**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System**

**Lab Assignment #4**

**Submitted by:**

Sazjan Neupane

013BSCCSIT036

**Submitted to:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav**  Lecturer  St. Xavier’s College |  |

**Date of Submission: August 10, 2015**

**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).

* **Conceptual Design**

Once all the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using a high level conceptual data model. This phase is called conceptual design.  
  
The result of this phase is an Entity-Relationship (ER) diagram or UML class diagram. It is a high-level data model of the specific application area. It describes how different entities (objects, items) are related to each other. It also describes what attributes (features) each entity has. It includes the definitions of all the concepts (entities, attributes) of the application area.  
  
During or after the conceptual shema design, the basic data model operations can be used to specify the high-level user operations identified during the functional analysis. This also serves to confirm that the conceptual schema meets all the indenfied functional requirements.

* **Logical Design**

The result of the logical design phase (or data model mapping phase) is a set of relation shcemas. The ER diagram or class diagram is the basis for these relation schemas.  
To create the relation shemas is quite a mechanical operation. There are rules how the ER model or class diagram is transferred to relation shemas. The relation schemas are the basis for table definitions. In this phase (if not done in previous phase) the primary keys and foreign keys are defined.

* **Physical Design**

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.  
  
In parallel with these activities, application programs are designed. The implementation of the programs can start when the database is created and data has been added in.

**Characteristics of Relation**

A database relation is a predefined row/column format for storing information in a relational database. Relations are equivalent to tables.

# Characteristic of relations

# ER to Relational mapping algorithm

## Mapping of regular entity types

* For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes.
* Include only the simple component attributes of a composite attribute.
* Choose one of key attributes of E as primary key of 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.

## Mapping of weak entity types

* For each weak entity type W in the ER schema with owner entity type E,
  + create a relation R, and
  + include all simple attributes (simple component of composite attributes) of W as attributes of R
* n addition, include as *foreign key* attributes of R the primary key attribute(s) of 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.
* It is common to choose the propagate (CASCADE) option for the referential triggered action on the foreign key in the identifying entity relation of the weak entity type, since a weak entity has an existence dependency on its owner entity.
* This can be used for both ON UPDATE and ON DELETE.

## Mapping of binary 1:1 relationship types

* For each binary 1:1 relationship type R in the ER schema, and S and T, the relations that participate in R
  + Choose one relation, for example S, and
  + Include as foreign key in S the primary key of T.
* It is better to choose an entity type with *total participating* in R as the role of S.
* Include all simple attributes (or simple component of composite attributes) of the 1:1 relationship type R as attributes of S.