Angela Tseng

[aat52@pitt.edu](mailto:aat52@pitt.edu)

INFSCI 1022

Dmitriy Babichenko

4 April 2020

**Table of Contents**

Abstract/Introduction ------------------------------------------------------------------------------------------------ 3

UML-compliant E-R Model ----------------------------------------------------------------------------------------- 4

Business Rules ------------------------------------------------------------------------------------------------------- 5

Entity/Attribute Descriptions --------------------------------------------------------------------------------------- 7

List of Questions ---------------------------------------------------------------------------------------------------- 11

Closing Section ----------------------------------------------------------------------------------------------------- 13

**Abstract/Introduction**

Hospitals are key institutions and there is need for efficient service delivery in the hospital. As a result of this, a system is needed that will enable hospital management in making effective and efficient decisions.

This paper aims to design a database management system to handle data at a hospital with facilities spread across a variety of locations. Ideally, it would provide easy and fast access to stored data as needed by different users across a hospital with different branches. Data can be added, deleted and updated into the database base. It would have a unique identifier for every person and stores the details of every patient and the staff. A user can search for details of a patient and contact the appropriate parties.

**UML-compliant E-R ModelA close up of a map

Description automatically generated**

**Business Rules**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity 1 | Entity 2 | E1 Cardinality | E2 Cardinality | Business Rules |
| ward | doctor | 1..\* | 1..1 | A ward can have one or more doctors. A doctor can only belong to one ward. |
| ward | nurse | 1..\* | 1..\* | A ward can have many nurses, and a nurse can be transferred across many wards. |
| ward | visit | 1..\* | 1..1 | A ward can have many visits, and a visit is only to one ward. |
| visit | prescription | 1..\* | 0..1 | A visit can lead to a prescription, and a prescription has to result from a visit. |
| visit | patient | 1..1 | 1..\* | A visit can only be traced to one patient, and a patient can visit the hospital multiple times. |
| visit | patient\_bill | 1..1 | 1..1 | A visit leads to one patient bill, and a bill can be traced back to only one visit. |
| patient\_bill | patient | 1..1 | 1..\* | A patient’s bill can only belong to one patient, and a patient can be billed multiple times |
| patient | emergency\_  contact | 0..\* | 1..\* | A patient can have multiple emergency contacts, and an emergency contact can be referenced by several patients. |
| patient | patient\_staff | 1..1 | 1..1 | A patient can have one set of staff referenced information, and a set of information for the staff can only belong to one patient. |
| patient | hospital\_branch | 1..\* | 1..\* | A patient can visit multiple hospital branches, and a hospital branch can have multiple patients |
| patient\_staff | doctor | 1..1 | 0..\* | A patient’s staff information can include only one doctor, and a doctor can be referenced by many staff information records. |
| patient\_staff | nurse | 1..1 | 0..\* | A patient’s staff records can only have one nurse, but a nurse can be present on multiple records. |
| nurse | hospital\_branch | 1..1 | 1..\* | A nurse can only belong to one branch, but one branch can have many nurses. |
| hospital\_branch | doctor | 1..\* | 1..1 | A hospital branch can have multiple doctors, a doctor can belong to one hospital branch. |

**Entity/Attribute Descriptions**

ward

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| ward\_id | int(11) | unique ward identification |
| department\_type | varchar(30) | type of ward |

doctor

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| doctor\_id | int(11) | unique doctor identification |
| first\_name | varchar(30) | first name of doctor |
| last\_name | varchar(30) | last name of doctor |
| phone\_number | double | doctor’s phone number |
| fk\_hospital\_id | int(11) | foreign key for hospital id |
| fk\_ward\_id | int(11) | foreign key for ward id |

nurse

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| nurse\_id | int(11) | unique nurse identification |
| first\_name | varchar(30) | first name of nurse |
| last\_name | varchar(30) | last name of nurse |
| phone\_number | double | nurse’s phone number |
| fk\_hospital\_id | int(11) | foreign key for hospital id |
| fk\_ward\_id | int(11) | foreign key for ward id |

hospital\_branch

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| hospital\_id | int(11) | unique hospital identification |
| hosp\_name | varchar(30) | hospital branch name |
| phone\_number | double | hospital’s phone number |
| address | varchar(30) | address of hospital branch |
| bed\_number | int(11) | number of beds |

prescription

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| prescription\_id | int(11) | unique prescription id |
| pres\_name | varchar(30) | name of prescription |

visit

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| visit\_id | int(11) | unique nurse identification |
| assigned\_room | int(11) | room number |
| reg\_date | datetime | date of registration |
| fk\_patient\_id | int(11) | foreign key for patient id |
| fk\_bill\_id | int(11) | foreign key for bill id |
| fk\_prescription\_id | int(11) | foreign key for prescription id |

patient\_bill

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| bill\_id | int(11) | unique bill identification |
| no\_of\_days | double | number of days at hospital |
| room\_charge | double | amount charged for room |
| operation\_charge | double | amount charged for operation |
| prescription\_charge | double | charge for prescription |

patient

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| patient\_id | int(11) | unique patient identification |
| first\_name | varchar(30) | first name of nurse |
| last\_name | varchar(30) | last name of nurse |
| date\_of\_birth | datetime | patient’s date of birth |
| phone\_number | double | patient’s phone number |
| gender | varchar(30) | gender of patient |
| address | varchar(50) | patient’s address |
| fk\_emerg\_cont\_id | int(11) | emergency cont. foreign key |
| fk\_hospital\_id | int(11) | foreign key for hospital id |

emergency\_contact

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| emerg\_cont\_id | int(11) | unique contact identification |
| first\_name | varchar(30) | first name of contact |
| last\_name | varchar(30) | last name of contact |
| phone\_number | double | contact’s phone number |
| relation | varchar(30) | relationship to patient |

patient\_staff

|  |  |  |
| --- | --- | --- |
| attribute | data type | description |
| staff\_id | int(11) | unique staff info identification |
| blood\_pressure | varchar(30) | patient’s blood pressure |
| heart\_rate | double | height of patient |
| weight | double | patient’s weight |
| fk\_patient\_id | int(11) | foreign key for patient id |
| fk\_doctor\_id | int(11) | foreign key for doctor id |
| fk\_nurse\_id | int(11) | foreign key for nurse id |

**List of Questions**

1. How many branches are there and where are they located?

This would help locate the nearest branch and their bed count.

SELECT\*FROM hospital\_branch;

1. What room is (name of specific patient) staying in?

By joining the visit table that has all the information regarding a patient’s visit with the patient profile table, a query can locate the room that someone is living in for whatever purpose, such as a visiting family member or mail.

SELECT visit.assigned\_room AS Room, CONCAT(patient.last\_name, ", ", patient.first\_name) AS `Name`

FROM patient AS patient

INNER JOIN visit ON patient.patient\_id = visit.fk\_patient\_id

WHERE patient.first\_name = 'Amanda';

1. Which wards have openings and where do specific doctors work?

By left joining the doctor table, openings at different wards are revealed, indicating where staff are needed.

SELECT ward.department\_type AS Ward, CONCAT(doctor.last\_name, ", ", doctor.first\_name) AS `Name`

FROM ward

LEFT JOIN doctor ON ward.ward\_id = doctor.fk\_ward\_id;

1. What is the average operation charge for a patient?

Taking the average operation charge and maximum operation charge gives a general sense of how much they should expect to pay for an operation at a hospital.

SELECT AVG(operation\_charge) AS AvgOperationCharge

FROM patient\_bill;

What is the most expensive procedure?

SELECT MAX(operation\_charge) AS operation\_max

FROM patient\_bill;

1. What are the heights of different patients, ordered from least to greatest?

Knowledge of the heights of different patients may be relevant when it comes to medication control.

SELECT fk\_patient\_id AS PatientID, MIN(height) AS MinHeight

FROM patient\_staff

GROUP BY fk\_patient\_id

ORDER BY MinHeight;

1. Which patients are located on the second floor?

This is just a basic query to find which patients are living on a specific floor. It may be useful when sending floor-wide alerts or notifications.

SELECT visit.assigned\_room AS Room, CONCAT(patient.last\_name, ", ", patient.first\_name) AS `Name`

FROM patient AS patient

INNER JOIN visit ON patient.patient\_id = visit.fk\_patient\_id

GROUP BY patient\_id

HAVING visit.assigned\_room > 200 AND visit.assigned\_room < 300;

1. Which patient paid the most for their operation in descending order?

This may be useful for accounting and record-keeping purposes. The query shows operation costs in decending order, which serves a similar purpose to querying for the maximum operation charge.

SELECT bill.operation\_charge AS OperationCharge, CONCAT(patient.last\_name, ", ", patient.first\_name) AS `Name`

FROM patient AS patient

INNER JOIN patient\_bill AS bill ON patient.patient\_id = bill.fk\_patient\_id

ORDER BY operation\_charge DESC;

1. Which patient has stayed the longest?

Knowing the patient who stayed the longest can show how effective or ineffective a treatment is, and if anything can be done to improve their condition.

SELECT bill.no\_of\_days AS NumberOfDays, CONCAT(patient.last\_name, ", ", patient.first\_name) AS `Name`

FROM patient AS patient

INNER JOIN patient\_bill AS bill ON patient.patient\_id = bill.fk\_patient\_id

ORDER BY no\_of\_days DESC

LIMIT 1;

1. How can patients and relevant parties be notified of a patient’s location change?

If a hospital branch determines that another branch would have more space to accommodate certain patients, the patient and relevant parties can be notified by isolating their contact information. A subquery filters out the patients that can be moved from the smaller branches to a larger facility.

SELECT CONCAT(patient.last\_name, ", ", patient.first\_name) AS `PatientName`, patient.phone\_number AS PatientNumber, CONCAT(econt.last\_name, ", ", econt.first\_name) AS EConName, econt.phone\_number AS EConNumber

FROM patient AS patient

JOIN emergency\_contact AS econt ON patient.fk\_emerg\_cont\_id = econt.emerg\_cont\_id

WHERE fk\_hospital\_id IN

(SELECT hospital\_id FROM hospital\_branch WHERE bed\_number < 800);

**Closing Section**

This project took a good amount of work. Not only was deciding on an idea for a database difficult, but actually filling out the data for each entity was more complicated than anticipated. Writing the SQL script was the easier part of the project and planning the structure out was definitely the most difficult. There are a lot of moving parts to a database; one entity can be linked to another, entities may have redundant information that can be written into another, and every entity encompasses a large amount of information. It is very clear now that databases are more efficient than spreadsheets; once the basic framework is set, data can be easily found, and edited when needed. I genuinely enjoyed creating and editing my own database and look forward to working with more in the future.