ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DBMS Lab Assignment #4**

**Submitted by:**

Pankaj Niroula  
013BSCCSIT025

**Submitted to:**

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

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

# Design

**Functional Design**

Functional design is the kind of design that is agreed between software architects and business customers. It describes what the system has to do in a way that both parties can understand and agree. It might describe user interface requirements, it might discuss database access, it might discuss web services or other forms of communication, it might discuss service level agreements or clustering and redundancy. The point is that it should be detailed enough to become the agreed definition of what the system will do.

**Database Design**

Database design is the process of producing a detailed data model of a database. This 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, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

**Conceptual Database 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 schema 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 indentified functional requirements.

There are several notations to draw the ER diagram.

**Logical Database Design**

The result of the logical design phase (or data model mapping phase) is a set of relation schemas. The ER diagram or class diagram is the basis for these relation schemas.

To create the relation schemas is quite a mechanical operation. There are rules how the ER model or class diagram is transferred to relation schemas.

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 Database 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 datatypes and have different datatypes.

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.

**Characteristics of Relations**

* Ordering of tuples in a relation r(R)
  + The tuples are not considered to be ordered , even though they appear to be in the tabular form.
  + Ordering of attributes in a relation schema R (and of values within each tuple):
  + We will consider the attributes in R(A1, A2, . .., An) and the values in t=<v1, v2, . .., vn> to be ordered.
* Values in a tuple:
  + All values are considered atomic (indivisible).
  + Each value in a tuple must be from the domain of the attribute for that column
  + If tuple t = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
  + Then each vi must be a value from dom(Ai)
* Notation:
  + We refer to component values of a tuple t by: t[Ai] or t.Ai

This is the value vi of attribute Ai for tuple t

Similarly, t[Au, Av, . .., Aw] refers to the subtuple of t containing the values of attributes Au, Av, . .., Aw, respectively in t.

**ER to relational mapping algorithm**

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**

* For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
* Choose one of the key attributes of E as the primary key forλ R.
* If the chosen key of E is composite, the set of simpleλ attributes that form it will together form the primary key of R

**Mapping of Strong Entity**

**Mapping of Weak Entity**

* For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
* Also, include as foreign key attributes of R the primary keyλ attribute(s) of the relation(s) that correspond to the owner entity type(s).
* The primary key of R is the combination of the primary key(s) ofλ the owner(s) and the partial key of the weak entity type W, if any.

**Mapping of Binary 1:1 Relation Types**

* For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

➲ There are three possible approaches:

1. **Foreign Key approach**: Choose one of the relations-say S-and 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. Example: 1:1 relation MANAGES is mapped by choosing the participatingν entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total. 1.
2. **Merged relation option**: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types 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 relat ion R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

**Mapping of Binary 1:M Relation Types**

* 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.
* Include as foreign key in S the primary key of the relation Tλ that represents the other entity type participating in R.
* Include any simple attributes of the 1:N relation type asλ attributes of S.

**Mapping of Binary M:N Relation Types**

* For each regular binary M:N relationship type R, create a new relation S to represent R.
* Include as foreign key attributes in S the primary keys of theλ relations that represent the participating entity types; their combination will form the primary key of S.
* Also include any simple attributes of the M:N relationship type (orλ simple components of composite attributes) as attributes of S.

**Mapping of Multivalued attributes**

* For each multivalued attribute A, create a new relation R.
* This relation R will include an attribute corresponding to A, plus theλ 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.
* The primary key of R is the combination of A and K. If theλ multivalued attribute is composite, we include its simple components.

**Mapping of N-ary Relationship Types**

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