WolfHospital Management System

For a hospital in North Carolina

CSC 540 - Database Systems

Project Report #1

Project Team U
Niharika Acharya, Ankit Manendra, Hasham Mukhtar, Vaibhav Nagvekar

Project Assumptions

Assumptions

1. Reception Staff has registration operations that they can do
    a. Registration of patients
    b. Check ins/outs
    c. Admittance to beds/wards
    d. Release of patients
2. Billing staff has all operations that has to do with billing
3. Doctors and Nurses can perform the same operations since both are allowed to treat patients
4. No detailed information about insurance companies are needed for billing except name
5. We are breaking down contact information further into phone number and address
6. We are breaking down billing records into fee charge, test charge, treatment charge and ward charge
7. The fees charge attribute in the billing account table is the combined fees of consultation, medication, and registration and is saved as one number in our system.
8. Beds are required to have a ward number associated with them
9. The total number of beds in the beds table will equal the total capacity of all the wards in the wards table
    a. If new wards are added, new beds are added to the table
10. Doctors are the only one with a department and professional title associated with them
11. Ward usage is the number of occupied beds in that ward out of the total beds available for that ward
12. Bed usage is the total number of beds occupied in the hospital out of the total beds available in the hospital
13. Every user is able to view a patient’s information and their medical record
14. The responsible nurse in a ward is like a head nurse for that ward. Patients can still be treated by other nurses
15. Only Nurses and Doctors can enter/update data about tests and treatments
16. Only Reception Staff can create check in information, patient information, medical records
17. Only Reception Staff, Doctors and Nurses can edit Medical Records
18. Only Reception Staff deals with ward/bed creation, updating and assignment
19. A reserved bed is one where the reserveBed boolean value is true for a bed in the table, but there is not a patient’s name
20. An assigned bed is one where the reserveBed boolean value is true and there is a patient’s id for that bed
21. An empty bed that anyone can use has the reserveBed boolean false and not patient name assigned to it
22. Every entity set/relation has a unique id associated with it to differentiate different entries in the table, even if those entries are related to a single patient (like for medical records or check ins)

Problem Statement

We are designing and building a database system for the WolfHospital Management System for a hospital in North Carolina. The hospital needs a database to maintain all the information of the hospital. This includes information about all the staff, patient information, medical records for their patients, billing accounts, check in information, and ward information. The database system will only be used by the staff of the hospital. Each user of the system will have their own tasks and operations that they can perform. The four biggest tasks that the database system has to take care of are information processing, maintaining medical records for each patient, maintaining billing accounts, and generating reports for the staff.

We chose to use a database as opposed to storing data in text files for this task because it would be difficult to maintain all the information with just text files. A database system handles the data more efficiently and helps in reducing redundancy by establishing relationships between the data. Database systems are also easier to handle which helps to organize data to query data, sort data, and manipulate data in various ways. A database also allows multiple users access to the data.

Intended Classes of Users for System

We have identified the following users for our system:

Doctors

Doctors will be giving seeing and treating patients. When they are with a patient, they will be able to view that patient's data and all medical records related to them. They will also need to know the bed information so that the doctors know where to find their patients, and they'll need to know the ward information so they know what nurse is responsible for ward their patient is in. Doctors can assign tests and treatments to their patients that can be done by themselves or other staff members. They can also edit and update the tests and treatments that the hospital can do.

Nurses

Nurses will be able to treat the patients. They will be able to view patient's medical records and can update the medical records. Additionally, they can view and are responsible to enter and update the treatment and the test information of the patients. They are also responsible to update the medical records of the patients with the results of the tests. A nurse can be assigned as the responsible nurse for a ward. The nurses should know the information of the ward and the bed a patient is assigned to in order to locate the patient.

Reception Staff

Whenever a new patient checks in, the reception staff has the responsibility of creating a record for that patient in the patient information table. Apart from the patient information, it also creates and entry in the medical records and the check-in information table. They can access the wards table to assign a particular ward and bed to a patient. When the patient checks out, they are responsible for releasing the bed occupied by the patient and making all other check out updates in the database.

Billing Staff

It is the task of the billing staff to maintain the billing account for all the patients. They also maintain the check in and check out details about the patients. Billing staff also have the ward information. They check if the billing for the patient is done so that the patients can be released.

Main Entities of System

These are the main entities that we feel we need to keep track of in the hospital system:

- **Staff Information**
    - staff ID, name, age, gender, job title (doctor, nurse, billing staff, etc), professional title (senior surgeon, anesthetist, etc), department (Neurology Department, Oncological Surgery, SCT Laboratory, etc), phone number, address
- **Medical Records**
    - Medical record ID, patient ID, start date, end date, responsible doctor, prescription, diagnosis, treatment, active
- **Patient Information**
    - patient ID, SSN, name, date of birth, gender, age, phone number and address, status (processing treatment plan, in ward, completing treatment)
- **Billing Account**
    - Billing Account ID, patient ID, SSN of the person responsible for the payment, billing address, payment method, card number, check number, insurance company name, ward charge, test charge, treatment charge, fee charge, total charge, start date, end date, settled
- **Ward Information**
    - ward ID, capacity (the number of beds), charges per day, responsible nurse

Usage Situations

When a new patient checks in to the hospital, we will create a new record for patient information in our database. Then we will create a new medical record and assign the patient to a bed in a ward.

When a patient is ready to leave the hospital, we will update the patient's medical record and check in information. We will also create a billing account for the visit.

API's

Information Processing

1. createStaff(name, gender, age, jobTitle, phone, address)
    a. Return staffID or NULL for error
2. createPatient(name, DOB, gender, age, phone, address, SSN)
    a. Return patientID or NULL for error
3. createWard(capacity, nurseID, wardNumber)
    a. Return wardID or NULL for error
4. createBed(wardID)
    a. Return bedID or NULL for error
5. updateStaff(name, gender, age, jobTitle, phone, address, professionalTitle, department, staffID)
    a. Return confirmation
    b. If the attributes, *name, gender, age, jobTitle, phone, address, professionalTitle, department* are not given, then this will not be updated
6. updatePatient(name, DOB, gender, age, phone, address, SSN, status, patientID)
    a. Return confirmation
    b. If the attributes, *name, DOB, gender, age, phone, address, SSN, status* are not given, this will not be updated
7. updateWard(capacity, nurseID, wardID, chargesPerDay)
    a. Return confirmation
    b. If the attributes, *capacity, nurseID, and chargePerDay* are not given, this will not update
8. updateBed(bedID, wardID, patient ID, reserveBed)
    a. Return confirmation
    b. If the attributes, *wardID, patientID, and reserveBed* are not given, this will not update
9. deleteStaff(staffID)
    a. Return confirmation
10. deletePatient(patientID)
    a. Return confirmation
11. deleteWard(wardID)
    a. Return confirmation
12. deleteBed(bedID)
    a. Return confirmation
13. reserveBed(bedID, wardID, reserveBed)

    a. Return confirmation
    b. This reserves a bed for a patient in a specific ward
        i. reserveBed is a boolean value to see if a bed is reserved
            1. True means reserve the bed
            2. False means anyone can use it

14. assignBed(bedID, wardID, patientID, reserveBed)
    a. Return confirmation
15. releaseBed(bedID, patientID)
    a. Return confirmation
    b. Frees up a bed that a patient is using so it can be used by others
16. checkWard(wardID, numberOfBeds)
    a. Return list of wardIDs or NULL if not available
    b. Checks to see if wards are available to put a patient in based on the patient's request (1 bed ward, 2 bed ward, etc)
17. createTest(name, price)
    a. Return testID or NULL for error
18. createTreatment(name, price)
    a. Return treatmentID or NULL for error
19. updateTest(testID, name, price)
    a. Return confirmation
    b. If name and price are not given, this will not update
20. updateTreatment(treatmentID, name, price)
    a. Return confirmation
    b. If name and price are not given, this will not update
21. deleteTest(testID)
    a. Return confirmation
22. deleteTreatment(treatmentID)
    a. Return confirmation

Maintaining Medical Records

1. emergencyCheckIn(patientID, startDate)
    a. return checkInID or NULL for error
    b. The above api is for emergency check-in as the patient will be treated immediately and won't have a ward/bed assigned
2. normalCheckIn(patientID, startDate, wardID, bedID)
    a. return checkInID or NULL for error

   b. The above api is for normal check-in of patients. First the wards will be checked for available bed and only then the patient will be checked in.

3. createMedicalRecord(patientID, startDate)
   a. return medicalRecordID or NULL for error
   b. The above api creates a medical record for every visit of the patient.
4. updateCheckIn(checkInID, patientID, startDate, endDate, wardID, bedID)
   a. return confirmation
   b. The above api updates the end date of the patient in the check-in information table whenever the patient is ready to leave the hospital.
   c. If the attributes *patientID*, *startDate*, *endDate*, *wardID*, *bedID* are not given, this will not update
5. updateMedicalRecord(medicalRecordID, patientID, startDate, endDate, doctorID, prescription, diagnosis, treatment, active)
   a. return confirmation
   b. If the attributes *patientID*, *startDate*, *endDate*, *doctorID*, *prescription*, *diagnosis*, *treatment*, *activ*e are not given, this will not update
6. insertTestInfo(patientID, startDate, testID)
   a. return confirmation
   b. The above api inserts Test information for a patient in tests_for_patients table.
7. insertTreatmentInfo(medicalRecordID, patientID, startDate, treatmentID)
   a. return confirmation
   b. Inserts treatment in the medical record table
8. updateTestInfo(patientID,startDate, testID, results)
   a. return confirmation
   b. The above api updates test results in the tests_for_patients table.

Maintaining Billing Accounts

1. createBillingAccount(patientID, startDate)
   a. Return billingAccountID or null for error
2. updateBillingAccount(billingAccountID, SSN, billAddr, payment type, credit card number, check number, test charge, ward charge, treatment charge, fee charge, total charge, patientID, startDate, endDate)
   a. Return confirmation
   b. If the attributes billingAccountID, SSN, billAddr, payment type, credit card number, check number, test charge, ward charge, treatment charges, fee charge, total charge, patientID, startDate, endDate are not given, this will not update
3. checkHospitalSpace()
   a. Return boolean (yes or no)
   b. Checks if there is space in hospital to admit a patient to a ward/bed

4. releasePatient(patientID, startDate, endDate)
    a. If all accounts for the patient is settled then return confirmation

Reports

1. generateMedicalReport(month, year, patientID)
    a. returns the start and end date of a particular treatment, responsible doctor, prescription, diagnosis details
        i. year needs to be provided for finding the medical records
        ii. patientID is also mandatory
        iii. if month is null, it will return all the records for that year
2. generatePatientsPerMonth (month, year)
    a. returns the total number of patient entries recorded for that month and year in the patients table
    b. for the already existing patients in the database who were admitted sometime in the past, we keep a boolean field active. Hence for patients who comes again to the hospital, we will just update this to true and use this to fetch this report.
3. generatePatientsUnderDoctor (staffID)
    a. returns relevant records of all the active patients whose current medical record has the same responsible doctor as the staff ID
    b. we have to use medical records table of the patient to fetch the responsible doctor here
4. generateHospitalStaff (jobTitle)
    a. returns all relevant information for the hospital staff whose job title is the same as the argument of this function
5. generateCurrentUsageWards(wardID)
    a. returns the usage status of the wards i.e. the number of beds available currently in the given ward
6. generateCurrentUsageBeds()
    a. returns the usage status of all the beds

Views of the Data

Doctors

Doctors will be able to see Patient, Medical Record, Bed, Staff, Test and Treatment data in four tables. One will combine Patient, Medical Record, Tests, and Bed information. The other table will show staff information about Doctors and Nurses. Then there will be separate tables for Tests and Treatments.

Nurses

Nurses will view the Patient, Medical Record,Staff, Bed and Treatment data. A ward has a responsible nurse and hence each nurse should be able to view Ward table. A nurse can view the Treatments and Tests data and can update the medical records of patients with test results.

Reception Staff

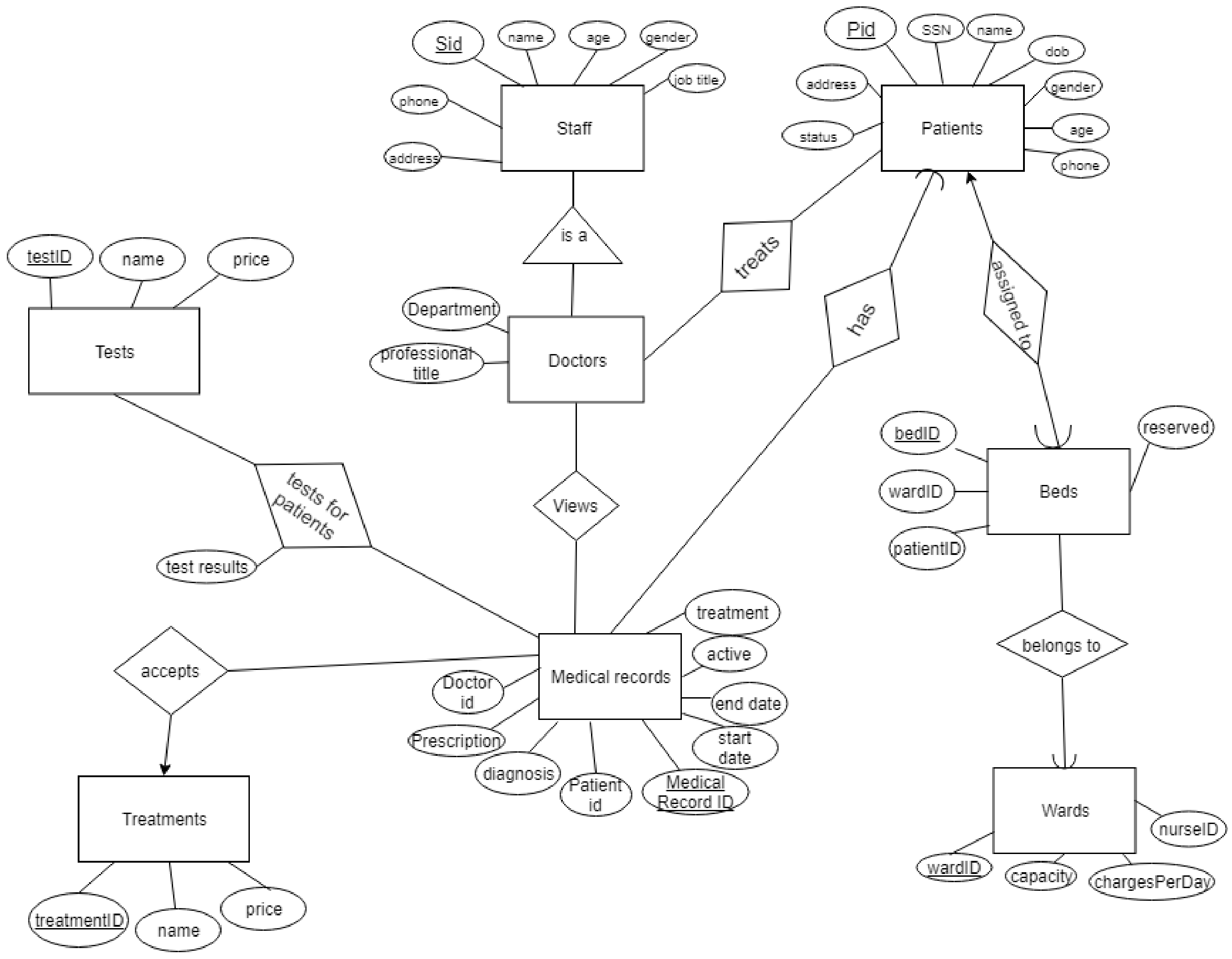
Reception staff will be able to view Staff information, Patient Information, Ward Information, Check-in information, Beds and the Medical Records table. We believe the Billing Account table shouldn’t be visible to the Reception Staff as the Billing Staff will be taking care of that.

Billing Staff:

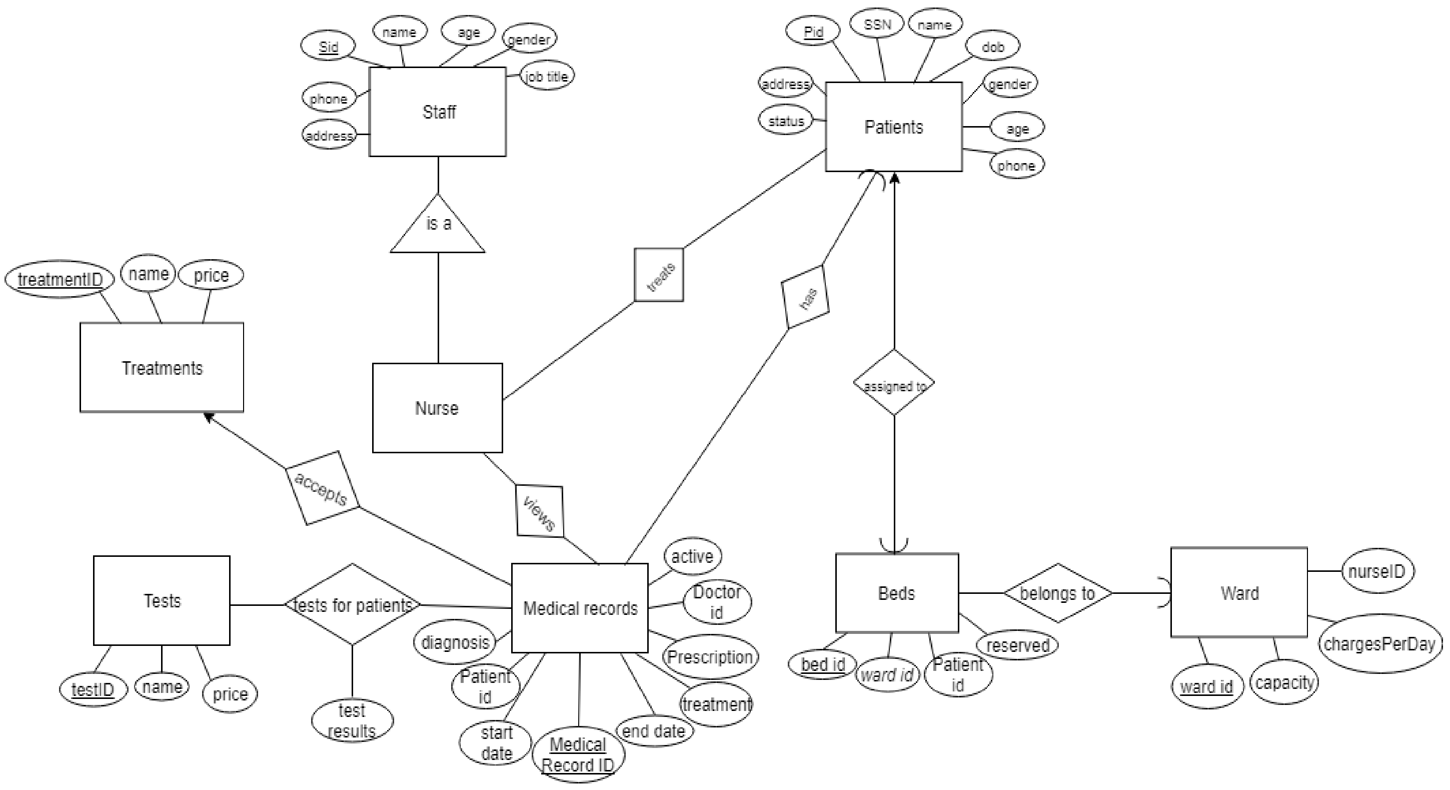
Billing staff must be able to view the patient’s information. They should be able to view the check in and check out information of the visitors/patients. Billing staff must have information about the billing accounts of the patients. They should also view ward information.

Entity Relationship Diagrams

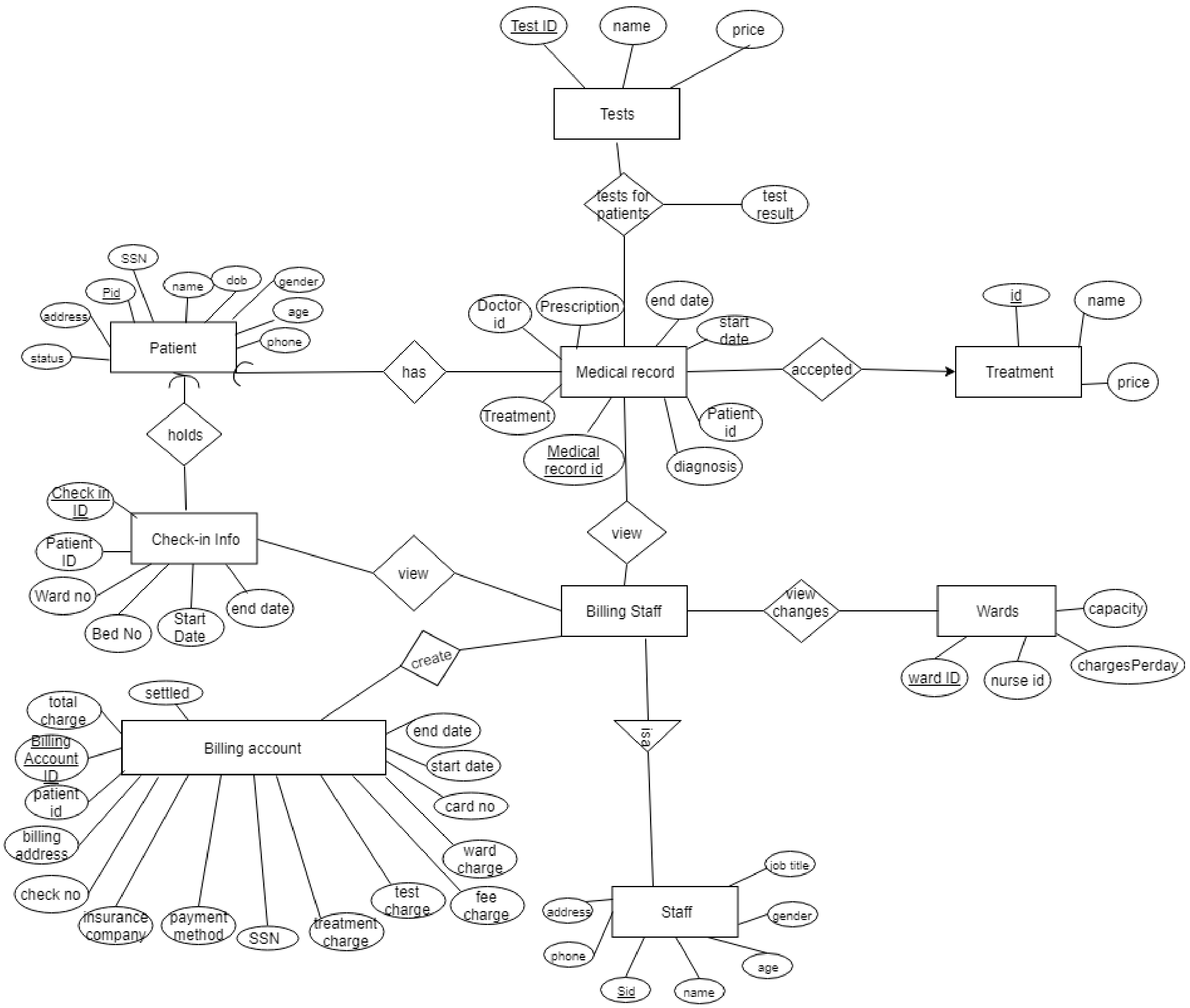
Doctor



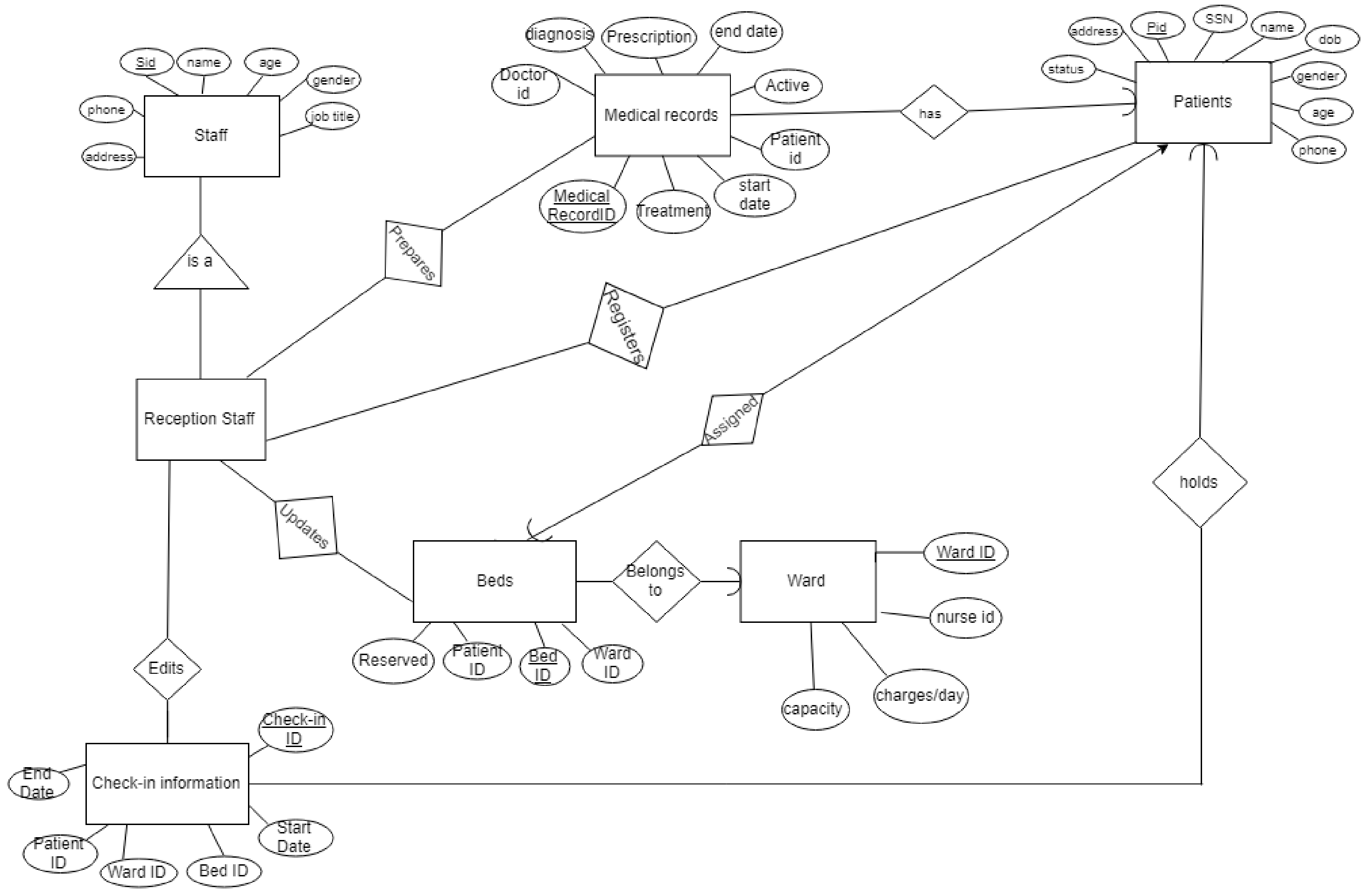
Nurse


Sid
name
age
gender
job title
phone
address
Staff
is a
Nurse
Pid
SSN
name
dob
address
gender
status
Patients
age
phone
treats
has
assigned to
treatmentID
name
price
Treatments
accepts
views
Tests
tests for patients
Medical records
active
Doctor id
diagnosis
Prescription
Patient id
treatment
start date
Medical Record ID
end date
testID
name
price
test results
Beds
belongs to
Ward
nurseID
chargesPerDay
reserved
bed id
ward id
Patient id
ward id
capacity

Billing Staff



Reception Staff



Entity Relationship Diagram Documentation

- The Staff in the hospital can be Reception Staff, Billing Staff, Doctors or Nurses which is why we have an ISA relationship from the Staff entity set in the ER Diagrams
- Patients, Medical Records, Billing Accounts, Check In Information, Wards are from the list of main entity sets listed previously in the report

- We created a Bed entity so that it would be easier and more efficient to track information about the beds
    - What ward they are in and what patient is assigned or reserved for that ward
    - This would reduce redundancy in the wards table by not having multiple entries for the same ward number
- We created a Test entity set that keeps track of all the tests that the hospital can do. This table has a price attribute so that it's easier to find the cost to bill the patient based on what tests they have taken
- We created a relationship called Test For Patients that connects Tests entity and Medical Records entity. This is used to keep track of the multiple tests that are associated with a single medical record. This table would have medical record and test id which is gets from the respective entity. It has its own attribute, test result for each test.
- We created a Treatment entity set to keep track of all the treatments a patient can have. Having it as an entity set and giving it a price attribute helps with billing.
- A medical record can have at most one treatment. No treatment means just consultation, otherwise they will have a particular treatment.
- Each entity set has their own unique id to identify them
- Relationships
    - Views
        - Billing Staff can view many Medical Records and Medical Records can be viewed by many Billing Staff.
        - Billing Staff can view many Wards and Wards can be viewed many Billing Staff.
        - Billing Staff can view many Check In information and Check In information can be viewed many Billing Staff.
        - Billing Staff can view many wards and wards can be viewed by many Billing Staff
        - Nurse can view many medical records of patients and each medical record can be viewed by several nurses.
        - Doctors can view many medical records of patients and each medical record can be viewed by many Doctors
    - Treats
        - This is a many to many relationship between nurse and patients. A nurse can treat many patients. Similarly a patient can be treated by many nurses.
        - Doctors can treat many patients and patients can be treated by many Doctors
    - Assigned/Assigned To
        - Patients are assigned to exactly one bed. However, a bed can up to one patient or no patients.
    - Has

        - A patient has medical records. It can amount to several medical records for a single patient for every visit in the hospital. However a particular medical record has to be for only one patient.
    - Creates
        - Billing staff can create many billing accounts and billing accounts are created by many billing staff
    - Edits
        - A reception staff can edit check-in information. When a new patient check-in, it creates a new entry and when he/she checks out, they update their records.
    - Belongs to
        - A bed belongs to a ward. A single bed can only belong to 1 ward but a ward can have multiple beds.
    - Accepts
        - A medical record can accept at most one treatment but a treatment can be associated with many medical records.
    - Updates
        - A reception staff can update the bed table as and when a patient is added or released from the hospital.
    - Holds
        - Every patient can have multiple check-in information. However, a particular check-in information must be linked to a single patient.
    - Registers
        - Whenever a new patient comes, the reception staff must register this patient into the patients table. This is a many to many relationship.
    - Prepares
        - A reception staff prepares medical records for every patient that comes. This is a many to many relationship.
    - Tests_for_patients
        - This is a many to many relationship between a medical records entity and tests entity. Each medical record can have many tests that the patient take and each test can be associated with multiple medical records.

Relation Schemas

Doctor

Staff(Sid, name, age, gender, job title, phone, address)
Doctors(Sid, department, professional title)
Patients(Pid, SSN, name, dob, gender, age, phone, address, status)
MedicalRecords(medical_record_id, patient_id, startDate, endDate, prescription, doctor_id, diagnosis, treatment, active)
Wards(ward_id, capacity, charges_per_day, nurse_id)
Beds(bed_id, ward_id, Pid, reserved)
Tests(test_id, name, price)
Tests_For_Patients(medical_record_id, test_id, result)
Treatments(treatment_id, name, price)
Views(docotor_id, medical_record_id)
Treats(doctor_id, Pid)

Nurse

Staff(Sid, name, age, gender, job title, phone, address)
Nurses(Sid)
Patients(Pid, SSN, name, dob, gender, age, phone, address, status)
MedicalRecords(medical_record_id, patient_id, startDate, endDate, prescription, doctor_id, diagnosis, treatment, active)
Wards(ward_id, capacity, charges_per_day, nurse_id)
Beds(bed_id, ward id, patient_id, reserved)
Tests(test_id, name, price)
Tests_For_Patients(medical_record_id, test_id, result)
Treatments(treatment_id, name, price)
Views(nurse_id, medical_record_id)
Treats(nurse_id, Pid)

Billing Staff

Staff(Sid,name,age,gender,job title,phone,address)
BillingStaff(Sid)
Patients(Pid, SSN, name, dob, gender, age, phone, address, status)

MedicalRecords(medical_record_id, patient_id, startDate, endDate, prescription, doctor_id, diagnosis, treatment, active)
Tests(test_id, name, price)
Wards(ward_id, capacity, charges_per_day, nurse_id)
Tests_For_Patients(medical_record_id, test_id, result)
Treatments(treatment_id, name, price)
Check-in-Information(check_in_id, patient_id, ward_id, bed_id, startDate, endDate)
ViewCheckIn(check_in_id, billing_Sid)
ViewMedicalRecords(medical_record_id, billing_Sid)
ViewWards(ward_id, billing_Sid)
Billing accounts(billing_account_id, patient_id, SSN, billing address, payment method, card number, check number, insurance company name, ward charge, test charge, treatment charge, fee charge, total charge, start date, end date, settled)
CreatesBillingAccount(billing_account_id, billing_Sid)

Reception Staff

Staff(Sid, name, age, gender, job title, prof title, dept, phone, address)
ReceptionStaff(Sid)
Patients(Pid, SSN, name, dob, gender, age, phone, address, status)
MedicalRecords(medical_record_id, patient_id, startDate, endDate, prescription, doctor_id, diagnosis, active)
Wards(ward_id, capacity, charges_per_day, nurse_id)
Beds(bed_id, ward_id, patient_id, reserved)
Check-in-Information(check_in_id, patient_id, ward_id, bed_id, startDate, endDate)
PreparesMedicalRecords(reception_Sid, medical_record_id)
RegistersPatients(Sid, Pid)
UpdatesBeds(reception_Sid, bed_id)
EditsCheckIns(reception_Sid, check_in_id)

Relation Schema Documentation

- All the main entities were made into a relation with the same set of attributes.
- All the many to many relationships were made into relations. They took the primary keys from the entities they connect and any additional attributes those relationships have.
- We used the E/R approach to deal with the ISA subclasses.
    - By having different relations for each user, it will be easier for our system to look up information specific to those users as opposed to looking at one giant table.

    - The common information for every staff can be retrieved through the staff table. The subclasses will include the differentiating attributes and the primary key of the superclass.
    - In this way for retrieving the common information for the staff, we look up **only one** table. Hence the query execution is faster.
- We have underlined all the primary keys for each relation