**WolfHospital Database Management System**

For a hospital in North Carolina

CSC 540 Database Systems

Project Report #1

Project Team Member:

Xiaoting Fu, Yifan Zhao, Xiao Ma, Xuejiao Fang

### 

### 

### 

# 1. Assumptions and Problem Statement

Assumptions

1. Each patient needs to check-in
2. One nurse can be responsible for multiple wards and when assigning a nurse to a ward the system will select a nurse who has the least number ward that he/she is in charge of.
3. The ward can be used for patient to rest or doctor to give treatment
4. Staffs can be subdivided into Doctors, Nurses and Operators
5. Only Doctors have professional title
6. One check-in can accept many treatments, each treatment generates a medical record.
7. When a patient arrives at the hospital, he will provide his requirement for a ward, if the ward exists, he will create an account otherwise his account is not created.
8. When creating a new patient account, his billing account created at the same time.
9. A billing record is created in the following three cases: when a new medical record is created, when a registration fee is generated and when the accommodation fee is generated.

Problem Statement

The WolfHospital Database Management System will be used for a hospital for North Carolina. The system will be used by patients, doctors, nurses and operators in the hospital. The database will maintain information about patients, staff, wards, check-in, billing accounts, billing records, and medical records. Each group of users will have certain operations they can perform with the system. The use of such a system is of significant advantages due to the following factors:

* A database management system (DBMS) is centralized compared to a file system that can reduce Data Redundancy. For example, if a patient wants to update his/her payment method, the changes will be reflected in the whole system, while user of a file system will need to change multiple files that record the information of this single patient.
* The DBMS can also enforce data consistency and integrity through normalization. The same data is available to all users of the DBMS. For example, the patient and nurse of the hospital will see the same set of the wards information.
* The database is only accessible to authorized users, this ensures data security.
* Faster data query is another advantages. For instance, an operator can check the availability of a two-beds-ward on a specific date by easily querying the database, while such operation is unavailable in a file system.

# 2. User Class

* **Patients** check in hospital and accept treatment. They can view their medical records and billing account , pay for bills when treatment completed.
* **Operators** assist the patients to check in. They can create accounts for patients. View the wards status and assign wards for patients. View patients’ billing account and bill each patient based on the medical records and check-in information.
* **Doctors** are those operate treatments (test included) on patients and generate medical records. Doctor can view patients’ medical history.
* **Nurses** are responsible for the works of wards. They can view the patients’ information and medical records.

# 3. Main Entities

* Patients - Patient ID, SSN, Name, DOB, Gender, Age, Status, Address, Phone Number
* Staff - Staff ID, Name, Age, Gender, Job Title, Professional Title, Department, Address, Phone Number
* Medical Record - Start Date, End Date, Prescription, Diagnosis, Staff ID, Patient ID
* Billing Records - Record ID, Record Type, Create Date
* Wards - Ward Number, Capacity, Charge Per Day, Available Beds
* Beds - Bed Number, Ward Number, Occupied

# 4. Realistic Situations

* A patient provide his requirement for a ward.
* An operator checked in the DBMS if such ward exists or not.
* If the ward exist, and the patient is not in the system, create billing account for the patient along with patient’s basic information then the patient was assigned to a ward and a doctor.
* The doctor diagnosed the patient and created the medical record (prescription and diagnosis). Then a corresponding billing information would be created for the patient.
* After the prescription was given, the billing information would be updated with a charge.
* A nurse is assigned to a ward and the patient at this ward. The nurse assist the doctor to process the treatment so the nurse has access to medical record.
* The medical record and billing account is updated every time a new diagnosis and prescription is given during the patient’s stay in the hospital. And the charge for the ward is updated at the end of the stay.

5.Application Program Interfaces

Information processing

*newStaff*(name, age, gender, department, phoneNumber, jobTitle, professionTitle, address)

return staffID or NULL for error

*updateStaffInfo*(name, age, gender, department, phoneNumber, jobTitle, professionTitle, address, staffID)

return confirmation

* if NULL value for name/age/gender/department/phoneNumber/address, these will not be updated

*deleteStaffInfo*(staffID):

return confirmation

*newPatient*(SSN, name, DOB, gender, age, phoneNumber, address, status)

return patientID or NULL for error

*updatePatientInfo*(SSN, name, DOB, gender, age, phoneNumber, address, status, patientID)

return confirmation

* if NULL value for SSN, name, DOB, gender, age, phoneNumber, address, status, these will not be updated

*deletePatientInfo*(patientID)

Return confirmation

*newWard*(capacity, chargePerDay)

return wardID or NULL for error

*updateWardInfo*(capacity, chargePerDay, wardID)

return confirmation

* if NULL value for SSN, name, DOB, gender, age, phoneNumber, address, status, these will not be updated

*deleteWardInfo*(wardID)

Return confirmation

*isWardAvaliable*(wardCapacity)

return a list of wards that has the specific wardCapacity, else return FALSE

# update check-in table

*assignWard*(patientID, wardID)

return confirmation

# update check-in table endDate, the ward with

*releaseWard*(patientID, wardID):

return

Medical Records Maintenance

*newMedicalRecord*(patientID, staffID, prescription, diagnosis)

return medicalRecordID, createDate or NULL for error

*updateMedicalRecord*(patientID, staffID, prescription, diagnosis, medicalRecordID, createDate)

Return confirmation

* if NULL patientID, staffID, prescription, diagnosis, medicalRecordID, createDate, these will not be updated

Billing Account and Records Maintenance

If *isWardAvaliable*(wardCapacity)== True && *isNewPatient*(name) == True:

createBillingAccount(patientID)

return billingAccountID

*createBillingAccount*(patientID, billingAddress)

return billingAccountID

*createBillingRecords*(billingAccountID, billingType)

return billingRecordID, recordCreateTime

*checkBillingAccountBalance*(billingAccountID):

Return the balance of billing account (if balance is not 0, then there is a payment due)

*checkBillingRecordsStatus*(billingRecordID):

Return the status of billing record( paid or unpaid)

Reports

*generateUserReportforPatient*(patientID, startDate, endDate):

Return

a list of all medical records and a list of all billing records happen between startDate and endDate

# this function don’t have any input but will aggregate information in the whole database

*generateHospitalUsageReport*(): Return numberofWard, numberofPatient/month, ward-usagePercentage

*generateUserReportforDoctor*(staffID):

Return list of patients seen by this doctor

*generateHospitalStaffReport*():

Return hospital staff grouped by their role

# 6. Data Views

Operators

Operators can view any entity information in our model. Operators can create patient accounts, query or update the tables of wards, medical records, billing accounts (including corresponding tables of billing records and payment methods

Operators are users that can view any entity i) and check-in information. Specifically, operators are able to view:

* The table of patients with patient ID, SNN, name, date of birth, gender, age, phone number, address, status
* The table of wards with capacity, charges per day, responsible nurse, and patient’s SSN
* The table of medical records with patient ID, start date, end date, responsible doctor, prescription, diagnosis
* The table of billing accounts with patient ID, SSN, billing address, payment methods, billing records, visit date
* The table of billing records with record ID, record type and create date
* The tale of payment methods with Payment Method ID, Payment Method name, card number and card valid number
* The table of check-in information with patient ID, ward number, bed number, start date, end date
* The table of doctors and nurses with staff ID, name, age, gender, job title, professional title, department, phone number and address

Patients

A patient in our models can view his information including his billing account, billing records, his medical records. He also can view information about doctors and nurses.

Doctors

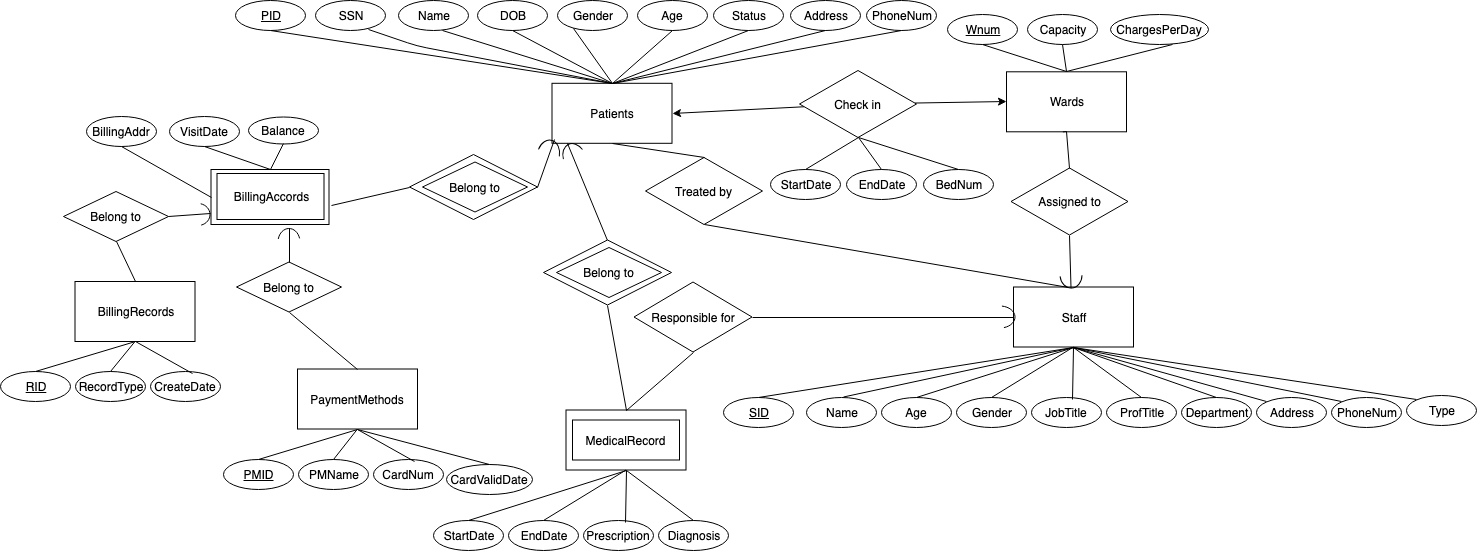
A doctor in our model can view his patients information, create medical records for his patients and view the information of other staff.

Nurses

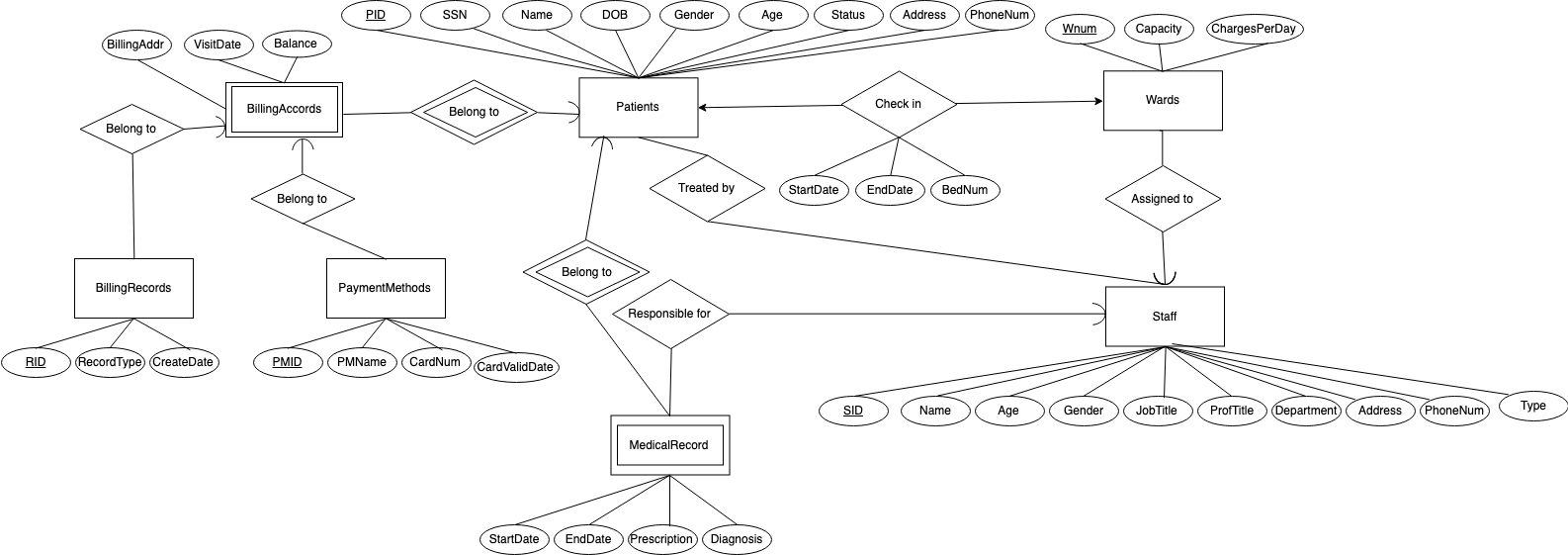
A nurse in our models can view the information of the ward she is in charge of and the patients who live in the wards. She also can view the information of other staff.

7. Local E/R

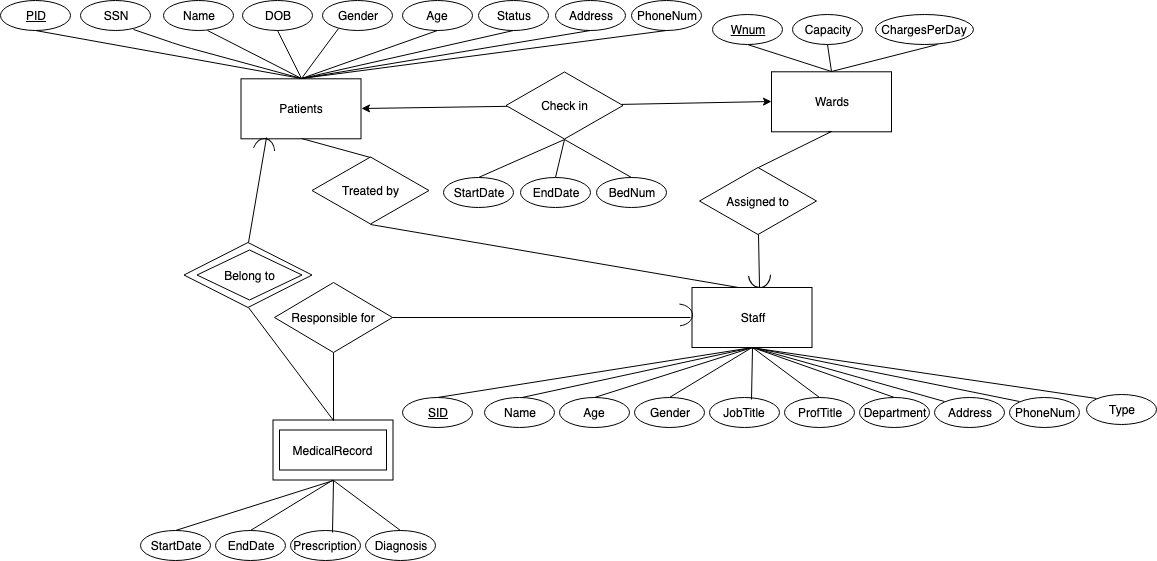
Operator view



Patient view



Doctor and Nurse View



8. Local E/R Documentation

1. There are three views for four types of users: operators view, patient view and doctorAndNurse view. The doctor and nurse share the same view because we have considered that the information they can see should be the same even if they are responsible for different things.
2. We divided the staff into three subclasses: doctor, operators and nurse, since they share the same basic attributes, but they all have different responsibilities and are associated with different other entities. We decided to divide the role of a staff who is a doctor and a staff who is a nurse because doctor is associated with making the diagnosis, prescription and generation of medical records while the nurse is associated with the patient’s ward and also generation of medical records.
3. The operators are in charge of the check-in and check out of patients. They can see patient information, word information, patient’s billing accounts, billing records and medical records as long as the staff related to the patients.
4. We decided that the billing account is created in the check-in process, it starts with zero balance and if the balance is positive the patient need to pay.
5. We also decided that to create a billing record with different TYPE. For example, when patient check-ins, a billing record with type “check-in fee” is created, when a medical record is generated with certain fee, a billing record is created with type “medical fee”, when the patient check out and need to pay the accommodation, a billing record with type “accommodation fee” and its respective quantity is created.
   * We choose to add the type feature to the BillingRecords table since each of the records happen on the moment or before a patient check out is clear with record generation time
6. We also add one entity called “PaymentMethod” since we consider that one patient can have multiple payment methods, i.e. cash, debit card1, credit card1, credit card2, etc. It is inconvenient to make it an attribute and it will be hard to update.

9. Local Relational Schemas

Operator

* Patients(PID, SSN, Name, DOB, Gender, Age, Status, Address, PhoneNum)
* CheckIn(PID, WNum, BNum, StartDate, EndDate)
* Wards(WNum, Capacity, ChargePerDay)
* Staff(SID, Name, Age, Gender, JobTitle, ProfTitle, Department, Address, PhoneNum, Type)
* AssignedTo(SID, WNum)
* MedicalRecord(MID, StartDate, EndDate, Prescription, Diagnosis)
* ResponsibleFor(PID, MID, SID)
* MedRecordOf(PID,MID)
* BillingRecords(RID, RecordType, CreateDate)
* PaymentMethods(PMID, PID, PMName, CardNum, CardValidDate)
* BillingAccounts(PID, BillingAddr, CreateDate, Balance)
* TreatedBy(PID, SID)
* Select(PMID, PID)
* Add(RID, PID)

Patient

* Patients(PID, SSN, Name, DOB, Gender, Age, Status, Address, PhoneNum)
* CheckIn(PID, WNum, BNum, StartDate, EndDate)
* Wards(WNum, Capacity, ChargePerDay)
* AssignedTo(SID, WNum)
* Staff(SID, Name, Age, Gender, JobTitle, ProfTitle, Department, Address, PhoneNum, Type)
* TreatedBy(PID, SID)
* Select(PMID, PID)
* Add(RID, PID)
* MedicalRecord(MID, StartDate, EndDate, Prescription, Diagnosis)
* ResponsibleFor(PID, MID, SID)
* MedRecordOf(PID,MID)
* BillingAccounts(PID, BillingAddr, CreateDate, Balance)
* BillingRecords(RID, RecordType, CreateDate)
* PaymentMethods(PMID, PID, PMName, CardNum, CardValidDate)

Doctors and Nurses

* Patients(PID, SSN, Name, DOB, Gender, Age, Status, Address, PhoneNum)
* CheckIn(PID, WNum, BNum, StartDate, EndDate)
* Wards(WNum, Capacity, ChargePerDay)
* Staff(SID, Name, Age, Gender, JobTitle, ProfTitle, Department, Address, PhoneNum, Type)
* MedicalRecord(PID, MID, StartDate, EndDate, Prescription, Diagnosis)
* ResponsibleFor(PID, MID, SID)
* MedRecordOf(PID,MID)

10. Local Schema Documentation

The entity sets were translated to relations with the same name and all the attributes.

The relationships were translated to relations with the same name or plus some remark for distinguishment and with the keys of the entity sets it connected with as the attributes.