**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Data Base Management System LabAssignment #4**

**Submitted by:**

Suman Malla  
013BSCCSIT047

Sem IV

**Submitted to:**

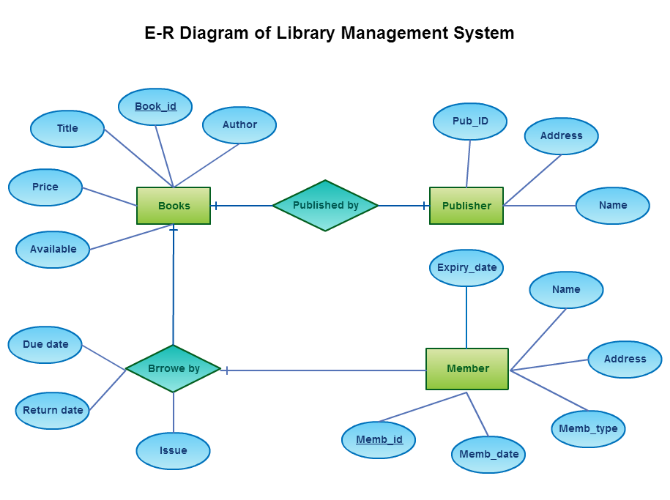
|  |  |
| --- | --- |
| Er. Sanjay Kr. Yadav  Lecturer |  |

**Submitted on: August 10, 2015**

1. **E-R Diagram with one case study**

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering), an entity–relationship model (ER model) is a [data model](https://en.wikipedia.org/wiki/Data_modeling) for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a [database](https://en.wikipedia.org/wiki/Database) such as a [relational database](https://en.wikipedia.org/wiki/Relational_database). The main components of ER models are [entities](https://en.wikipedia.org/wiki/Entities) (things) and the relationships that can exist among them.

Entity–relationship modeling was developed by [Peter Chen](https://en.wikipedia.org/wiki/Peter_Chen) and published in a 1976 paper. However, variants of the idea existed previously, and have been devised subsequently such as super type and subtype data entitiesand commonality relationships.

****

1. **Design**
2. **Functional Design**

Functional Design is a paradigm used to simplify the design of hardware and software devices such as computer [software](https://en.wikipedia.org/wiki/Software) and increasingly, [3D models](https://en.wikipedia.org/wiki/3D_model). A [functional](https://en.wikipedia.org/wiki/Function_%28engineering%29) design assures that each modular part of a device has only one responsibility and performs that responsibility with the minimum of side effects on other parts. Functionally designed modules tend to have low [coupling](https://en.wikipedia.org/wiki/Coupling_%28computer_science%29). The advantage for implementation is that if a software module has a single purpose; it will be simpler, and therefore easier and less expensive, to design and implement.

Systems with functionally designed parts are easier to modify because each part does only what it claims to do.

Since maintenance is more than 3/4 of a successful system's life,this feature is a crucial advantage. It also makes the system easier to understand and document, which simplifies training. The result is that the practical lifetime of a functional system is longer.

1. **Database Design**

Database design is the process of producing a detailed [data model](https://en.wikipedia.org/wiki/Data_model) of a [database](https://en.wikipedia.org/wiki/Database). This [logical data model](https://en.wikipedia.org/wiki/Logical_data_model) contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a [data definition language](https://en.wikipedia.org/wiki/Data_definition_language), which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall [database system](https://en.wikipedia.org/wiki/Database_system). Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the [relational model](https://en.wikipedia.org/wiki/Relational_model) these are the [tables](https://en.wikipedia.org/wiki/Database_table) and [view](https://en.wikipedia.org/wiki/Database_view). In an [object database](https://en.wikipedia.org/wiki/Object_database) the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the [database management system](https://en.wikipedia.org/wiki/Database_management_system) (DBMS).

There are three different type of database design**:-**

* **Conceptual Database Design**

Conceptual design of the database specifies that how the data is normalized and reduce data insufficiency and conflicts.

* **Logical Database Design**

Once the relationships and dependencies amongst the various pieces of information have been determined, it is possible to arrange the data into a logical structure which can then be mapped into the storage objects supported by the [database management system](https://en.wikipedia.org/wiki/Database_management_system). In the case of [relational databases](https://en.wikipedia.org/wiki/Relational_databases) the storage objects are [tables](https://en.wikipedia.org/wiki/Database_table) which store data in rows and columns.

* **Physical Database Design**

The physical design of the database specifies the physical configuration of the database on the storage media. This includes detailed specification of [data elements](https://en.wikipedia.org/wiki/Data_element), [data types](https://en.wikipedia.org/wiki/Data_type), [indexing](https://en.wikipedia.org/wiki/Index_%28database%29) options and other parameters residing in the DBMS [data dictionary](https://en.wikipedia.org/wiki/Data_dictionary). It is the detailed design of a system that includes modules & the database's hardware & software specifications of the system.

1. **Characteristics of the Relation**

Some of the characteristics are show below:-

* **The use of keys**

Each row of data in a table is identified by a unique "key", called the primary key. The primary key is often an automatically incrementing number like 1, 2, 3 4.... etc. Data in different tables can be linked together using keys. The primary key values of one table can be added to rows of a different table, thereby linking those rows together.

## Avoiding data redundancy

In a database design that adheres to the rules of the relational model, each data item, a username for example, is stored only once, that is, in one location. This avoids having to maintain the same data in multiple locations. The duplication of data is called **data redundancy** and this should be avoided in a good database design.

## Constraining the input

Using a relational database you can specify what sort of data a database column is allowed to contain. You can create fields that contain numbers, decimal numbers, small texts, large texts, dates, etc.When defining a database table you supply a type for each column.

## Maintaining data integrity

By setting field properties, by linking tables and by setting constraints you can increase the reliability of your data.

## Rights

Most relational database systems offer a rights structure with which rights can be assigned to different users. Some of the operations that can be allowed or disallowed to a user are SELECT, INSERT, DELETE, ALTER, CREATE, etc. These rights correspond to the operations that can be performed using the Structured Query Language (SQL).

## Structured Query Language (SQL)

In order to actually perform operations on the database, like storing new data, and selecting and altering existing data, SQL queries are used. The Structured Query Language is relatively easy to understand and it allows advanced database operations, such as the selection of linked data from multiple tables with JOIN queries. As previously discussed, SQL is out of scope for this article

## Portability

The relational model is a standard. By adhering to the rules of the relational model you ensure that your data can be transferred between relational database systems relatively easily.

1. **E-R to relational mapping algorithm**

* **Step1:-Mapping of regular entity type**

Create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

* **Step 2:-Mapping of weak entity type**

Create a relation R, and include all simple attributes (or simple components of composite attributes) of W as attributes of R. In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of the identifying relationship type of W. The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

* **Step 3:- Mapping binary 1:1 relation type**

Identify the relations S and T that correspond to the entity types participating in R. Choose one of the relations—S, say—and include as foreign key in S the primary key of T.  Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.

* **Step 4:-Mapping binary 1:M relation type**

Identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.  (WHY?)

* **Step 5:- Mapping binary M:N relation type**

Create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.

* **Step 6: -Mapping binary Multivalued attributes.**

Create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as an attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

* **Step 7:-Mapping of binary N-ary relation types**

Where n > 2,create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.