WolfHospital Management System

For WolfHospital

CSC 440 Database Management Systems
Project 1

Renata Zeitler, Andy Zhang, Nicole Hlebak, Alex Sawyer
September 25th, 2017

Assumptions:

1. Nurses, Receptionists, and Admins do not have professional titles.
2. We are not going to check duplicate SSN for patients.
3. No patient will start multiple treatments on the same day- only the day/month/year is recorded.
4. Patients can reserve a specific ward/bed, but they must tell the receptionist at check-in.
5. To check into a ward, SSN is not required since it is optional patient data; however, patientID is.
6. Each type of fee is the same for every patient (for example, a registration fee is always the same amount).
7. The nurse in charge of each ward attends to the patients in the ward. Nurses do not attend to or treat patients outside of the ward they are assigned to.
8. Only one doctor will be performing a test.
9. Patients are limited to one ward/bed per visit; this will not change throughout their visit.

1. Problem Statement:

We will design a WolfHospital Management System for the staff at WolfHospital to maintain staff information, medical records, patient information, billing account information, ward information, and check-in information. There are four major types of tasks that will need to be performed: information processing, maintaining medical records for each patient, maintaining billing accounts, and creating reports that contain patient medical information, hospital ward status, or other statistics pertaining to the hospital. In regards to patients, users (hospital staff) will be able to enter all information pertaining to that patient, as well as assign them to a ward. The patients medical records will also be maintained throughout their visit.

Because there will be multiple users accessing and updating the information concurrently, it is better that we use a database system rather than a simple set of files. Given that large amounts of data will be entered and updated frequently, it is more efficient to use a database and to query for data as opposed to searching through files that could potentially be outdated. Having to rewrite an entire file or having each staff member reload a file before use is incredibly inefficient, so using something like a simple set of files is out of the question. Using a database eliminates the possibility of staff overwriting each other when updating information or patients only entering partial information- this prevents the possibility of losing or skewing patient data.

2. Intended Users:

Receptionists: manage registration, patient check-in/check-out, assigning wards, and patient billing. They are responsible for creating the patient billing accounts.

Doctors: analyze medical records to determine treatment plans and report on tests conducted on behalf of the patients.

Nurses: are responsible for tending to wards and patients residing in the wards they are responsible for.

Administrative staff (Admins): have access to all information pertaining to the hospital. These are the users responsible for adding, editing, or deleting staff, and can view the personal information of the staff members. They have complete control of the WolfHospital Management System.

3. Five Main Entities:

1. Staff Information: staff ID, name, age, gender, job title, professional title, department, phone number, and address
2. Patient Information: patient ID, SSN, name, DOB, gender, phone number, address, status
3. Billing Accounts: account ID, SSN of person who is paying, billing address, payment method, card number, Insurance.
4. Ward Information: ward number, capacity, charges per day, responsible nurse
5. Medical Records: patient ID, start date, end date, treatment, prescription, diagnosis details

4. Tasks and Operations- Realistic Situations:

- Situation 1: A patient named Rob Gronkowski, comes into the hospital. As the receptionist is getting Gronkowski's patient information, Rob notices his address was entered incorrectly from his previous visit to the hospital. He talks to the receptionist, who then updates the address of the patient.

- Situation 2: Patient Rob from the previous scenario is finished receiving treatment and goes to checkout with a receptionist. The receptionist updates his billing account, checks him out, and releases the ward.

5. Application Program Interfaces:

- **Information Processing:**
  - enterPatientInfo(patientId, SSN, name, DOB, gender, phone number, address, status, visitDate)
    return confirmation

  - updatePatientInfo(patientId, SSN, name, DOB, gender, phone number, address, status, visitDate)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

  - deletePatientInfo(patientId, SSN, name, DOB, gender, phone number, address, status, visitDate)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

  - enterStaffInfo(staffID, name, age, gender, jobTitle profTitle, department, phone number, address)
    return confirmation

  - updateStaffInfo(staffID, name, age, gender, jobTitle profTitle, department, phone number, address)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

  - deleteStaffInfo(staffID, name, age, gender, jobTitle profTitle, department, phone number, address)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

  - enterWardInfo(wardNumber, capacity, patientId, charges, responsible nurse)
    return confirmation

  - updateWardInfo(wardNumber, capacity, patientId, charges, responsible nurse)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

  - deleteWardInfo(wardNumber, capacity, patientId, charges, responsible nurse)
    return confirmation
    * If NULL value for any of the fields, then they will not be deleted

  - checkWardAvailability(wardNumber)
    return number of beds available

  - assignWard(wardNumber, bedNumber, patientId)
    return confirmation

  - reserveWard(wardNumber, bedNumber, patientId)
    return confirmation

- releaseWard(patientId)
  return confirmation

- **Maintaining Medical Records for each patient:**
  - getMedicalRecords(patient)
    returns list of medical records for the patient (patient ID, ward number, bed number, start date, end date, responsible doctor, prescription, treatment, diagnosis details, testInformation)

  - updateMedicalRecord(patient ID, ward number, bed number, start date, end date, responsible doctor, prescription, diagnosis details)
    returns true if updated and false otherwise.
    * If NULL value for any of the fields, then they will not be updated

  - addMedicalRecord(patient ID, ward number, bed number, start date, end date, responsible doctor, prescription, diagnosis details)
    returns true if added, and false otherwise
    * If NULL value for any of the fields, then they will not be updated

  - addTest(testID, testName, results, cost, startDate, patient ID)
    returns true if added, and false otherwise
    * If NULL value for any of the fields, then they will not be updated

  - updateTest(testID, testName, results, cost, startDate, patient ID)
    returns true if updated, and false otherwise
    * If NULL value for any of the fields, then they will not be updated

- **Maintaining Billing accounts:**
  - checkAvailabilty(listOfWards)
    returns true if available, false otherwise

  - addBillingAcount(patientID, medicalRecord, date, insurance, payerSSN, billingAddress, paymentMethod, cardNumber)
    returns true if billing account was successfully created, false otherwise
    *NULL value allowed for insurance and cardNumber

  - maintainBillingAccount(patientID, accountID, date, medicalRecord, insurance, payerSSN, billingAddress, paymentMethod, cardNumber)
    returns true if billing account was successfully edited, false otherwise
    * If NULL value for any of the fields, then they will not be updated

  - getBillingAccount(patientID, accountID)
    returns the given patient's billing account.

- **Reports:**
  - getHistory(patientID, startDate, endDate)
    returns records for the patient whose date is between startDate and endDate

- getWardStatus( )
  returns a list of usage status for all wards, the number of patients per month, and the ward-usage percentage

- getPatients(staffID)
  returns the patients the doctor is treating

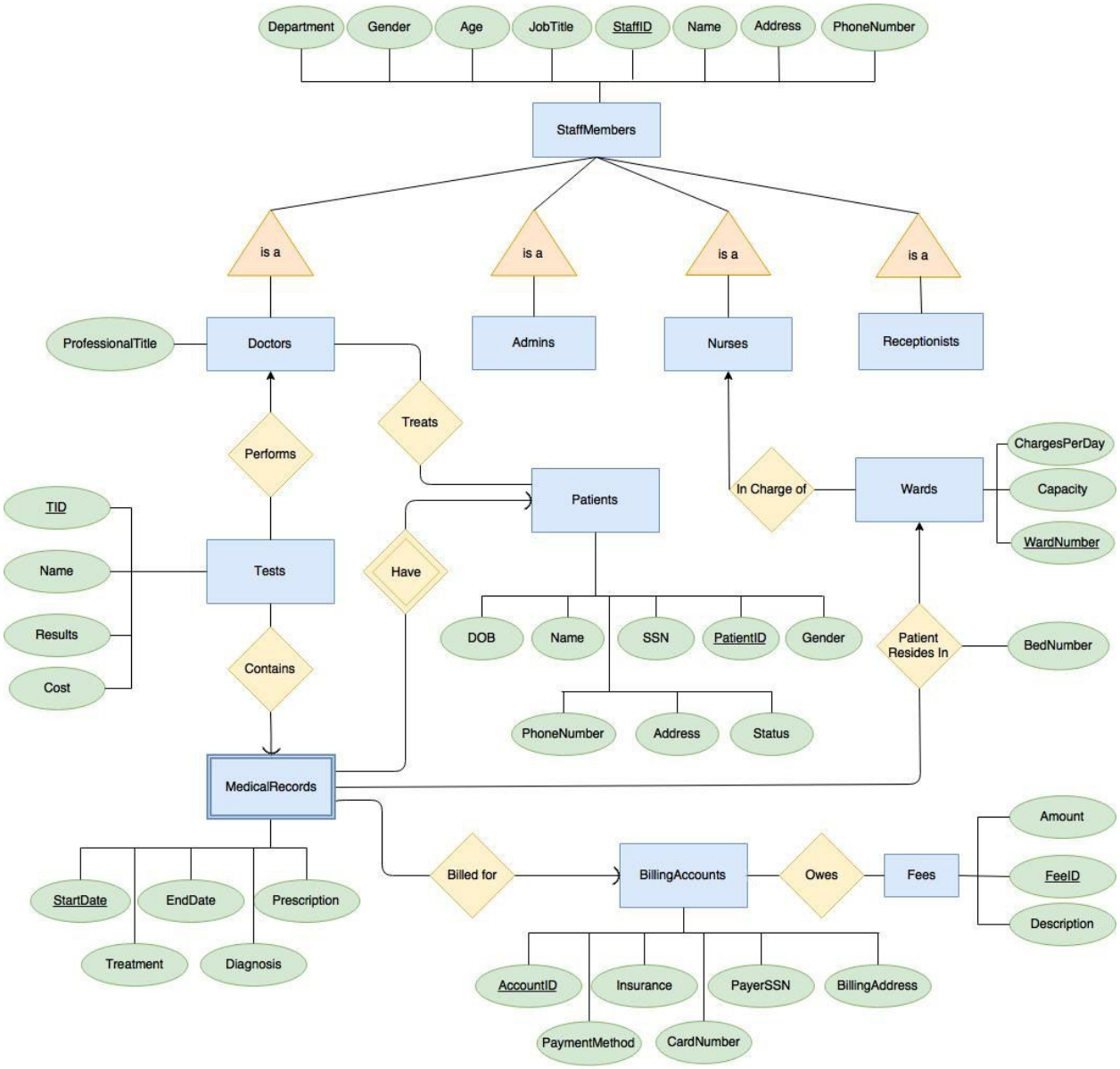
- getStaffInfo( )
  returns information on hospital staff, grouped by role

6. Description of Views:

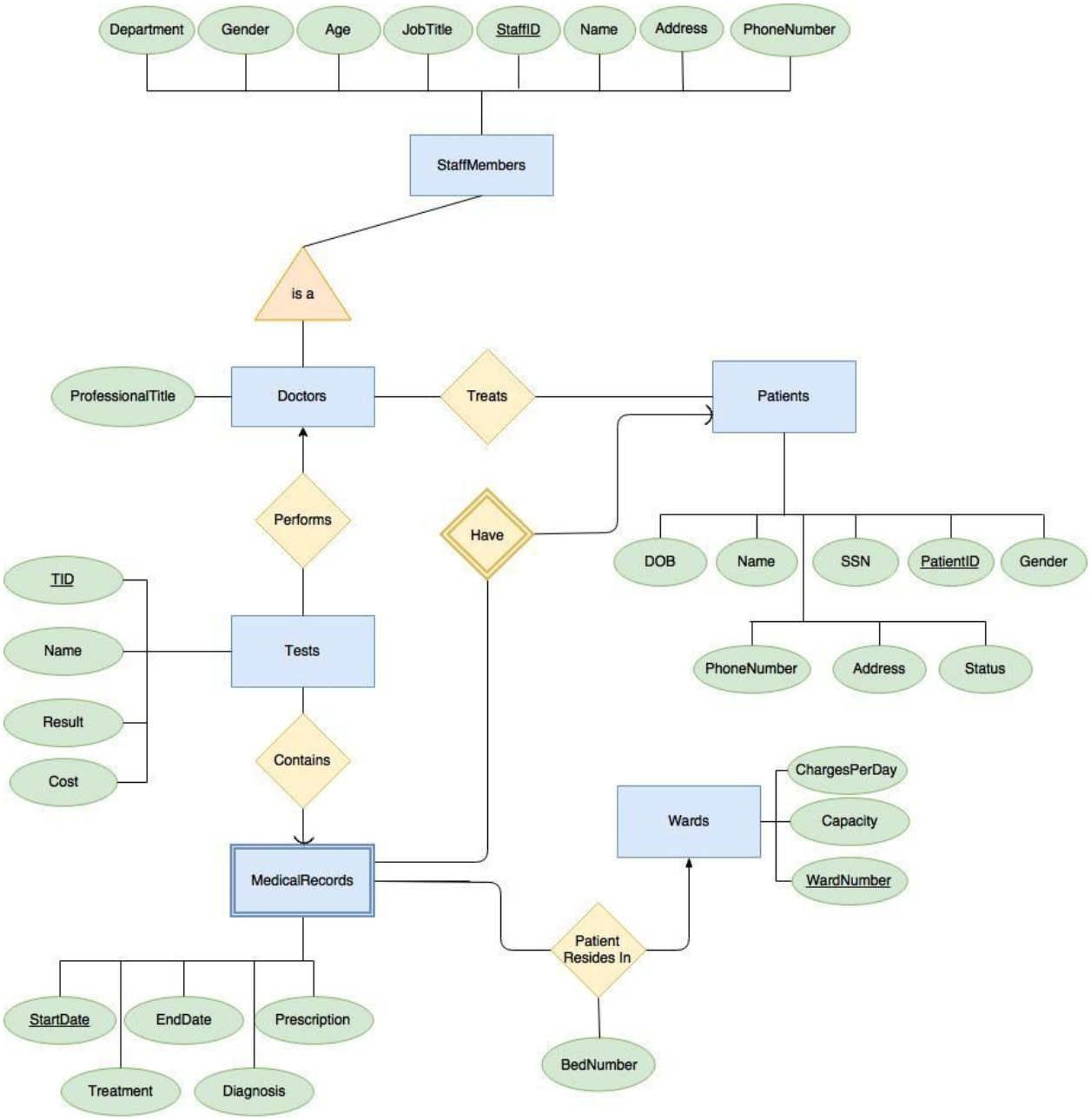
- **Nurses** - can view Staff Information for other nurses, Patient Information, Ward Information, Tests performed on Patients, and Medical Records. They need a general view of everyone working and which wards are filled so that they may be able to do their jobs as well.
- **Doctors** - will be able to access information about Patients, Medical Records, other Doctors, Tests performed on Patients, and Ward information. It is not necessary for them to access data that concerns Billing, which is handled by another class of users.
- **Receptionists** - can access information about Billing Account (and Fees), Patient information, Medical Records and Ward information. They do not need to access Staff Information.
- **Administrators** - will have access to everything in the database. She/He can view all Staff Information, look at all Billing Account information (including Fees), Medical Records, and who is located where in the hospital (Staff, Patient, and Ward information).

7. Local E/R Diagrams:

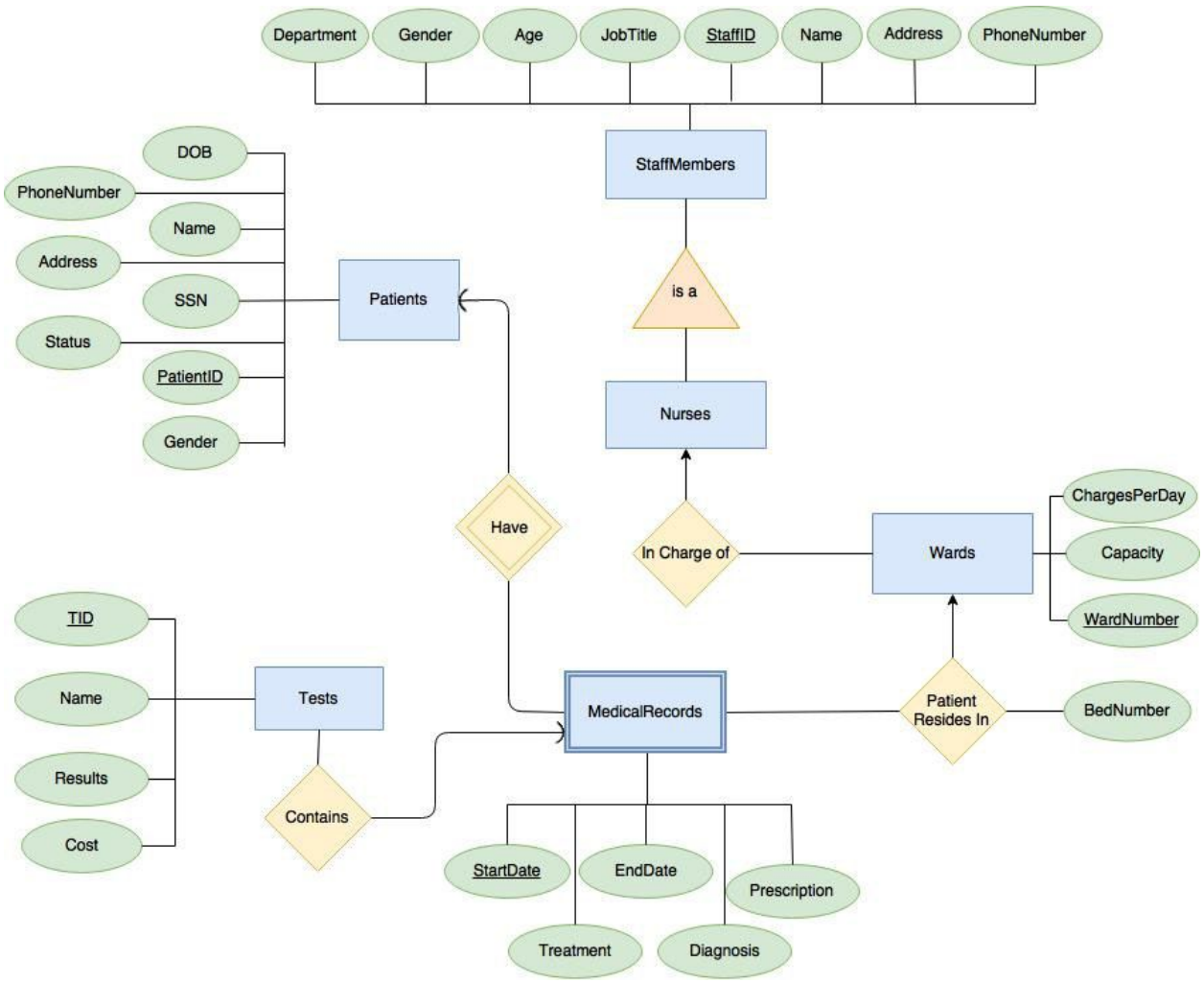
Admin's View:



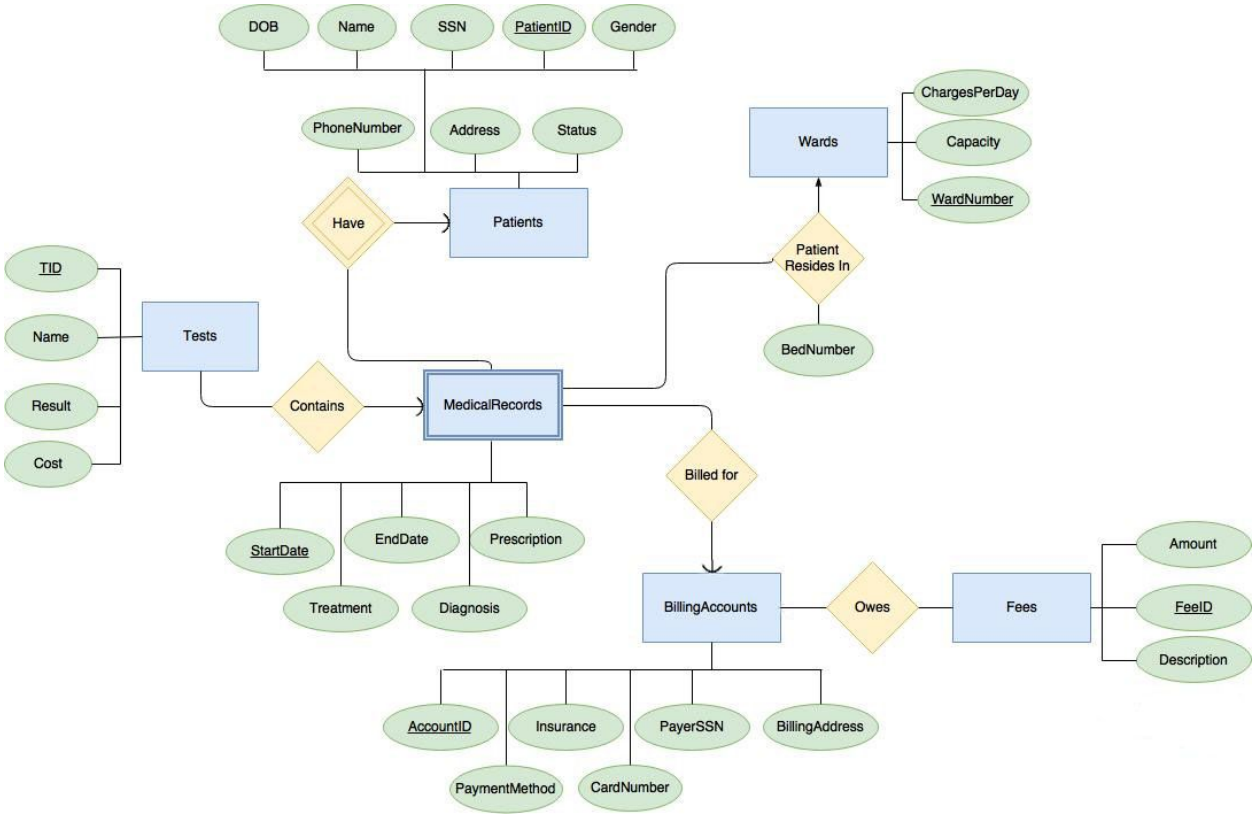
Doctor's View:



Nurse's View:


Department
Gender
Age
JobTitle
StaffID
Name
Address
PhoneNumber
StaffMembers
is a
Nurses
DOB
PhoneNumber
Name
Address
SSN
Status
PatientID
Gender
Patients
Have
In Charge of
Wards
ChargesPerDay
Capacity
WardNumber
TID
Name
Results
Cost
Tests
Contains
MedicalRecords
Patient Resides In
BedNumber
StartDate
EndDate
Prescription
Treatment
Diagnosis

Receptionist's View:



8. Description of Local E/R diagrams:

- The staff working at the hospital can be nurses, doctors, administrators, and receptionists.
- Medical Records is a weak entity set that uses start date and patient id as the key, since no patient will have two medical records with the same start date and a medical record does not exist without a patient.
- Wards uses the Ward Number as a key, since no two distinct wards can have the same number.
- Patients uses a unique id as a key; even though SSNs uniquely identify a person, they are optional.
- Staff Members uses a unique id as a key to account for different people having identical characteristics.
- Tests use unique ids in order to differentiate otherwise identical tests that happen during the same visit.
- Billing Accounts have unique ids (AccountID) that functions as a key. This is so that patients can distinguish between their accounts, if they have multiple with similar information. This also allows the same account to be reused if the payer information is all the same each visit.
- Fees have unique ids so that the may be reused. For instance, one fee can be a registration fee, that can be applied to all billing accounts when they are first created.
- Each nurse can be in charge of multiple wards, but wards can only have at most one nurse in charge as per specified by the narrative.
- Each patient can have many medical records, and each medical record must have exactly one patient. Medical records can only pertain to one person, so it should be limited to a single patient.
- Each medical record can have at most one associated wards, and each ward can have many associated medical records. See assumption #9.
- Each medical record can contain many tests and each test can be contained by exactly one medical record. This is because the test results pertain to one patient only, and should not be reused in other records.
- Each doctor can treat many patients and each patient can be treated by many doctors. Different doctors can perform different procedures depending on their professional title, as specified by the narrative.
- Each test is performed by at most one doctor and each doctor can perform many tests. See assumption #8.
- Each billing account can have many associated medical records and each medical record must have exactly one billing account. This is so that patients are not double charged for a visit if they have multiple billing accounts.
- Each billing account can owe many fees and each fee may be owed by many billing accounts. This is to allow a specific fee to be applied to multiple accounts and reduce

redundancy (We don't want to duplicate fees when every patient has a registration fee of the same amount).

- Age is not stored in Patients, because it can be computed from the patient's date of birth.

9. Local Relational Schemas:

Admins View:

StaffMembers(StaffID, Name, Age, Gender, JobTitle, Dept, Phone, Addr)
Doctor(ProfessionalTitle, StaffID)
Nurse(StaffID)
Receptionist(StaffID)
Admin(StaffID)

Patients(PID, SSN, Name, Dob, Gender, Phone, Addr, Status)
MedicalRecords(PatientID, StartDate, EndDate, Treatment, Diagnosis, Prescription)
BillingAccounts(BID, PayerSSN, payerAddr, PaymentMethod, CardNumber, Insurance)
BilledFor(AccountID, StartDate, PatientID)
Fees(FID, Amount, Details)
Owes(FID, BillingAccountID)

Ward(WardNumber, ResponsibleNurseID, Capacity, ChargesPerDay)
PatientResidesIn(BedNumber, StartDate, PatientID, WardNumber)

Treats(DoctorID, PatientID)
Tests(TID, Name, Results, Cost, StartDate, PatientID, DoctorID)

Doctor's View:

StaffMembers(StaffID, Name, Age, Gender, JobTitle, Dept, Phone, Addr)
Doctor(ProfessionalTitle, StaffID)

Patients(PID, SSN, Name, Dob, Gender, Phone, Addr, Status)
MedicalRecords(PatientID, StartDate, EndDate, Treatment, Diagnosis, Prescription)

Ward(WardNumber, Capacity, ChargesPerDay)
PatientResidesIn(BedNumber, StartDate, PatientID, WardNumber)

Treats(DoctorID, PatientID)
Tests(TID, Name, Results, Cost, StartDate, PatientID, DoctorID)

Nurse View:

StaffMembers(StaffID, Name, Age, Gender, JobTitle, Dept, Phone, Addr)
Nurse(StaffID)

Patients(PID, SSN, Name, Dob, Gender, Phone, Addr, Status)
MedicalRecords(PatientID, StartDate, EndDate, Treatment, Diagnosis, Prescription)

Ward(WardNumber, ResponsibleNurseID, Capacity, ChargesPerDay)
PatientResidesIn(BedNumber, StartDate, PatientID, WardNumber)

Tests(TID, Name, Results, Cost, StartDate, PatientID)
PatientResidesIn(BedNumber, StartDate, PatientID, WardNumber)

Receptionists View:

Patients(PID, SSN, Name, Dob, Gender, Phone, Addr, Status)
MedicalRecords(PatientID, StartDate, EndDate, Treatment, Diagnosis, Prescription)
BillingAccounts(BID, PayerSSN, PayerAddr, PaymentMethod, CardNumber, Insurance)
BilledFor(AccountID, StartDate, PatientID)
Fees(FID, Amount, Details)
Owes(FID, BillingAccountID)

Ward(WardNumber, Capacity, ChargesPerDay)

Tests(TID, Name, Results, Cost, StartDate, PatientID)

10. Local Schema Documentation:

Entity Sets to Relations:

- The entity sets in our diagram were made into relations, with the attributes the same for StaffMembers, Patients, Fees, and BillingAccounts.
- The Entity sets that are subsets of StaffMembers were made into relations based on the E/R viewpoint to avoid redundancy and save table space.

Combining Many-One Relationships:

- In the following places an attribute was made by combining a many-one relationship. This reduces redundancy and decreases the overhead that many tables causes. It makes queries quicker.
  - Weak entity set MedicalRecords was made into a relation with all its attributes plus the PatientID attribute, which is a foreign key from the Patients Entity set. Including the PatientID represents the “Have”

relationship in the diagram between MedicalRecords and Patients; therefore, we do not have a separate relationship in the schema for “Have”. MedicalRecords was made into a weak entity set because it is a many-one relationship that we need to know the patient id in combination with the medical record id to look it up.

- Entity set Wards was made into a relation using all its attributes plus a ResponsibleNurseID that comes from combining its many-one relationship to Nurses. Thus we do not have a separate relationship for the “InChargeOf” relation from the E/R diagram.
- Entity set Tests was made into a relation using its attributes and the PatientID and StartDate from its many-one relationship (“Contains”) with MedicalRecords. It also includes the DoctorID from its many-one relationship with Doctors through the “Performs” relationship.

Relationships to Relations:

- Relationships BilledFor, PatientResidesIn, Owes, and Treats from the E/R diagrams have each been turned into relations in our schema. Their attributes in the schema are the keys of the entities they represent. BilledFor and PatientResidesIn also have the additional attribute of PatientID, which they get from their connection to MedicalRecords (as stated above, MedicalRecords has the PatientID attribute because it is a weak entity set).