2021

ER Model Design

ACADEMIC PROJECT

ASENATH DANDE

Table of contents

Table of Figures

Table of Tables

Entity Relationship Diagram for a hospital database

ER diagrams help convert functional requirements into high-level conceptual data models (Elmasri and Navathe, 2015) that can support the understanding of technical and non-technical stakeholders. An explanation of the requirements and the ER diagram for the hospital database is provided below.

The requirements for the hospital database

The following requirements have been established for the hospital database:

- The hospital has multiple doctors. Each doctor has at least one patient. Each patient has visited at least one doctor and could visit the same doctor several times.
- Each patient must have at least one case and can have multiple cases, and each case can have multiple appointments.
- A case may or may not require a prescription as it depends on the diagnosis.

The design

ER diagram uses entities representing real-world objects, and relationships that characterize relations between them. The ER diagram in Figure 1 depicts the model built for the hospital database system. Using Tables 1 and 2, the report attempts to effectively convey the ER model's design. Table 1 provides the details of the entities identified for the hospital use case, and Table 2 explains the relationships between the entities. Table 1 provides the details of the six entities, the associated real-world objects, their entity types and the reasoning why the entities were defined as strong or weak. Table 2 lists the relationship name, the entities it relates, and the type of participation and cardinality ratios. The entities can either exhibit total or partial participation and the cardinality ratios have been ensured to not contain any many-to-many relations.

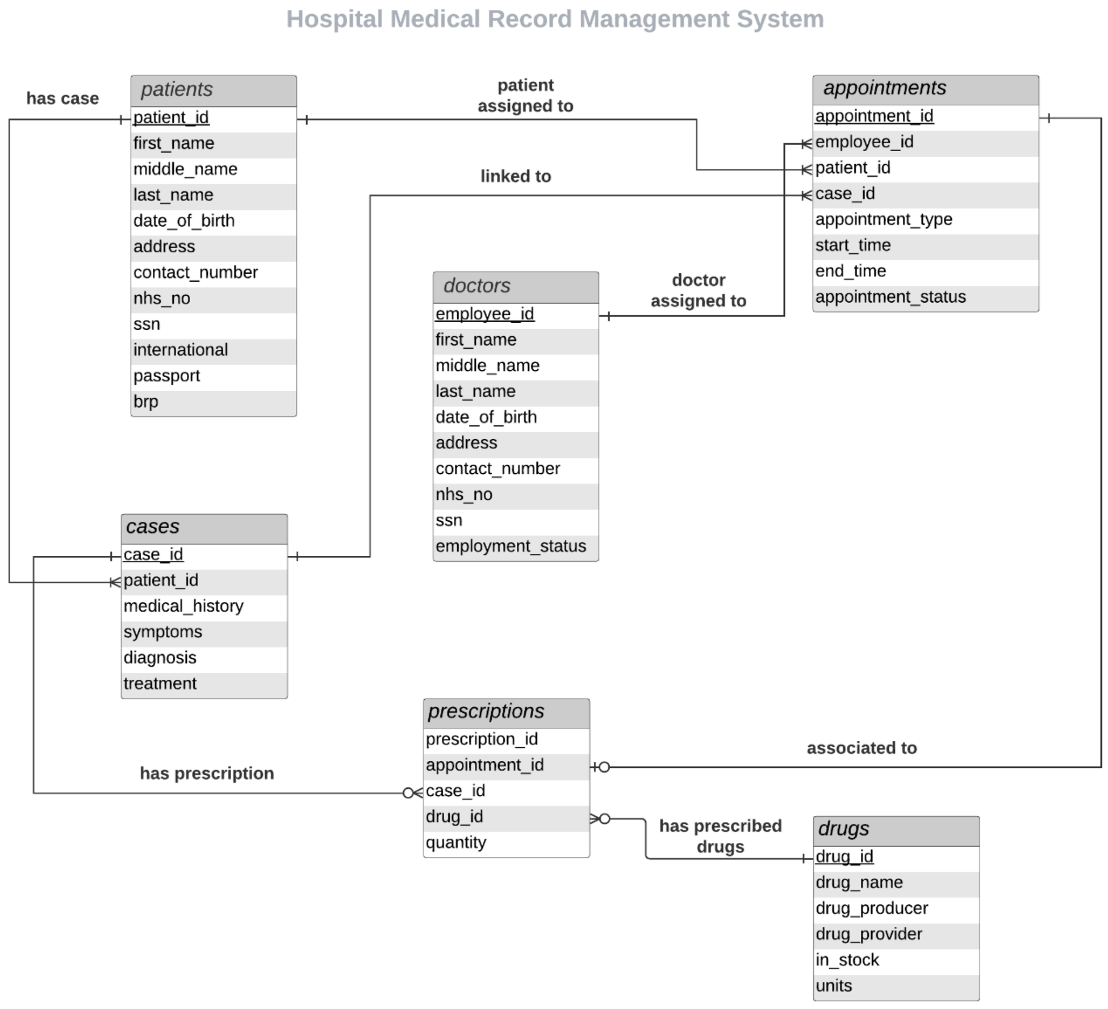

Figure 1: ER Diagram for Hospital Medical Record Management System

Entity	Real world object	Entity type	Reasoning for entity type definition
patients	Patients that visit the hospital.	Strong	Each patient can be identified individually.
doctors	Doctors who are working for the hospital.	Strong	Each doctor can be identified individually.
cases	Cases hold the patient’s information and can be considered as their medical records.	Strong	Each case can be identified individually however, a case is created only after the patient has been created.

appointments	Appointment is a conceptual object that links doctors, patients and cases.	Strong	Each appointment can be identified separately.
drugs	Drugs that the hospital's pharmacy stocks and sells.	Strong	Each drug can be identified separately.
prescriptions	Prescriptions list the details of the drugs the doctor prescribed to the patient.	Weak	Each prescription cannot be identified separately as each prescription can have multiple drugs. It needs case, appointment and drug details to uniquely identify one record.

Table 1: Describing the entities

Relation	**Entities it relates**	**Participation**	**Cardinality**
has case	*patients* *cases*	Total participation from both ends	1: N (*patients*: *cases*)
patient assigned to	*patients* *appointments*	Full participation from both ends	1: N (*patients*: *appointments*)
doctor assigned to	*doctors* *appointments*	Full participation from both ends	1: N (*doctors*: *appointments*)
linked to	*cases* *appointments*	Full participation from both ends	1: N (*cases*: *appointments*)
has prescription	*cases* *prescriptions*	Partial participation from *cases* and full participation from *prescriptions*	1: N (*cases*: *prescriptions*)
has prescribed drugs	*drugs* *prescriptions*	Partial participation from *drugs* and full participation from *prescriptions*	1: N (*drugs*: *prescriptions*)
associated to	*appointments* *prescriptions*	Partial participation from *appointments* and full participation from *prescriptions*	1: 1 (*appointments*: *prescriptions*)

Table 2: Describing the relationships amongst entities

The details of the attributes can be found in the ER diagram from Figure 1. The underlined attribute within an entity is the primary key as it helps to identify each record uniquely. The attributes (1) contact_number in *patients* and *doctors* tables, (2) diagnosis and treatment in *cases* table, and (3) drug_provider in *drugs* table are multivalued attributes meaning they can hold multiple values separated by a delimiter. For example, a person can have two contact numbers. All the other attributes are single-valued. The first, middle, and last name attributes originated from one composite attribute called name. Composite attributes are split into single-valued attributes for ease of handling.

Although, the cardinality ratio for *appointments* and *prescriptions* is 1:1, while implementing in the database, multiple entries for the same appointment will be allowed as the prescription table creates a separate record for each drug.

Solving the many-to-many relation between patients and doctors

As per the requirement, a many-to-many relationship exists between *patients* and *doctors*, and implementing such a relationship is not ideal. Therefore, to overcome the associated issues, a new entity named *appointments* is introduced to split the many-to-many relationship into two 1-to-many (1: N) which are: (1) 1: N (*patients*: *appointments*), and (2) 1: N (*doctors*: *appointments*). Nevertheless, the *patients* and *doctors* still have a many-to-many relationship through the *appointments* table.

Potential implementation issues

As discussed above, several relationships require full participation from both entities. However, this cannot be realized in the database as it results in an issue where the database will refuse to make a new entry in one table before it exists in the other table, thereby never inserting the record. Therefore, to overcome the block, the database tables are implemented with partial participation from one side or both sides, and triggers are used to handle the total participation requirement.

References

Elmasri R. and Navathe S.B. (2015). Fundamentals of Database Systems. Seventh edition. Boston: Pearson, pp.59.