Prof. Namchul Shin

Database Project

Hospital Management System



Submitted by:

Ligia Galvão

Meenal Sawant

Rawaf Alsehli

Table of Contents

1. Introduction
2. Problem Description and Solution
3. Defining Entities and Attributes
4. Entity Relationship Diagram
5. Relational Database
   1. Logic Design
   2. Relational Design
6. Queries
7. Example Report
8. **Introduction**

River Edge Hospital located in New Jersey, Bergen county. They offer a comprehensive range of services to patients in a caring, culturally sensitive environment, accessing all of the specialties and resources. River Edge Hospital is the only care facility in the town of River Edge that is promised in providing the best medical care to their patients and striving to meet the needs of its local residents and be part of the North Bergen community. River Edge Hospital has 300 staff members with over 100 patients are admitted and registered every day. As a small local hospital that has limited staff members and large incoming patients daily maintain their database system is mandatory.

1. **Problem Description and Solution**

Hospitals are key institutions they play a major role in a human’s life. Therefore hospitals should be able to provide the best medical services and the best medical facility to their patients. Providing an efficient service delivery will eventually lead to a happy society. As a result, hospitals need a system that will enable their management to make effective and efficient decisions. For a hospital to able to provide the best medical assistance, the management of the hospital must be disciplined, well-versed in its technique in providing services. Through designing and constructing a user-friendly, easy to use and reliable database system, records of inpatient and outpatient will be easily tracked without being overlapped and can be used by all hospital staff and practitioners. While if the hospital managers are still operating on the manual system, records will be maintained on paper, which is not very efficient or reliable, this system showed a lot of setbacks and problems to medical practitioners and patients throughout the years. Also, this system process is very time consuming that will not be helpful in an environment where time is extremely valuable. The large number of patients, doctors, and hospital staff members make data processing more crucial to be organized in a system that will make it easy to access. The main goal of automating hospital systems is to minimize the paperwork as minimum as possible. It will also help in maintaining the patient’s information, arranging doctors & nurses’ schedule, accessing patient’s records to give the best diagnosis. Managing the database of a hospital can also help patients to easily access their medical files, lab results and all the information that doctors provide about their case and diagnosis.

1. **Defining Entities and Attributes**

An Entity Relationship Model (ERM) is the result of systematic analysis to define and describe what is important to process in an area of business. However, it does not define the business processes; it only presents a business data schema in graphical form. Also, ERM defines a data or information structure implemented in a database, typically a relational database. Through graphic visualization, Entity Relationship diagram shows the various entities and relations between entities, which can be described as:

• 1:1 (One to One relationship)

• 1: N (One to Many relationships)

• N:M (Many to Many relationships)

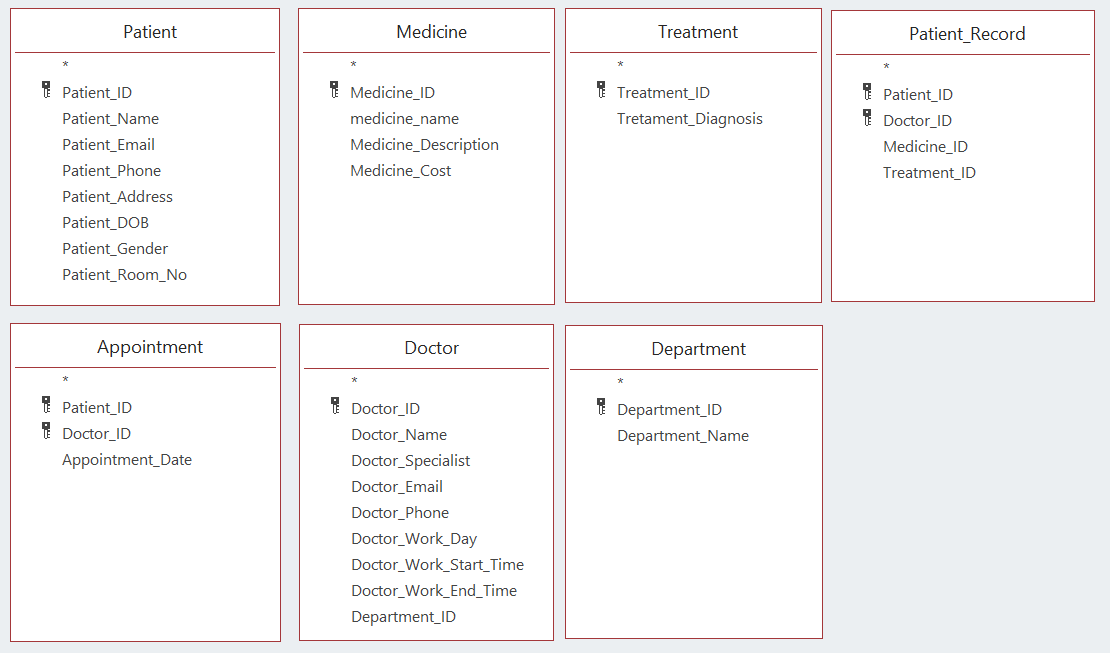
Besides these, ER model shows attributes (columns), entities, and identifiers for each table. As an ERM well-drawn are more likely to succeed in business, the group dedicated several hours to analyze and creating an ER model that most represent the database relational from a hospital company. In this section, there is all the information about the Entity Relationship Model.

* **Entities, Attributes, and Identifiers**

To begin the process, the following entity classes and related attributes were identified:

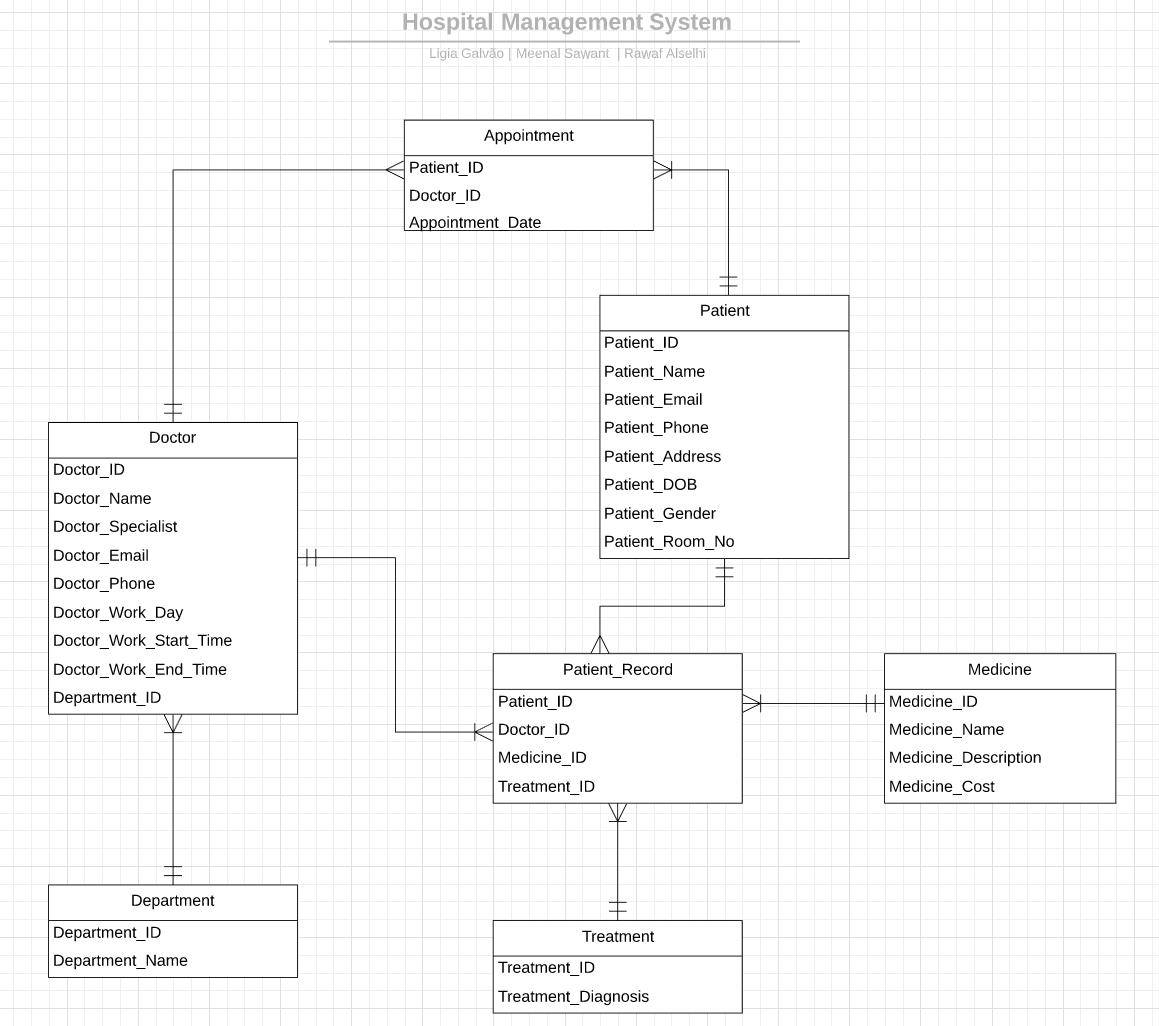
* **PATIENT**: All patient related information who have an appointment in a hospital. The following entity types were identified for Patient: Patient\_ID, Patient\_Name, Patient\_Email, Patient\_Phone, Patient\_Adress, Patient\_DOB, Patient\_Gender, Patient\_Room\_No. Patient\_ID is the identifier for this entity class.
* **DOCTOR**: Doctor related information responsible for each appointment. The following entity types were identified for Doctors: Doctor\_ID, Doctor\_Name, Doctor\_Specialist, Doctor\_Email, Doctor\_Phone, Doctor\_Work\_Day, Doctor\_Work\_Start\_Time, Doctor\_End\_Time, Department\_ID. Doctor\_ID is the identifier for this entity class.
* **APPOINTMENT**: All information about the appointments. The following entity types were identified for Appointment: Patient\_ID, Doctor\_ID, Appointment\_Date. Patient\_ID and Doctor\_ID are the identifiers for this entity class.
* **MEDICINE**: All information about medicines managed in each appointment. The following entity types were identified for Medicine: Medicine\_ID, Medicine\_Name, Medicine\_Description, Medicine\_Cost. Medicine\_ID is the identifier for this entity class.
* **TREATMENT**: All information about treatment managed by the doctor in each appointment. The following entity types were identified for Treatment: Treatment\_ID, Treatment\_Diagnosis. Treatment\_ID is the identifier for this entity class.
* **PATIENT\_RECORD**: All information about the agenda of patient. The following entity types were identified for Patient\_Record: Patient\_ID, Doctor\_ID, Medicine\_ID, Treatment\_ID. Patient\_ID and Doctor\_ID are the identifier for this entity class.
* **DEPARTMENT**: All information about the department of each hospital employee. The following entity types were identified for Department: Department\_ID, Department\_Name. Department\_ID is the identifier for this entity class.

Below there are the various entity classes and attributes to be included in the ER diagram:

  
*Figure 1*: Classes and Attributes from Hospital Management System

1. **Entity Relationship Diagram**

After the identification of the various entity types and attributes, the next step is to determine how each of the entity types are related to each other. Examining the entities described above, the following relationships were identified for the entities:

*Figure 2*: Entity Relationship Model Diagram

* **1:1 (One to One relationship)**

A one-to-one relationship is a type of cardinality that refers to the relationship between two entities A and B in which one element of A may only be linked to one element of B, and vice versa. For instance, think of A as the set of all countries, and B as the set of all their capital city. Any country from A can and must have only one capital city from B, and vice versa.

In this database, no relationship is a one to one relationship.

* **1: N (One to Many relationships)**

A one-to-many relationship is a type of cardinality that refers to the relationship between two entities A and B in which an element of A may be linked to many elements of B, but a member of B is linked to only one element of A. For instance, think of A as mother, and B as kids. A mother can have many kids, but a kid can only have one mother.

* PATIENT has a one to many relationships with APPOINTMENT. This relationship is necessary because at least one patient can make more than one appointment.
* DEPARTMENT has a one to many relationships with DOCTOR. This relationship is necessary because at least one department can be of a doctor employee.
* DOCTOR has a one to many relationships with APPOINTMENT. This relationship is necessary because at least one doctor can be more than one appointment.

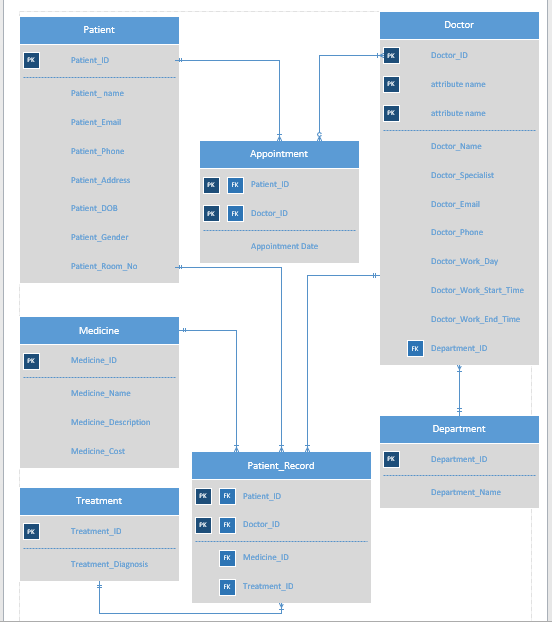
Due to the fact that many medicines can be prescribed to many treatments and many doctors can be many appointments, it was introduced the additional entity PATIENT\_RECORD. These relationships are described below:

* TREATMENT has a one to many relationships with PATIENT\_RECORD: This relationship is necessary because at least one treatment have more than one medicine.
* DOCTOR has a one to many relationships with PATIENT\_RECORD. This relationship is necessary because at least one doctor can make more than one appointment.
* PATIENT has a one to many relationships with PATIENT\_RECORD. This relationship is necessary because many patients can receive more than one treatment.
* MEDICINE has a one to many relationships with PATIENT\_RECORD. This relationship is necessary because many medicines can be prescribed more than one treatment.
* **N:M (Many to Many relationships)**

A many-to-many relationships is a type of cardinality that refers to the relationship between two entities A and B in which A may contain a parent instance for which there are many children in B and vice versa. For example, think of A as Classes, and B as Students. A Class can have several Students, and a Student can be registered in several Classes.

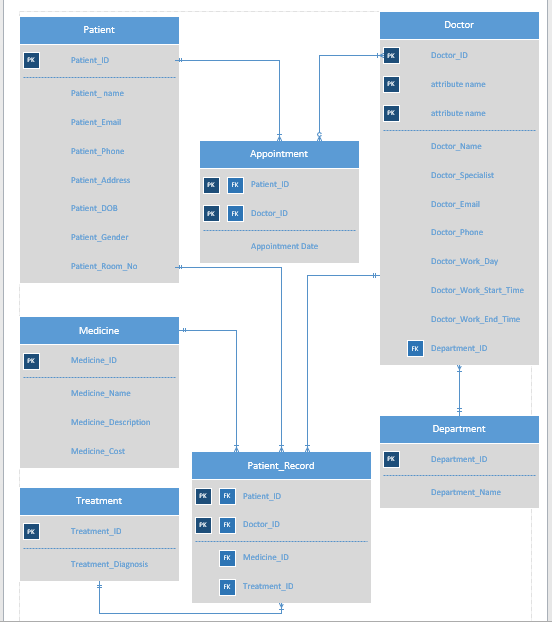
As was described in the section before, we introduce the additional entities PATIENT\_RECORD. However, in this database, no relationship is a many to many relationship.

1. **Relational Database**
2. **Logical design**

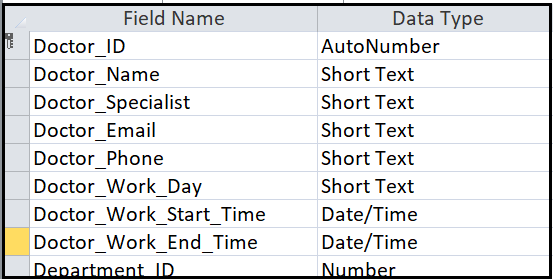
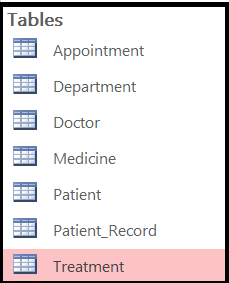
****

*Figure* 3: Logical Design

1. **Relational Design**

The Relational database represents the relation between tables. Primary key uniquely identifies the records the row of that table. The advantage of the relational model are insert, select, modify and delete these operations can be performed in Relational Model and it minimizes the data redundancy issues.

In this project, we have created seven tables. Below is the screenshot of tables and data types :



**The relational design:**

**Patient**(Patient\_ID(PK), Patient\_Name, Patient\_Email, Patient\_Phone,Patient\_Address, Patient\_DOB, Patient\_Gender, Patient\_Room\_No)

**Appointment**(Patient\_ID(PK, FK),Doctor\_ID(PK, FK), Appointment\_Date)

**Doctor**(Doctor\_ID(PK), Doctor\_Name, Doctor\_Specialist, Doctor\_Email, Doctor\_Phone, Doctor\_Work\_Day, Doctor\_Work\_Start\_Time, Doctor\_Work\_End\_Time, Department\_ID(FK))

**Department**(Department\_ID(PK), Department\_Name)

**Treatment**(Treatment\_ID(PK), Treatment\_Diagnosis)

**Medicine**(Medicine\_ID(PK), Medicine\_Name, Medicine\_Description, Medicine\_Cost)

**Patient\_Record**(Patient\_ID(PK, FK), Doctor\_ID(PK, FK), Medicine\_ID(FK), Treatment\_ID(FK)

**Foreign Keys and Constraints**:

A foreign key links two tables in database. Foreign key of one table is a primary of another table. Referential integrity constraints is based on the concept of Foreign Keys. Referential integrity constraint ensures that the values of one column in a table are valid based on values in another table. However, that key element in child table (FK) must exist in the parent table(PK).

For all the above relationships, We have enforced Referential Integrity to avoid the loss or inadvertent updating of data in the tables. Primary key in the table which are referenced in other tables as foreign keys have this integrity enforced. Also, we have applied Cascade Update, Cascade Delete so that whenever we update/delete the data in the parent table, the respective foreign key in the referenced table would get updated. It is shown in figure 4.

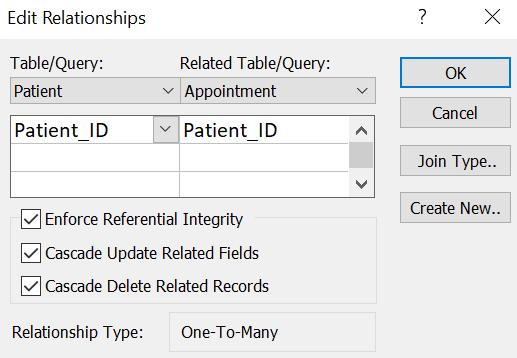
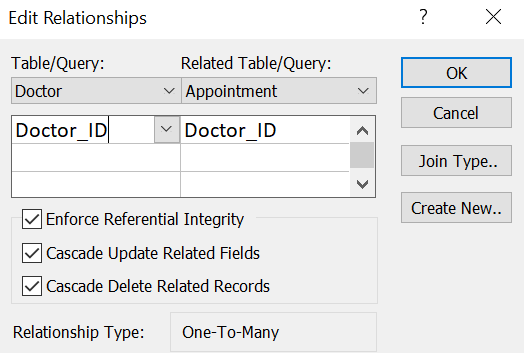


Figure 5: Constraints

There are few exceptions, like Department\_ID in Doctor and Nurse table has not enforced Cascade Update/Delete, so that deletion of Department does not delete the Doctor and Nurse. It is shown in figure 5as below:

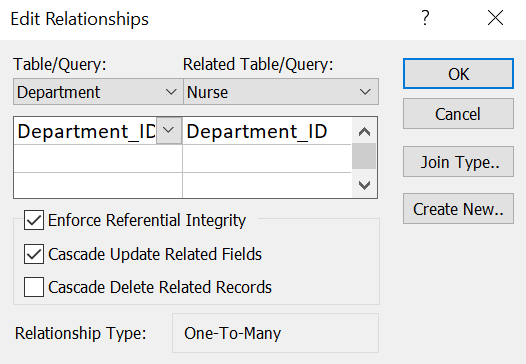
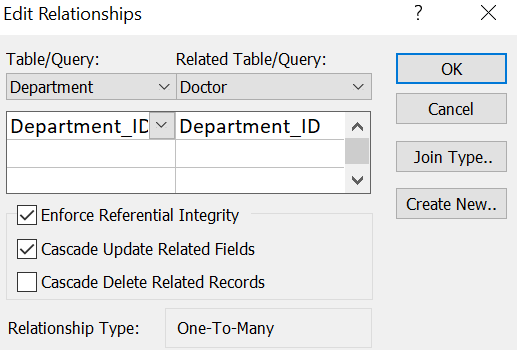
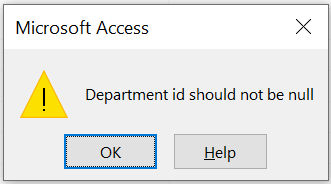


Figure 5: Department\_ID Constraints

We have also added validation rule ‘Is not NULL’ for several fields in our tables where we wanted to enforce no null values for the field. Example: Department\_ID in the Doctor table.



1. **Queries**
2. Find all Doctors who in the Pediatric department:

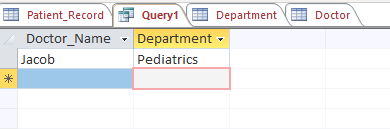
SQL:

SELECT Doctor.Doctor\_Name, Department.Department\_name

FROM Doctor INNER JOIN Department ON Doctor.Department\_ID = Department.Department\_ID

WHERE Department.Department\_Name = "Pediatrics";

Result:



1. Number of diagnoses to Polen Allergy:

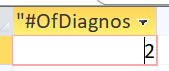
SQL:

SELECT Count (Treatment\_Diagnosis) AS "#OfDiagnosis"

FROM Treatment

WHERE Treatment\_Diagnosis = "Polen allergy";

Result:

****

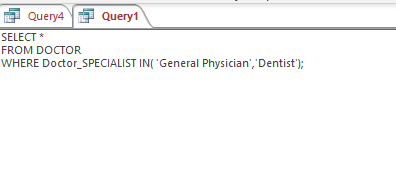
1. Number of doctors who their specialist is either “ General Physician” or “Dentist”:

SQL:

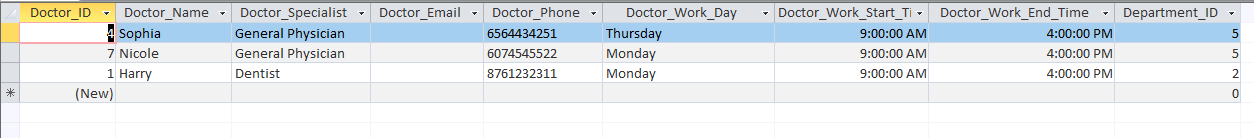
SELECT \*

FROM DOCTOR

WHERE Doctor\_SPECIALIST IN( 'General Physician','Dentist');



Result:



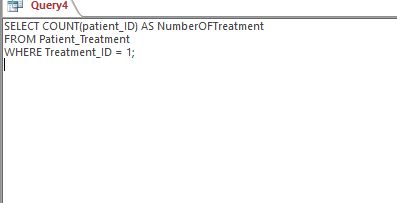
1. Number of patients who got a root canal treatment:

SQL:

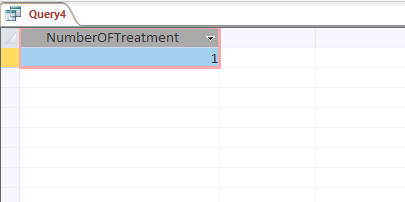
SELECT COUNT(patient\_ID) AS NumberOFTreatment

FROM Patient\_Treatment

WHERE Treatment\_ID = 1;

****

Result:

****

1. **Example Report**

* Sample report of patients information:

