**ST. XAVIER’S COLLEGE**

**Maitighar, Kathmandu**



DATABASE MANAGEMENT SYSTEM

theory Assignment #4

**Submitted by:**

Dibash Poudel

013BSCCSIT017

**Submitted to:**

Er. Sanjay Kr. Yadav

Lecturer, St. Xavier’s college

Department of Computer Science

**Date of Submission:**

August 10, 2015

1. **ER diagram with one case study**

### Tips on How to Draw ER Diagrams

Because ER diagrams are simple enough to understand, just about anyone can create them. However, two different ER diagrams describing the same system may still be radically different in terms of their simplicity, completeness, and efficiency at communicating the system. In other words, there are good ER diagrams and there are poor ones.

Because this ER tutorial focuses on beginners below are some tips that will help you build effective ER diagrams:

1. Identify all the relevant entities in a given system and determine the relationships among these entities.
2. An entity should appear only once in a particular diagram.
3. Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.
4. Remove vague, redundant or unnecessary relationships between entities.
5. Never connect a relationship to another relationship.
6. Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams.

**Scenario**

A small accounting firm wants a simple HR application that will help it to keep track of its employees, their positions, allowances, salary scales, and which company vehicles their employees drive.

The application must keep track of all the positions at the firm, the employees filling these positions, the allowances for these positions, the salary scales for these positions, and the company vehicles assigned to these positions.

**ER-Diagram**

****

**2. Design**

**-Functional design**

**Functional Design** is a paradigm used to simplify the design of hardware and software devices such as computer software and increasingly, 3D models. A functional 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.

**Advantage**

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,[[1]](https://en.wikipedia.org/wiki/Functional_design#cite_note-1) 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.

In a system of programs, a functional module will be easier to reuse because it is less likely to have side effects that appear in other parts of the system.

**-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).[[1]](https://en.wikipedia.org/wiki/Database_design#cite_note-1)

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

* Determine the relationships between the different data elements.
* Superimpose a logical structure upon the data on the basis of these relationships

[Database design](http://www.wisegeek.com/what-is-database-design.htm) involves identifying the existing relationships between separate pieces of data and mapping out those relationships in an organized way that makes sense. There are several types of database design: conceptual database design, logical database design, and physical database design. Logical and physical database design are perhaps the most straightforward. Conceptual database design is a bit more ambiguous because during this phase there is no direct work on a [database model](http://www.wisegeek.com/what-is-a-database-model.htm). The process is solely an exercise in the identification of relevant data.

Two principal things that are being identified in conceptual database design are entities and relationships — entities being actual objects in the material world, and relationships being the network of connections linking one entity to another indefinitely. Here arises the central notion of this type of design: the entity-relationship model. This does not feature the overall organization and structure that will be inherent in logical database design; it is, however, a precursor to it.

Relationship cardinalities are an essential part of the entity-relationship model used in conceptual database design. Cardinalities express how regularly an entity experiences a particular relationship with another entity. In the actual model these are denoted by the points at which an entity on the diagram branches out to link with single or multiple entities. Various “attributes” such as names, qualities and quantities associated with the entities and relationships are depicted in the model as well.

**Types of Database Design**

**-conceptual database design**

A conceptual diagram is a visual representation of the ways in which abstract concepts are related. It is used as an aid in visualizing processes or systems at a high level through a series of unique lines and shapes. Conceptual diagrams are widely employed in fields as diverse as business, science, and manufacturing, and they may be created in a variety of ways to suit the project’s needs.

In a typical conceptual diagram, abstract ideas are written as text, enclosed in shapes on a blank background, and connected by a series of arrows or lines. Often a central concept is included at the heart of the diagram and other factors are shown in connection to this central concept. For example, if someone were to create a conceptual diagram to show factors that contribute to getting a good night’s sleep, the diagram would include a box or other shape representing restful sleep at the center. This box would be surrounded by shapes showing various factors, such as exercise and stress that would be connected to the center box with lines or arrows. The lines and arrows indicate visually how secondary factors are related to the central concept of restful sleep.

The versatility of its visual format allows the conceptual diagram to be adapted to almost any situation where a system can be explained through a network of abstract ideas. Such a diagram might be used to show the flow of a manufacturing process, or the interactions of a complex ecosystem. In business, these representations are especially popular because they provide a way for managers to show employees a business strategy in a clear and direct fashion. Conceptual diagrams are used not only for planning, but also for conceptual problem solving — brainstorming solutions to an abstract problem.

When constructing a conceptual diagram, several design factors must be considered before beginning to draw. The first step is to identify the central idea that the diagram intends to communicate. Once the central concept is identified, the other concepts can be prioritized to fit into the available visual space. Effective representations tend to communicate information in a concise way, without confusing the viewer with tangled networks of lines and shapes, so only the most important information is usually included.

The next step is to consider the audience for which that the diagram is intended. Scientists with specialized knowledge might understand a network of highly technical concepts, but a popular audience may not. The goal of the diagram format is to communicate an idea so that the appropriate audience understands it without lengthy explanation.

Although less important than the factors mentioned above, visual style is also a consideration in designing a conceptual diagram. Color, shapes, and overall view can affect the impact of a visual. If symbols are being used, a key should be included to indicate what each symbol represents. Clarity is usually the foremost concern, but certain projects may call for more originality and abstraction in the representation.

**-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.

Each table may represent an implementation of either a logical object or a relationship joining one or more instances of one or more logical objects. Relationships between tables may then be stored as links connecting child tables with parents. Since complex logical relationships are themselves tables they will probably have links to more than one parent.

In an [Object database](https://en.wikipedia.org/wiki/Object_database) the storage objects correspond directly to the objects used by the [Object-oriented programming language](https://en.wikipedia.org/wiki/Object-oriented_programming_language) used to write the applications that will manage and access the data. The relationships may be defined as attributes of the object classes involved or as methods that operate on the object classes.

**-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), types, indexing 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.

The goal of the last phase of database design, physical design, is to implement the database. At this phase one must know which database management system (DBMS) is used. For example, different DBMS's have different names for data types and have different data types. The SQL clauses to create the database are written. The indexes, the integrity constraints (rules) and the users' access rights are defined. Finally the data to test the database is added in.

1. **Characteristics of relation**
2. ER to relational mapping algorithm

Step1: Mapping of regular entity types

Step2: mapping of weak entity type

Step3: Mapping of binary 1 is to 1 relation type

Step4: mapping of binary 1 is to N relation type

Step5: mapping of binary M is to N relation type

Step6: mapping of multi value attributes

Step7: mapping of N-array relation types

5. Mapping of multi value