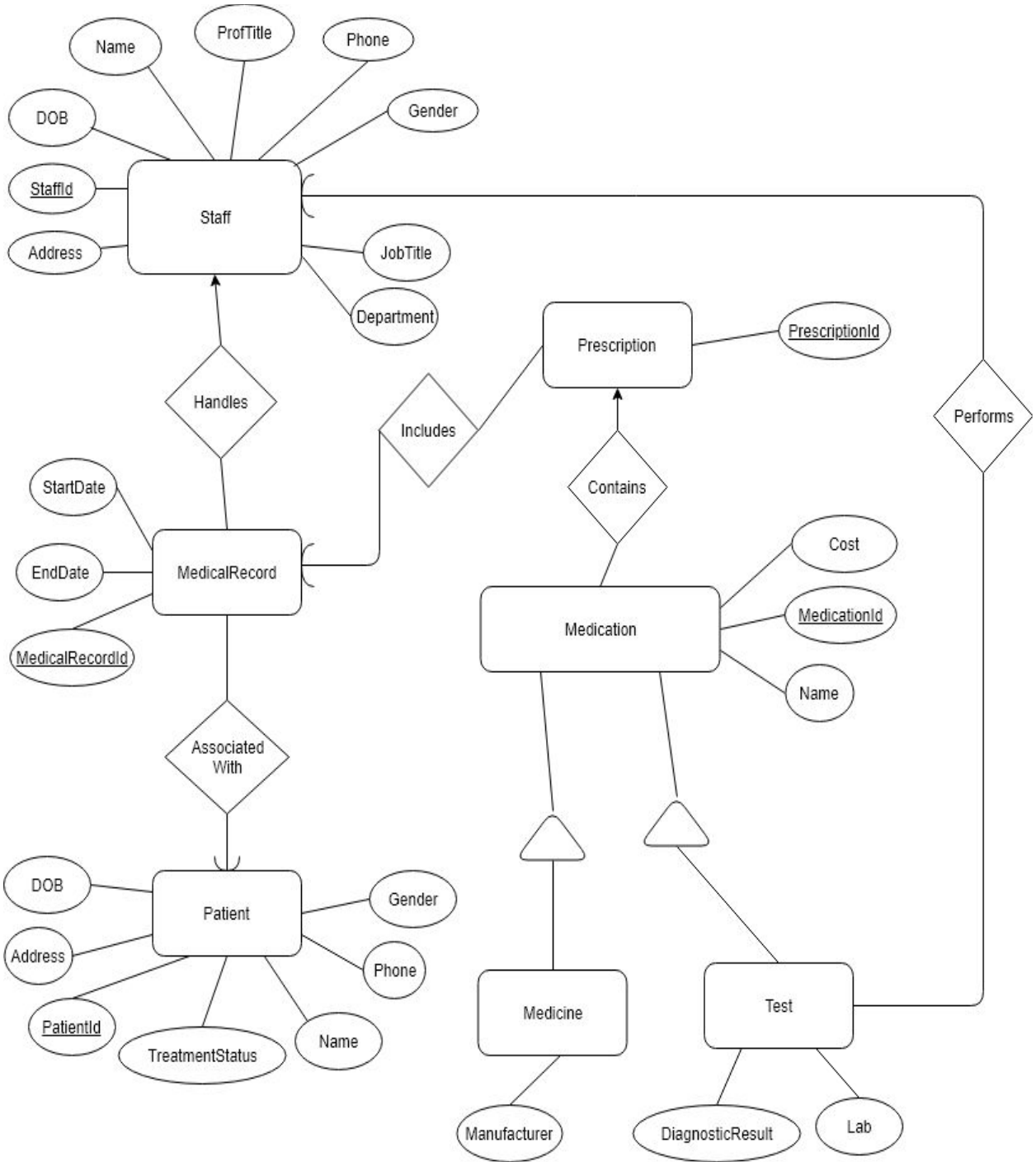
The nurses can view staff details like Name, DOB, Gender, Job Title, ProfTitle, Department, Phone, Address for all the staff members. They can view ward information like WardNumber, BedID, Staff and information about the patients associated with their assigned ward(s) like Name, DOB, Treatment Status. They can also view the medical records of their patients including medication name of the prescribed medications for their patients, test name, specialist doctor information, along with their diagnosis results.

Patients:

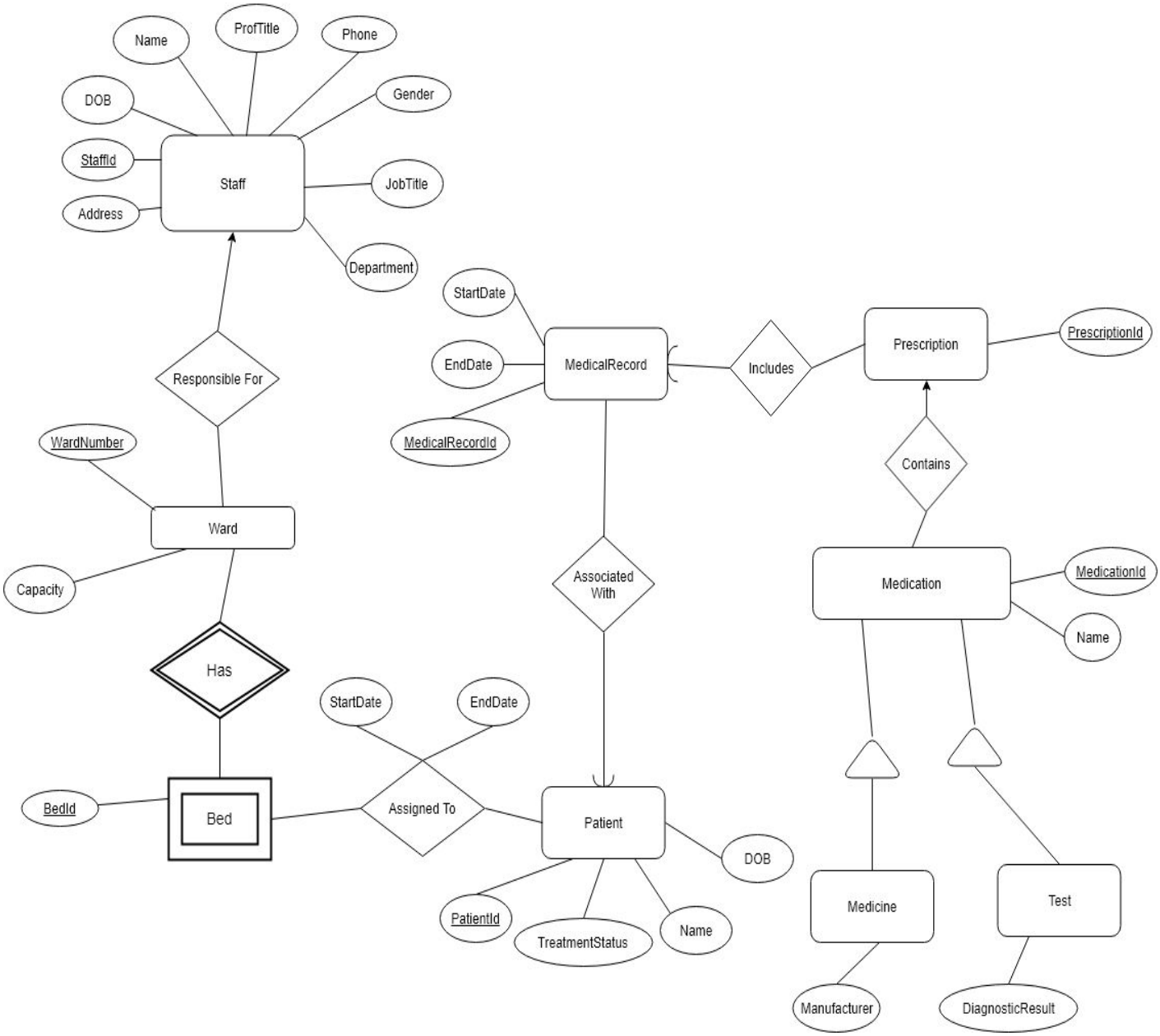
The patient can view his personal details, his medical records including the prescribed medications and the prescribed tests that he was prescribed, along with his billing records. He can also view the Name, ProfessionalTitle, Dept, Gender, PhoneNo of his assigned doctor.

7. Local ER Diagrams:

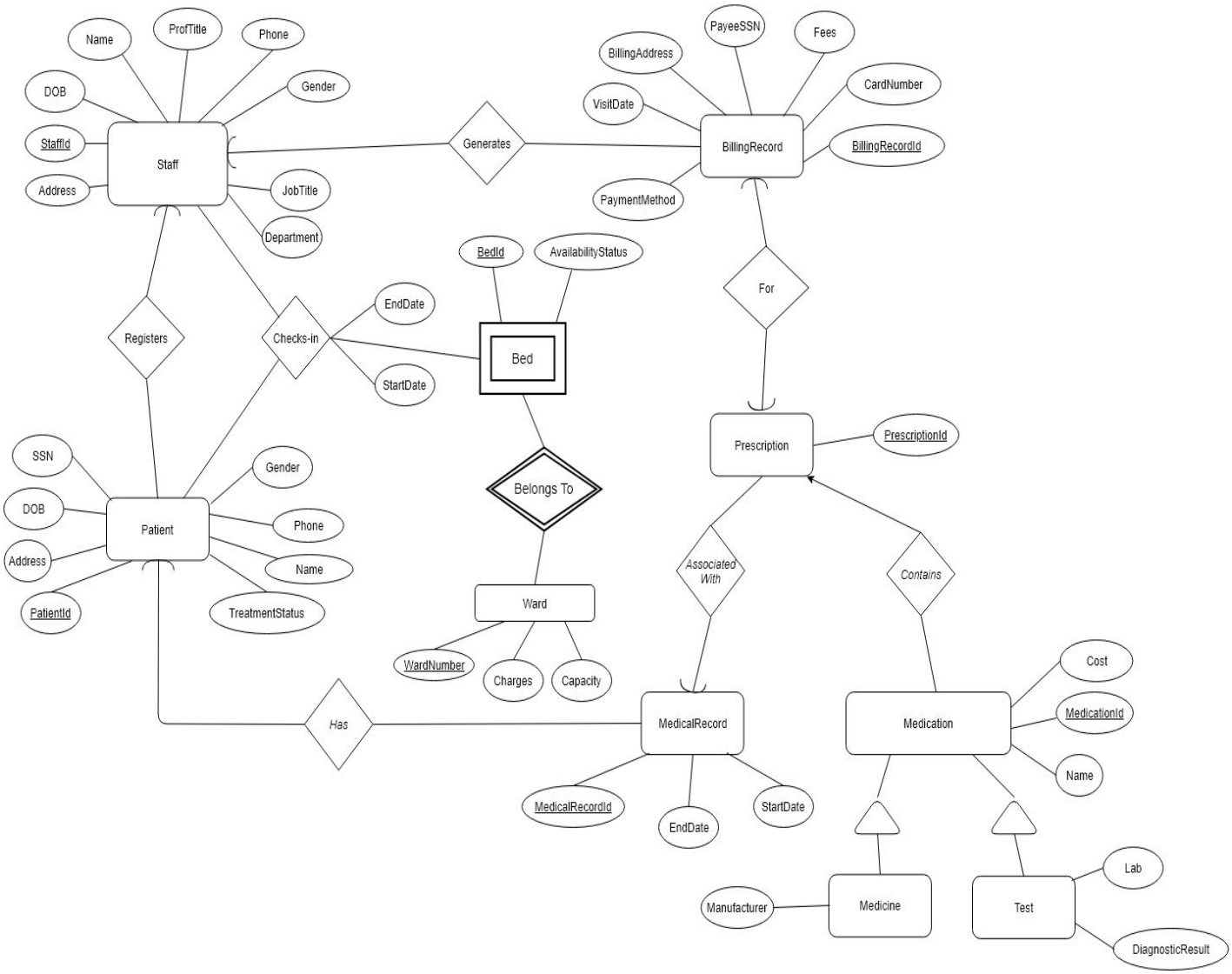
Doctor:



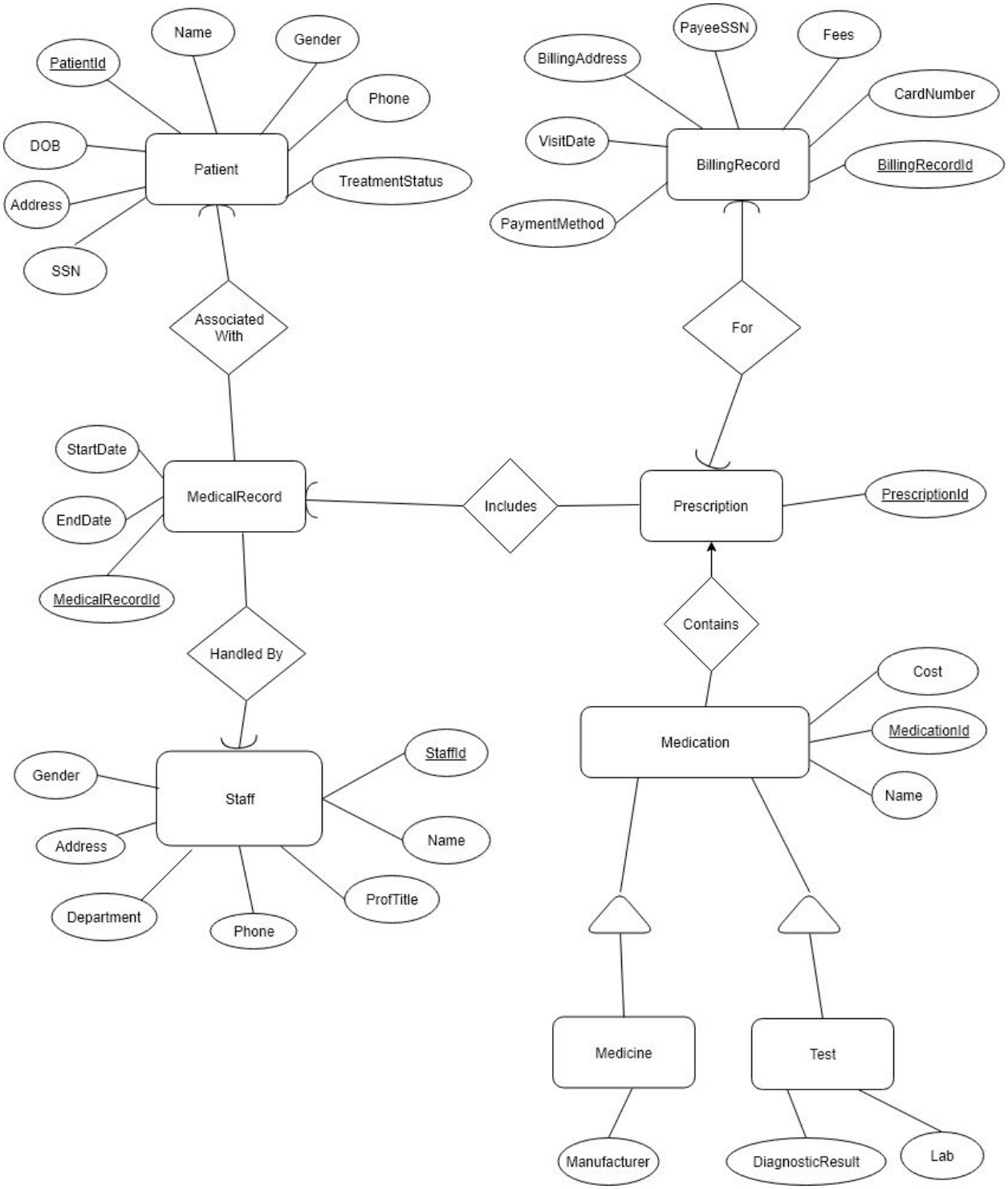
Nurse:


Name
ProfTitle
Phone
DOB
Gender
StaffId
Staff
JobTitle
Address
Department
StartDate
EndDate
MedicalRecord
Includes
Prescription
PrescriptionId
MedicalRecordId
Responsible For
Contains
WardNumber
Ward
Capacity
Associated With
Medication
MedicationId
Name
Has
StartDate
EndDate
BedId
Bed
Assigned To
Patient
DOB
PatientId
TreatmentStatus
Name
Medicine
Test
Manufacturer
DiagnosticResult

Registration/Billing Staff:



Patient:


Name
Gender
PatientId
Phone
DOB
Patient
TreatmentStatus
Address
SSN
Associated With
PayeeSSN
Fees
BillingAddress
CardNumber
VisitDate
BillingRecord
BillingRecordId
PaymentMethod
For
StartDate
MedicalRecord
Includes
Prescription
PrescriptionId
EndDate
MedicalRecordId
Handled By
Contains
Cost
Medication
MedicationId
Name
Gender
Staff
StaffId
Address
Name
Department
Phone
ProfTitle
Medicine
Test
Manufacturer
DiagnosticResult
Lab

8. Local E/R Documentation:

The users of the WolfHospital Management System are Registration/Billing Staff, Doctor, Nurse, and Patient.

Design Highlights:

- We have maintained a single table called “staff” to maintain the details of registration staff, doctors and nurses. We did not divide the staff class into subclasses assuming that the registration staff and nurses will also have professional titles.
- Each medical record will be created by a doctor which will be associated with one specific patient for a treatment.
- Each medical record can have multiple prescriptions and each prescription can have one or more medications(medicines/tests) associated with it. A prescription will be there for each visit of the patient and hence a billing record is directly related to a unique prescription.
- Medication, medicine and test have a parent child relationship (is-a relationship) since medicines and tests are a type of medication which are sharing similar attributes like cost and name.
- We have identified bed as a weak entity since bed IDs are unique only within each ward and they can be uniquely identified along with ward number.

Decisions related to the primary attributes of the entities:

- **Staff**: we have identified StaffId as the primary key since none of the attributes(or a combination) can uniquely identify a staff. Multiple staff can have similar attributes.
- **Patient**: A patient will have a unique SSN. As SSN attribute is made optional for a patient and it can have null values, it cannot be used as a primary key. So, we have identified PatientId as the primary key for this table.
- **MedicalRecord**: A medical record is created for a patient for a treatment and a patient can have multiple treatment going on at the same time(same duration) in the same hospital. Therefore none of the attributes can uniquely identify a medical record. So, we use MedicalRecordID as the primary key for this entity.
- **Bed**: The Bed attribute has Bed Id attribute and as the same bed id could be used in other wards as well, we have identified wardNo and BEdId as the composite primary key for this entity.
- **Ward**: The ward can be uniquely identified by using the ward number attribute and it is used as the primary attribute for ward entity.
- **BillingRecord**: A Billing record could be uniquely identified by Patient Id and Visit date of the patient in some scenarios where the same patient could not visit the hospital

twice. In order to allow the generation of bill for any patient who visits the hospital multiple times on the same day, we identified Billing Record Id as the primary attribute for this entity.

- **Medication(Test/Medicine)**: The name for two medicines or lab test can be similar and can have similar cost associated with them. Hence the field attributes cannot be used as a primary key.

Key relationships:

- **Checks In/Assigned To**: A registration staff checks in a patient to a bed of a particular ward. As this connection type is many-to-many, we will have a separate relational table called “CheckIn”.
- **Belongs To**: This is a weak relationship that connects a weak entity type Bed with Ward. Bed entity uses its BedId and borrows WardNumber from Ward entity to uniquely identify its entries in Bed table.
- **Handles**: Doctor handles(creates and maintains) medical records of his/her patients.
- **Associated With**: Medical records are associated with patients.
- **Includes**: A medical record can include one or more prescriptions.
- **Contains**: A prescription can have zero or more medications/tests.
- **Performs**: A doctor performs a specific clinical test for a particular patient.
- **Responsible Fo**r: A nurse is responsible for a ward.
- **Registers**: Registration staff are responsible for registering patients to hospital.
- **For**: Each billing record will be associated with one prescription given to a particular patient.
- **Has**: Each patient will have one or more medical records associated with them.
- **Generates**: Registration staff generates billing record for each visit of a patient to the hospital.

9. Local Relational Schemas:

Doctor:

Staff(, Address, DOB, Name, ProfTitle, Phone, Gender, JobTitle, Department)

Patient(PatientId, DOB, Address, Name, Phone, Gender, TreatmentStatus)

MedicalRecord(MedicalRecordId, StaffId, PatientId, StartDate, EndDate)

Prescription(PrescriptionId, MedicalRecordId)

Medicine(MedicationId, Name, Cost, Manufacturer, PrescriptionId)

Test(MedicationId, Name, Cost, DiagnosisResult, Lab, StaffId, PrescriptionId)

Nurse:

Staff(StaffId, Address, DOB, Name, ProfTitle, Phone, Gender, JobTitle, Department)
Patient(PatientId, DOB, Name, TreatmentStatus)
Ward(WardNumber, Capacity, StaffId)
Bed(BedId, WardNumber)
AssignedTo(BedId, WardNumber, PatientId, StartDate, EndDate)
MedicalRecord(MedicalRecordId, PatientId, StartDate, EndDate)
Prescription(PrescriptionId, MedicalRecordId)
Medicine(MedicationId, Name, PrescriptionId)
Test(MedicationId, Name, Cost, DiagnosisResult, PrescriptionId)

Registration/Billing Staff:

Staff(StaffId, Address, DOB, Name, ProfTitle, Phone, Gender, JobTitle, Department)
Patient(PatientId, DOB, Address, Name, Phone, Gender, TreatmentStatus, StaffId)
MedicalRecord(MedicalRecordId, PatientId, StartDate, EndDate)
BillingRecord(BillingRecordId, StaffId, PaymentMethod, CardNumber, Fees, PayeeSSN, PaymentMethod, VisitDate, BillingAddress)
Prescription(PrescriptionId, MedicalRecordId, BillingRecordId)
Medicine(MedicationId, Name, Cost, Manufacturer, PrescriptionId)
Test(MedicationId, Name, Cost, DiagnosisResult, Lab, StaffId, PrescriptionId)
Ward(WardNumber, Charges, Capacity)
Bed(BedId, WardNumber, AvailabilityStatus)
ChecksIn(BedId, WardNumber, PatientId, StartDate, EndDate, StaffId)

Patient:

Staff(StaffId, Address, DOB, Name, ProfTitle, Phone, Gender, JobTitle, Department)
Patient(PatientId, DOB, Address, Name, Phone, Gender, TreatmentStatus)
MedicalRecord(MedicalRecordId, PatientId, StartDate, EndDate, StaffId)
Prescription(PrescriptionId, MedicalRecordId, BillingRecordId)
BillingRecord(BillingRecordId, StaffId, PaymentMethod, CardNumber, Fees, PayeeSSN, PaymentMethod, VisitDate, BillingAddress)
Medicine(MedicationId, Name, Cost, Manufacturer, PrescriptionId)

Test(MedicationId, Name, Cost, DiagnosisResult, Lab, StaffId, PrescriptionId)

10) Local Schema Documentation

- All the strong entity sets are converted into relations with the same set of attributes.
- All relationships (many-to-many) with attributes (such as AssignedTo in Nurse and ChecksIn in Registration/Billing Staff) are converted into relations with the same set of attributes.
- Entities with many-to-one relationships (such as MedicalRecord and Prescription) were combined with other relations ([Staff and Patient] and MedicalRecord respectively) to efficiently produce results for queries involving attributes of one relation.
- The E/R viewpoint strategy was used to create relations that included the key attributes from the root and any attributes belonging to Entity sets (such as Medicine and Test).
- The relation for the weak entity set(such as bed) includes all attributes of bed and also the key attributes of the supporting entity set (Ward).