**ST.XAVIER’S COLLEGE**

MAITIGHAR, KATHMANDU



Database Management System

Assignment #4

Submitted By:

Aabhash Dhakal

013BSCCSIT001

2nd year/ 4th semester

Submitted to:

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav  Lecturer  Department of Computer Science |  |

# ER diagram with case study

# DESIGN

Database systems are designed to manage large bodies of information. These large bodies of information do not exist in isolation. They are part of the operation of some enterprise whose end product may be information from the database or may be some device or service for which the database plays only a supporting role. The design of any information system is divided into two tasks, first the database design and secondly the functional design or as it is sometimes known, operational design.

* 1. **Database Design**

In database design, the necessary structure and organization of the data are established.

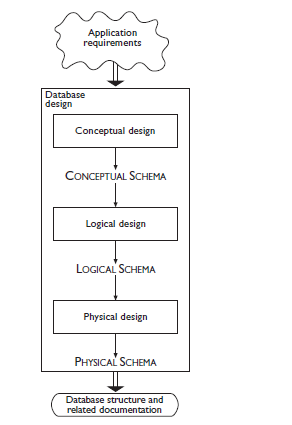


Fig. Phases of database design

* + 1. **Conceptual Design**

The purpose of conceptual design is to represent the informal requirements of an application in terms of a formal and complete description, but independent of the criteria for representation used in database management systems.

The product of this phase is called the conceptual schema. Conceptual models allow the description of the organization of data at a high level of abstraction, without taking into account the implementation aspects.

* + 1. **Logical Design**

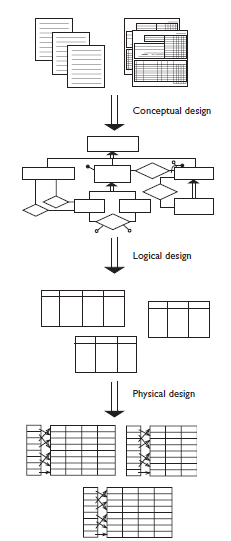
This consists of the translation of the conceptual schema defined in the conceptual phase, into the data model adopted by the database management system available.

The product of this phase is called the logical schema of the database and refers to a logical data model.

A logical model represents data in a way that is still independent of the physical details, although the database management system used for the implementation must be one that supports that data model. In this phase, the designer must also take into account some optimization criteria, based on the operations to be carried out on the data.

* + 1. **Physical Design**

In the physical design phase, the logical schema is completed with the details of the physical implementation on a given DBMS. The product of this phase is called the physical schema.

****

* 1. **Functional Design**

In functional design, the characteristics of the application programs are defined.

# CHARACTERISTICS OF RELATIONS

Relationships represent logical links between two or more entities. In ER schema, each relationship has a unique name and is graphically represented by means of a diamond, containing the name of the relationship, and by lines that connect the diamond with each component.

# ER TO RELATIONAL MAPPING

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into relational model, but an approximate schema can be generated.

There are several processes and algorithms available to convert ER Diagrams into Relational Schema. Some of them are automated and some of them are manual.

# MAPPING OF REGULAR ENTITY TYPES

1. For each entity type E in the ER schema, we create a relation R that includes all the simple attributes of E.

2. Then we add only simple components from any composite attributes in E.

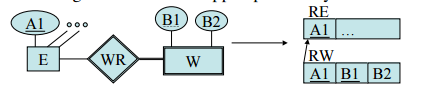
3. We choose one of the key attributes of E to be a primary key of R.



# MAPPING OF WEAK ENTITY TYPES

1. For each weak entity type W with owner type E create a new relation RW that includes all the simple attributes of W as attributes of RW.
2. In addition include a foreign key reference to the key of the translation RE of E.

3. The key of RW will be the key of foreign key together with the mapped partial key from W.



# MAPPING OF BINARY 1-TO-1 RELATION TYPES

To map binary 1:1 relation types there are three possible approaches:

1. Foreign Key approach: Choose one of the relations-say Sand include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.

2. Merged relation option: An alternate mapping of a 1:1 relationship type is possible by merging the two entity yp p t es and the relationship into a single relation. This may be appropriate when both participations are total.

3. Cross-reference or relationship relation option: The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

**4.3 MAPPING OF BINARY 1-TO-N RELATION TYPES**

1. For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.

2. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.

3. Include any simple attributes of the 1: N relation type as attributes of S.

# MAPPING OF BINARY M-TO-N RLEATIONSHIP TYPE

1. For each regular binary M:N relationship type R, create a new relation S to represent R.
2. 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.
3. Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

# MAPPING OF MULTIVALUED ATTRIBUTE

1. For each multivalued attribute A, create a new relation R.
2. 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 of relationship type that has A as an attribute.
3. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

# MAPPING OF N-ARY RELATIONSHIP TYPE

1. For each n-ary relationship type R where n>2 ary relationship type R, where n>2, create a new relation S to represent R.
2. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
3. Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.