**Database modeling**

Database modeling is an important part of the database design process. It provides a structured, systematic approach that supports the development of well-structured and high performance databases. If databases are well designed potential problems associated with data redundancy and poor performance due to ill-structured links between tables can be avoided or minimized.

Database modeling is utilized as just part of the design process of database systems. It is undertaken once the user needs have been identified and before and implementation is undertaken, that is before any consideration is given to the choice of DBMS or hardware. This paper focuses on the conceptual modeling stage of the database design process (in particular the creation of entity-relationship diagrams (ERDs)), and also implementing the design (ERD) using a relational database. Development of ER diagrams is a skill that can be developed through experience. Skilled database modelers are highly valued people within system development project-teams.

Figure 1 shows a sample entity relationship diagram that represents the relationships between patients admitted to a hospital and the doctors who treat them. The important entities in this situation are the patients and doctors. The diagram indicates that a doctor may admit many patients to the hospital, and just one doctor admits each patient. A doctor may treat many patients in the hospital, and a patient may be treated by a number of different doctors while they are in the hospital. Details of the treatment (date, time and results) are recorded for each treatment.

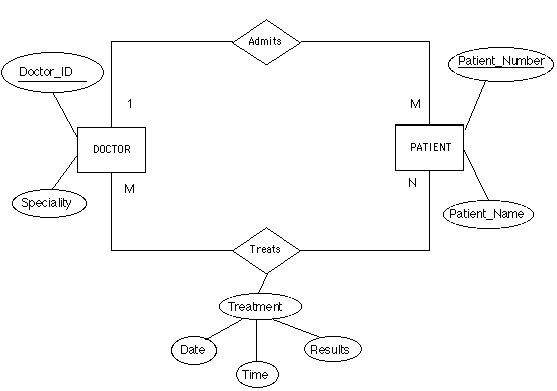


Figure 1: A sample entity relationship diagram that represents the relationships between patients admitted to a hospital and the doctors who treat them.

Let us now systematically look at each of the components that make up an ERD and how they are used.

**ER Model Concepts**

ER model has three main concepts:

􀂄 Entities (and their entity types and entity sets)

􀂄 Attributes (simple, composite, multivalued)

􀂄 Relationships (and their relationship types and relationship sets)

**Entities**

Entity may be an object with

* a physical existence for example student.
* Entity may be an object with a conceptual existence for example a company, a job, a university course.

An entity is a 'thing' of interest to an organisation about which data is to be stored. Example entities for a school database system could include students, teachers, classrooms, etc. Output from systems (e.g. reports) and individual objects (like the principal of a school) are not valid entities in an ERD. (Hint: when reading a description of a situation to be represented using an ERD entities can often be identified by paying attention to nouns).

Entities are represented in an ERD using a rectangle that is labeled by writing the name of the entity in the center in upper-case letters. In Figure 1 there are two entities they are DOCTOR and PATIENT.

***Instances***

An occurrence of an entity is called an instance. A database table ultimately holds each instance of an entity as records, or rows in a table. An example of an instance of the entity DOCTOR could be Dr. Peter Benjamin and examples of a instances for PATIENT could be Mrs. Helen Bowles or Mr. Paul Pappos.

**Attributes**

Attributes are properties used to describe an entity.

For example an STUDENT entity may have the attributes ID,Name, Address, Sex, BirthDate.

The attribute values that describe each entity become a major part of the data stored in the database.

A specific entity will have a value for each of its attributes.

For example a specific Student entity may have Name=‘Suda Jaidee', ID='123456789-0', Address ='731, Muang, Khon Kaen,40002', Sex='M', BirthDate='09-JAN-55‘.

Each attribute has a value set (or data type) associated with it – e.g. integer, string, …

Characteristics of entities are termed 'attributes' of an entity. Attributes of a STUDENT may be name, date of birth, address, phone number, year advisor etc.

Attributes of entities are represented in an ERD using an ellipse that has the name of an attribute in lower-case (again a convention) letters inside. Each ellipse is connected to the relevant entity using a straight line. In Figure 1 the attributes of a DOCTOR are the Doctor\_ID and Specialty. The attributes of a patient are Patient\_Name and Patient\_Number.

Types of Attributes

* 􀂇 Simple or Atomic Attribute
* 􀂇 Composite Attribute
* 􀂇 Single-Valued Attribute
* 􀂇 Multivalued Attribute
* 􀂇 Stored Attribute
* 􀂇 Derived Attribute

**Simple or Atomic attribute**

Atomic attribute is an attribute that is not divisible. Example SSN or Sex.

**Composite attribute**

Composite Attributes can be divided into smaller subparts,which represent more basic attributes with independent meanings.

The attribute may be composed of several components. For example:

Address(House#, Street, City, State, ZipCode, Country), or

Name(FirstName, MiddleName, LastName).

**Single valued vs. Multivalued**

Single valued attribute has a single value for a particular entity.

Multivalued attribute has multiple values.

**Stored vs. Derived attributes**

Derived attribute: its attribute values can be determined from related attribute.

Age attribute is said to be derivable from the Birth\_date attribute, which is called stored attribute.

**Key Attribute**

Each entity must have a unique identifier, called a key attribute. The key attribute must be unique for the life of the entity and must be original for each instance. Key attributes are shown in ERD by underlining the name of the appropriate attribute. In Figure 1 Doctor\_ID is the key attribute of the entity DOCTOR and Patient\_Number is the key attribute of the entity PATIENT. When an identifier is made up of two or more attributes it is called a composite key.

**Relationships**

A relationship in an ERD describes the associations between entities that are of interest to the organisation and users. Relationships are shown in an ERD by a diamond with a description of the relationship written (in lower case letters) in the diamond. Lines connect the diamond to the related entities. In Figure 1 there are two relationships, 'admits' and 'treats'. (Hint: when reading a description of a situation to be represented using an ERD relationships can often be identified by paying attention to verbs).

The number of entities in a relationship identifies the 'degree' of a relationship. The degree of a relationship can be unary, binary or ternary. Examples of particular types of relationships are shown below in Figures 2, 3 and 4.

**Binary Relationships** (involving two entities)

A binary relationship describes a relationship that exists between two entities. Binary relationships are the type most frequently found in ERDs. See Figure 2 for an example of a binary relationship that describes the relationship between employees in an organisation and their assignment to projects. Both of the relationships in Figure 1 are binary.

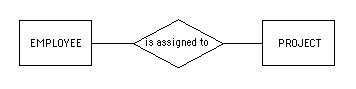


Figure 2: An example of binary relationship

**Unary Relationships** (involving one entity)

A unary relationship describes a relationship that exists involving just one entity, or the relationship between instances within the same entity. An example is the description of the marriage relationship between people. The notation used for this unary relationship is shown in Figure 3. There are no unary relaionships in Figure 1.

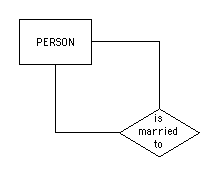


Figure 3: An example of a unary relationship

**Cardinality**

Cardinality in ERDs expresses the specific number, or range of numbers, of entities involved in a relationship. Cardinality is usually describes as 'one-to-one' (1:1), 'one-to-many' (1:M) or many-to-many' relationships (M: N). Cardinality is indicated on an ERD using a numeral 1 or the letters M or N on the relevant relationship lines. For a relationship involving a 'many' an 'M' or 'N' includes a range of numbers from one to infinity. A relationship that involves a '1' includes a range of only one.

Figure 1 shows both a one-to-many (admits) relationship and a many-to many relationship (treats). Figure 5 shows the part of Figure 1 that forms the 'admits' relationship. When describing the 'admits' relationship it would be said that one doctor admits a patient to the hospital, but a doctor can admit one or more patients.

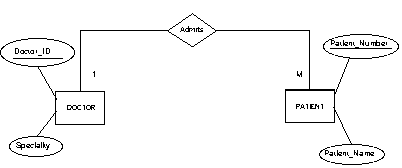


Figure 5: The 'admits' relationship from Figure 1

Figure 6 shows the part of figure 1 that forms the 'treats' relationship. When describing the 'treats' relationship it would be true to say that a patient can be treated by one, or a number of doctors (many) while in hospital, and that doctors treat one or many patients in the hospital.

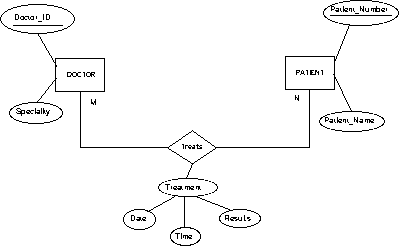


Figure 6: The 'treats' relationship from Figure 1

Constraints on Relationships

Constraints on Relationship Types

(Also known as **ratio constraints**)

􀂄 **Cardinality Ratio** (specifies *maximum* participation)

One-to-one (1:1)

One-to-many (1:N) or Many-to-one (N:1)

Many-to-many (M:N)

􀂄 **Existence Dependency** Constraint (specifies *minimum*

participation) (also called participation constraint)

**zero** (optional participation, not existence-dependent)

**one or more** (mandatory participation, existence-dependent)

**Types of relational keys:**

\_ Candidate Key

An attribute or set of attributes that uniquely identifies a row is called a Candidate Key.

\_ Primary Key

The candidate key that you choose to identify each row uniquely is called the Primary Key.

\_ Alternate Keys

A Candidate key that is not chosen as a Primary key is an Alternate key.

\_ Foreign Key

When a primary key one table appears as an attribute in another table. It is called the Foreign key in the second table. A Foreign key is used to relate two tables.