**ST. XAVIER’S COLLEGE**

(Affiliated to Tribhuvan University)

Maitighar, Kathmandu



Database Management System

Theory Assignment #4

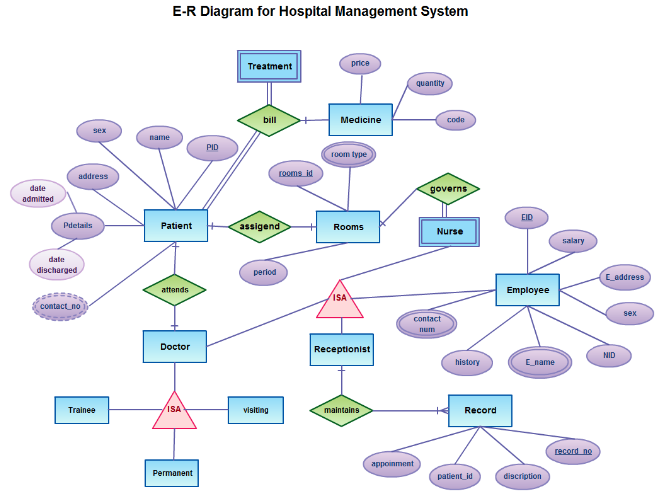
**Submitted by Submitted to**

Victor Kumar Sapkota Er. Sanjay Kr. Yadav 013BSCCSIT047 Lecturer

Sem IV Department of Computer Science

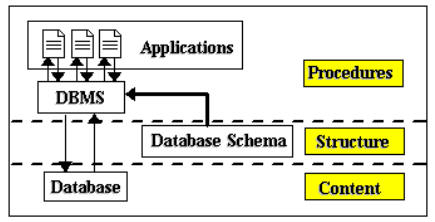
**Submitted on:** August 10, 2015

**E-R diagram with a case study**

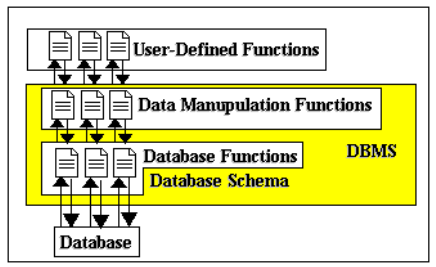


**Functional design**

In conventional database systems, procedures, data structures and actual content are usually separated. Thus, a conventional database management systems (DBMS) provides users with a possibility to store, modify or retrieve data that structured in accordance with a current database schema.



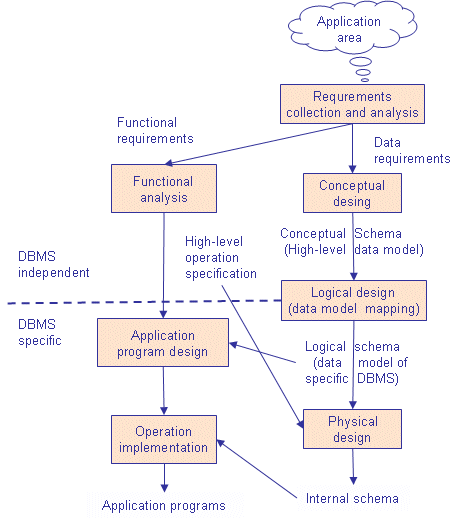
It should be especially noted, that a DBMS retrieves data as they were stored into the database and additional procedures can be applied to such data as an independent level of application programs. In contrast, the functional data model provides a unified approach to manipulation both data and procedures. Main idea of the functional data model is a definition of all components of an information system in the form of functions. Thus, for example, the functional data model defines data objects, attributes and relationships as so-called database functions.



Moreover, a Functional Data Manipulation Language is a number of data manipulation functions which can be applied to database functions. Finally, users are provided with a special mechanism which is called Lambda Calculus to define their own functions which can be seamlessly combined with database and data manipulation functions mentioned above.

**Database design**

In the picture below there are the main phases of database design. Database design is connected with application design.  
  
The requirements and the collection analysis phase produce both data requirements and functional requirements. The data requirements are used as a source of database design. The data requirements should be specified in as detailed and complete form as possible.  
  
In parallel with specifying the data requirements, it is useful to specify the known functional requirements of the application. These consist of user-defined operations that will be applied to the database (retrievals and updates). The functional requirements are used as a source of application software design. Of course some functions may produce also needs for database design.  
  
Note that some phases are database management system independent and some are dependent. The idea is to design first the database without thinking about the actual database system - just to concentrate on the data.



*Fig.: Main phases of database design*

**Conceptual Design**

Once all the requirements have been collected and analyzed, the next step is to create a conceptual shema 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.  
  
There are several notations to draw the ER diagram.

**Logical Design**

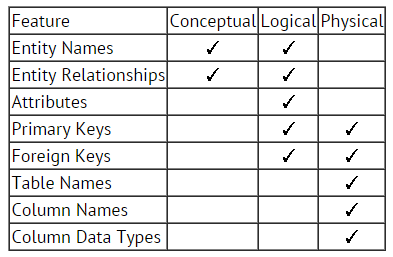
The result of the logical design phase (or data model mapping phase) is a set of relation shcemas. The ER diagram or class diagram is the basis for these relation schemas.  
  
To create the relation shemas is quite a mechanical operation. There are rules how the ER model or class diagram is transferred 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.

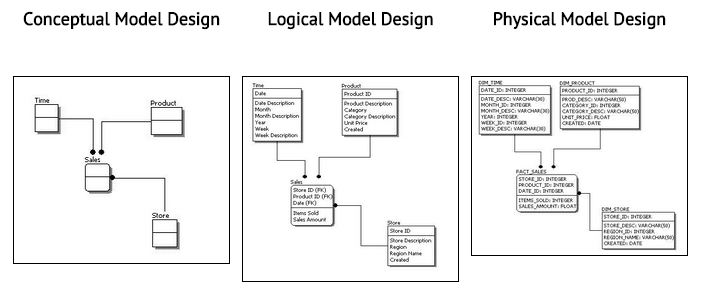
**Normalization**

Normalization is the last part of the logical design. The goal of normalization is to eliminate redundancy and potential update anomalies.  
  
Redundancy means that the same data is saved more than once in a database. Update anomaly is a consequence of redundancy. If a piece of data is saved in more than one place, the same data must be updated in more than one place.  
  
Normalization is a technique by which one can modify the relation schema to reduce the redundancy. Each normalization phase adds more relations (tables) into the database.

**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 datatypes and have different datatypes.  
  
The SQL clauses to create the database are written. The idexes, 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**

No Duplicate Tuples - A relation cannot contain two or more tuples which have the same values for all the attributes. i.e., In any relation, every row is unique.   
• Tuples are unordered - The order of rows in a relation is immaterial.   
• Attributes are unordered - The order of columns in a relation is immaterial.   
• Attribute Values are Atomic - Each tuple contains exactly one value for each attribute.  
It may be noted that many of the properties of relations follow the fact that the body of a relation is a mathematical set.

**E-R to relational mapping algorithm**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EMPLOYEE | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  |  |  | | --- | --- | --- | --- | | Fname | Lname | SSN¯ | SupervisorSSN | | |
| DEPARTMENT | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | |  |  |  |  | | --- | --- | --- | --- | | Name | NUMBER¯ | MANAGER-SSN | StartDate | | |
| DEPENDENT | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | |  |  |  | | --- | --- | --- | | Relationship | EMPL-SSN¯ | Name¯ | | |
| DEP-LOCATION | |  |  |  |  | | --- | --- | --- | --- | |  | |  |  | | --- | --- | | Location | DEP-NUMBER | | |
| WORKS-FOR | |  |  |  |  | | --- | --- | --- | --- | |  | |  |  | | --- | --- | | EmployeeSSN | DeptNumber | | |

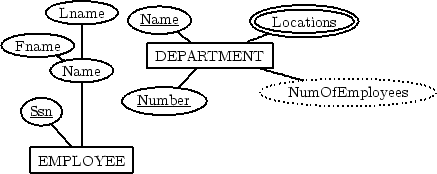
**Comments:**

|  |  |  |
| --- | --- | --- |
| EMPLOYEE (SupervisorSSN) | → | EMPLOYEE (SSN) |
| DEPARTMENT (MANAGER-SSN) | → | EMPLOYEE (SSN) |
| DEPENDENT (EMPL-SSN) | → | EMPLOYEE (SSN) |
| DEP-LOCATION (DEP-NUMBER) | → | DEPARTMENT (NUMBER) |
| WORKS-FOR (EmployeeSSN) | → | EMPLOYEE (SSN) |
| WORKS-FOR (DeptNumber) | → | DEPARTMENT (NUMBER) |
| unique: DEPARTMENT.NAME |  |  |

create table EMPLOYEE{    
  Fname...    
  Lname...    
  SSN... primary key    
  SupervisorSSN... reference EMPLOYEE(SSN)    
}    
create table DEPARTMENT{    
  NAME...    
  NUMBER... primary key    
  MANAGER-SSN... references EMPLOYEE(SSN)    
  StartDate    
}    
create table DEPENDENT{    
  Relationship...    
  EMPL-SSN... references EMPLOYEE(SSN)    
  Name ...    
  primary key(EMPL-SSN,Name)    
}    
create table DEP-LOCATION{    
  Location... primary key    
  DEPNUMBER... references DEPARTMENT(Number)    
}    
create table WORKS-FOR{    
  EmployeeSSN... references EMPLOYEE(SSN)    
 DeptNumber... references DEPARTMENT(Number)    
}

**(Strong) Entity Type into Relation**

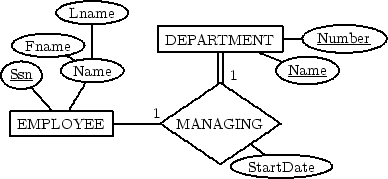
* Include the simple attributes
* Include the simple components of the composite attributes
* Identify the primary keys
* Don’t include: non-simple components of composite attributes, foreign keys, derived attributes, relational attributes

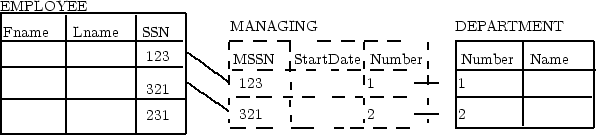


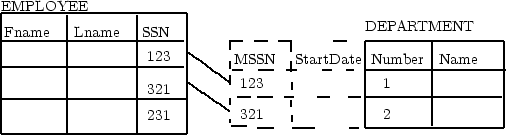
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | EMPLOYEE | |  |  |  | | --- | --- | --- | | Fname | Lname | SSN¯ | | |
| |  |  |  |  | | --- | --- | --- | --- | | DEPARTMENT | |  |  | | --- | --- | | Name | Number¯ | | |

**Binary 1:1 Relationship Types into Foreign Keys**

* Include as foreign keys, in the relation of one entity type, the primary keys of the other entity type
* Include also the simple attributes of the relationship type
* If possible, the first entity type should have total participation in the relationship (to save memory!)



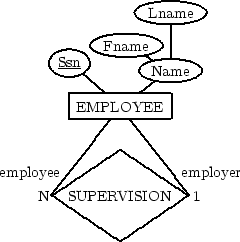


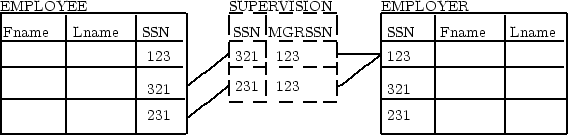


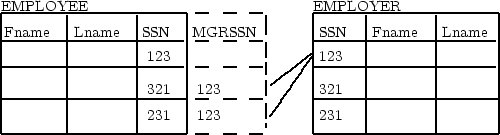
|  |  |  |  |
| --- | --- | --- | --- |
| DEPARTMENT | |  |  | | --- | --- | | MANAGER-SSN | StartDate | |

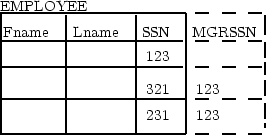
**Binary 1:N Relationship Types into Foreign Keys**

* Add as foreign keys, to the relation of the entity type at the N side, the primary keys of the entity type at the 1 side (don’t duplicate records!)
* Include also the simple attributes of the relationship type





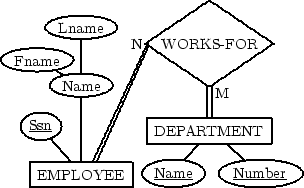


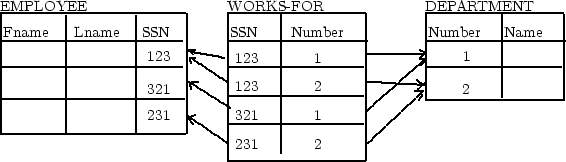


|  |  |  |
| --- | --- | --- |
| EMPLOYEE | |  | | --- | | SupervisorSSN | |

**Binary M:N Relationship Type into Relation**

* We don’t want to duplicate records!
* Set as foreign keys the primary keys of the participating entity types
* Include the simple attributes of the relationship type





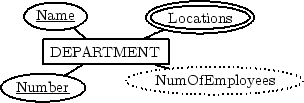
|  |  |  |  |
| --- | --- | --- | --- |
| WORKS-FOR | |  |  | | --- | --- | | EmployeeSSN | DeptNumber | |

**N-Ary Relationship Type**

Similar to binary M:N relationship type

**Multivalued Attribute into Relation**

* Include the given attribute
* Include as foreign keys the primary attributes of the entity/relationship type owning the multivalued attribute
* Keys not designated within primary keys are to be mentioned as such in side comments



|  |  |  |  |
| --- | --- | --- | --- |
| DEP-LOCATION | |  |  | | --- | --- | | Location | DEP-NUMBER¯ | |

Resembles the treatment of a relationship type.