ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System Assignment #4**

**Submitted by:**

Bikash Paneru  
013BSCCSIT012

**Submitted to:**

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

**Date of Submission: August 12th, 2015**

**ER Diagram with One Case Study**

**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.

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, 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**

**Conceptual Database Design**

It is a process of constructing a data model for each view of the real world problem which is independent of physical considerations.

This step involves:

* Constructing the ER Model
* Check the model for redundancy
* Validating the model against user transactions to ensure all the scenarios are supported

**Logical Database Design**

It is a process of constructing a model of information , which can then be mapped into storage objects supported by the Database Management System.

This step involves:

* Table Generation From ER Model
* Normalization of Tables

**Physical Database Design**

The physical design of the database specifies the physical configuration of the database on the storage media.

This step involves describing the base relations, file  organisations, and indexes design used to achieve efficient access to the data, and any associated integrity constraints and security measures.

**Characteristics of Relations**

* 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.

**ER to Relational Mapping Algorithm**

1. **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 for 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.
* Example: We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram.
  + SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.

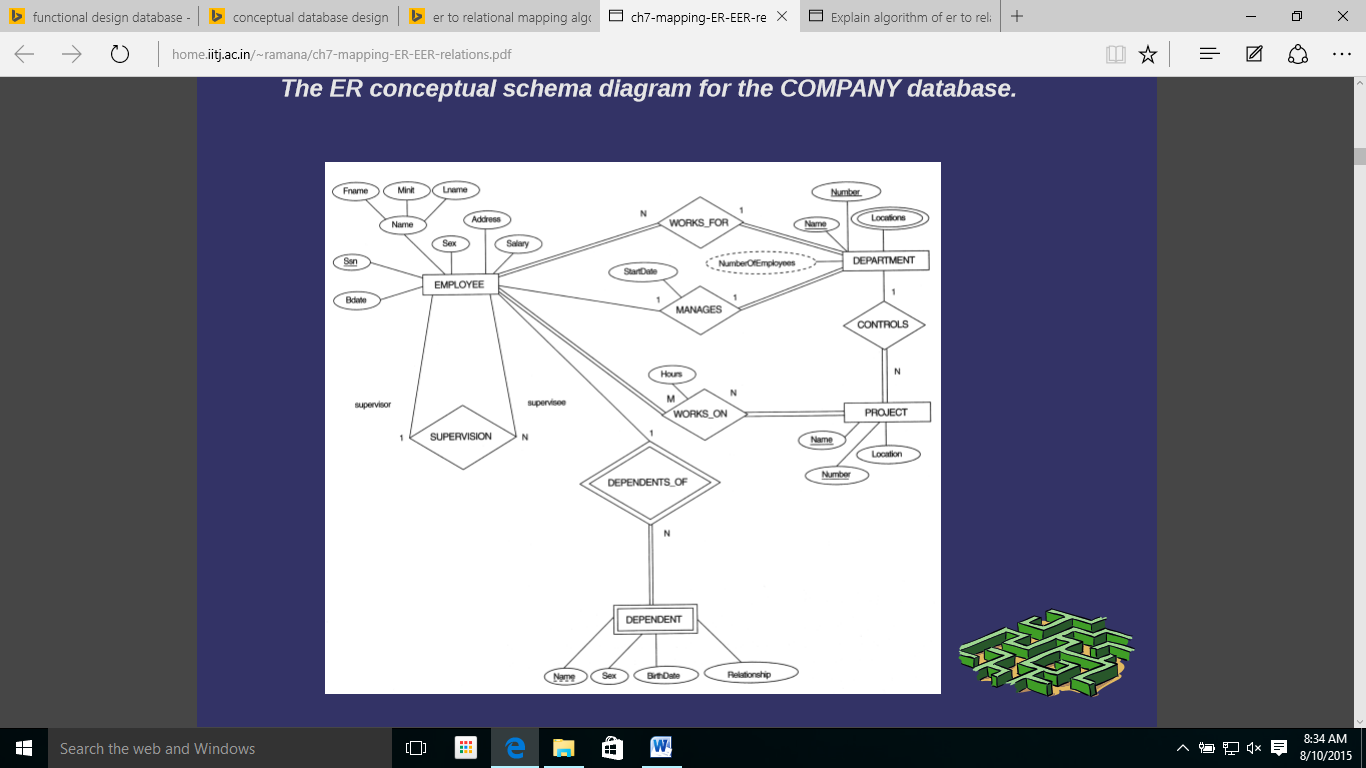


Figure 1 The ER conceptual schema diagram for the COMPANY database.

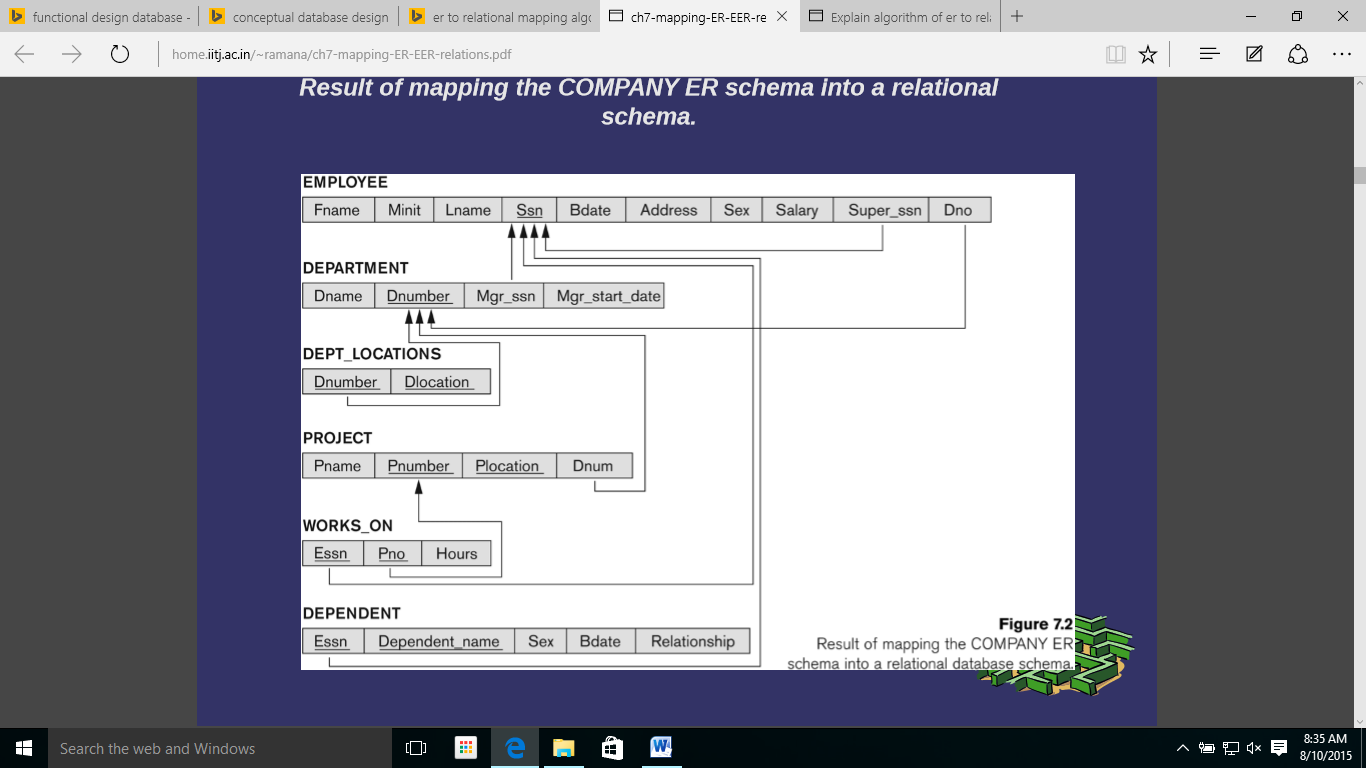


Figure 2 Result of mapping the COMPANY ER schema into a relational schema.

1. **Mapping of Weak Entity Types**

* 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 Was attributes of R.
* Also, include as foreign key attributes of R the primary key 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) of the owner(s) and the partial key of the weak entity type W, if any.
* Example: Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.
  + Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).
  + The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT\_NAME} because DEPENDENT\_NAME is the partial key of DEPENDENT.

1. **Mapping of binary one-to-one relationship 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 participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
  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 relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

1. **Mapping of binary one-to-n relationship 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 T that represents the other entity type participating in R.
* Include any simple attributes of the 1:N relation type as attributes of S.
* Example: 1:N relationship types WORKS\_FOR, CONTROLS, and SUPERVISION in the figure.
  + For WORKS\_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

1. **Mapping of binary m-to-n relationship 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 the 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 (or simple components of composite attributes) as attributes of S.
* Example: The M:N relationship type WORKS\_ON from the ER diagram is mapped by creating a relation WORKS\_ON in the relational database schema.
  + The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS\_ON and renamed PNO and ESSN, respectively.
  + Attribute HOURS in WORKS\_ON represents the HOURS attribute of the relation type. The primary key of the WORKS\_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

1. **Mapping of binary multivalued attributes**

* For each multivalued attribute A, create a new relation R.
* 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.
* The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.
* Example: The relation DEPT\_LOCATIONS is created.
  + The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key represents the primary key of the DEPARTMENT relation.
  + The primary key of R is the combination of {DNUMBER, DLOCATION}.

1. **Mapping of binary 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 the relations that represent the participating entity types.
* Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.
* Example: The relationship type SUPPY in the ER on the next slide.
  + This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}