Transforming Entity Relationship diagrams into Relations

References

McFadden Chapter 6

Types of entities

Regular (strong): entities that have an independent existence and generally represent real-world objects such as persons and products.

Weak entities: entities that cannot exist except with an identifying relationship with an owner (regular) entity type.

Mapping regular entities

1. Each regular entity type in an ER diagram is transformed into a relation.
2. Each attribute of the entity type becomes the attribute of the relation.
3. The identifier of the entity type becomes the primary key of the corresponding relation.

Example in Fig 1.

Mapping weak entities

1. For each weak entity type, create a new relation and include all the simple attributes as attributes in this new relation.
2. Then include the primary key of the owner relation as a foreign key attribute in this new relation.
3. The primary key of the new relation is the combination of this primary key of the owner and the partial identifier of the weak entity type.

Example in Fig 2.

Mapping binary relationships

The representation of relationships depends on the degree of the relationships (unary, binary, ternary), the cardinalities of the relationships and the membership/participation (mandatory/obligatory, optional/non-obligatory).

Mapping One-to-Many (1:M) relationships

Membership is mandatory/obligatory:

1. For each binary 1:M relationship, first create a relation for each of the two types participating in the relationship, using the procedure for regular entities.

2. Include the primary key attribute (or attributes) of the entity on the one-side of the relationship as a foreign in the relation that is on the many-side of the relationship (rule: *the primary key migrates to the many side*)

Example in Fig 3.

Membership/participation is optional/non-obligatory:
1. For each binary 1:M relationship, first create a relation for each of the two types participating in the relationship, using the procedure for regular entities.
2. Create a separate relation/table for the relationship. The new relation contains only two attributes that are the identifiers of the two entity types. The primary key of the new relation is a composite key made up of the two primary keys of the other relations.

Mapping Binary Many-to-Many Relationships

1. First create a relation for each of the two types participating in the relationship, using the procedure for regular entities.
2. Create a separate relation/table for the relationship. The new relation contains only two attributes that are the identifiers of the two entity types. The primary key of the new relation is a composite key made up of the two primary keys of the other relations.

Example in Fig 4.

Mapping One-One (1:1) Relationships

Binary one-to-one relationships can be seen as a special case of one-to-many relationships.

Membership/participation is mandatory for one entity type

1. Create two relations, one for each of the participating entity type.
2. Include the primary key of the relation corresponding to the mandatory entity type as a foreign key in the relation corresponding to the optional entity type.

Example in Fig 5

Membership/participation is mandatory for both entity types

1. Create one single relation that includes the attributes of both entity types.
2. Select one identifier from the two identifiers as primary key.

<u>Membership/participation is optional for both entity types</u>

1. Create two relations, one for each of the participating entity types.
2. Create a separate relation/table for the relationship. The new relation contains only two attributes that are the identifiers of the two entity types. The primary key of the new relation is a composite key made up of the two primary keys of the other relations.

Mapping Unary Relationships

Unary One-to-Many (1:M) Relationships

1. The entity type in the unary relationship is mapped to a relation using the procedure for regular entities.
2. Then a foreign key attribute is added within the same relation that references the primary key values (this foreign key must have the same domain as the primary key).

A **recursive foreign key** is a foreign key in a relation that references the primary key values of that same relation.

Fig. 6 shows a unary one-to-many relationship named *manages* that associates each employee of an organisation with another employee who is his or her manager. Each employee has exactly one manager; a given employee may manage zero to many employees.

Unary Many-to-Many Relationships

With this type of relationships two relations are created: one to represent the entity type in the relationship and the other an associative relation that represents the M:N relationship itself. The primary key of the associative relation consists of two attributes. These attributes (which need not have the same name) take their values from the primary key of the other relation. Any nonkey attribute of the relationship is included in the associative relation.

Example in given in Fig 7.

Exercises

For each of the following cases,

1. **Draw the entity relationship diagram**
2. **Transform the diagram into relations.**

1. Company cars are assigned to employees and no car is shared between employees, and no employees has the use of more that one car. Employees and cars are identified by employee# and car#, respectively. Every employee has a company car, and every company car is used by an employee.

2. Same scenario as in 1. above, except that not every employee has a car.

3. Same scenario as in 1. above, except that employees do not necessarily have company cars, and cars are not necessarily used by employees.

4. Consider the assignments of patients to hospital wards. A ward may contain many patients. Patients and wards are identified by patient# and ward#, respectively. Every patient must belong to a ward.

5. Same scenario as in 4., except that some patients do not belong to a ward.

6. A lecturer can teach many students, and a student can be taught by many lecturers. Lecturers and students are identified by lecturer# and student# respectively.

7. A library keeps information about its books and borrowers, including a record of which books are currently on loan to which borrower. Each copy is identified by an accession#, and each borrower by a borrower#.

8. Suppose every employee has employee#, home-address and salary. Only those employees who are salesmen have associated values of sales-quota and sales-bonus.

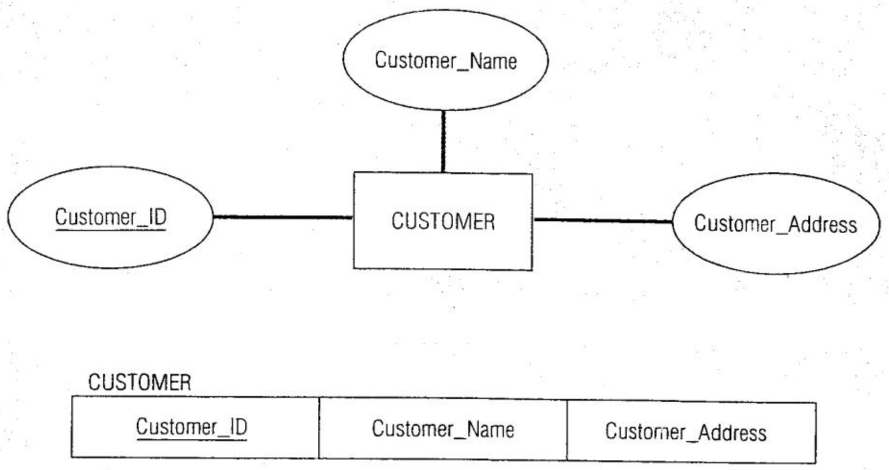


Figure (1
Mapping the regular entity CUSTOMER

(a) CUSTOMER entity type

(b) CUSTOMER relation

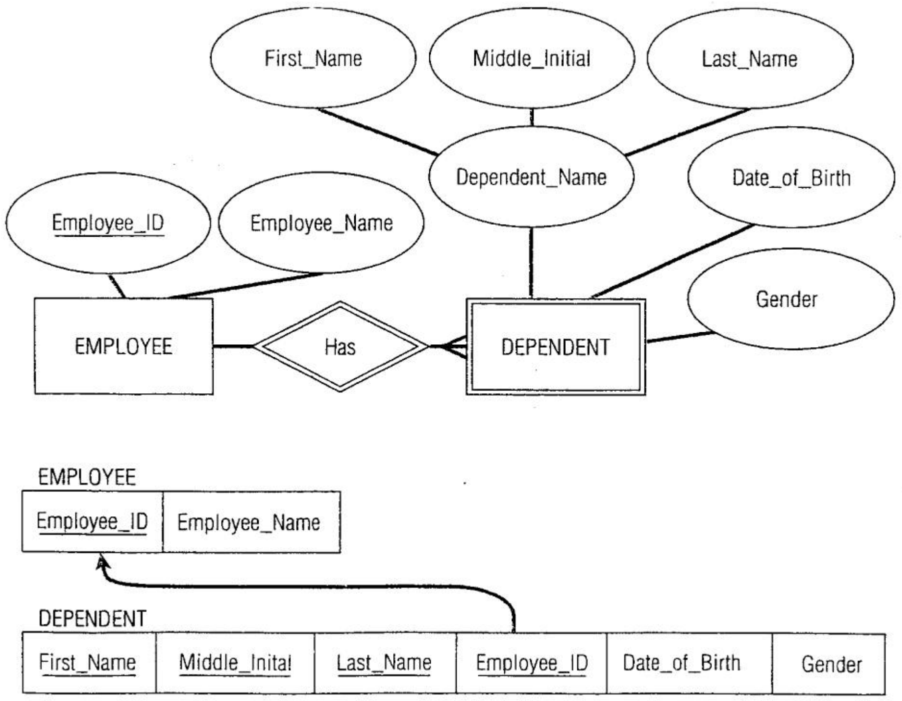


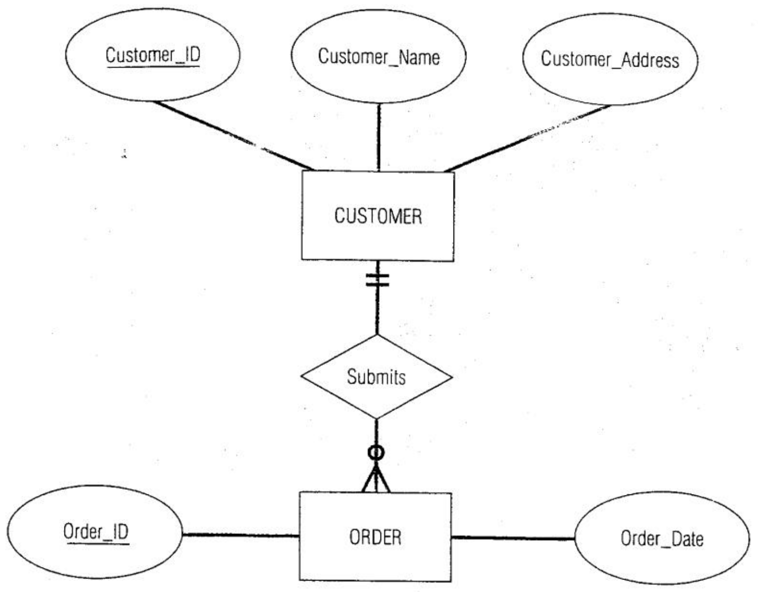
Figure (2
Example of mapping a weak entity

(a) Weak entity DEPENDENT

(b) Relations resulting from weak entity

Figure 3
Example of mapping a 1:M relationship

(a) Relationship between customers and orders



(b) Mapping the relationship

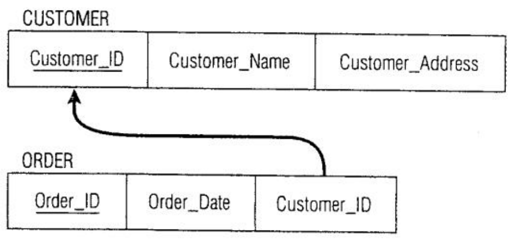
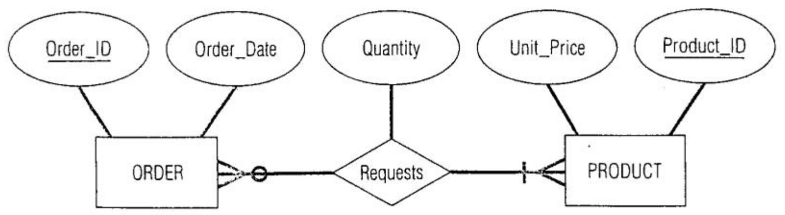


Figure 4
Example of mapping an M:N relationship

(a) Requests relationship (M:N)



(b) Three resulting relations

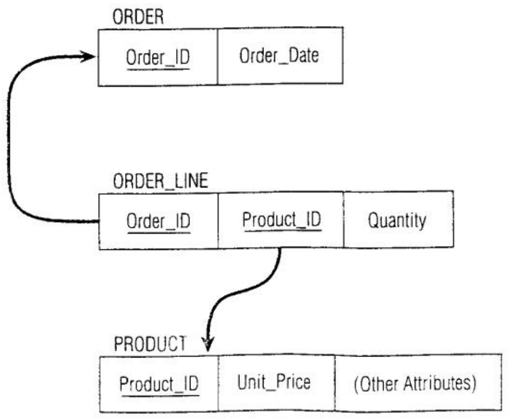
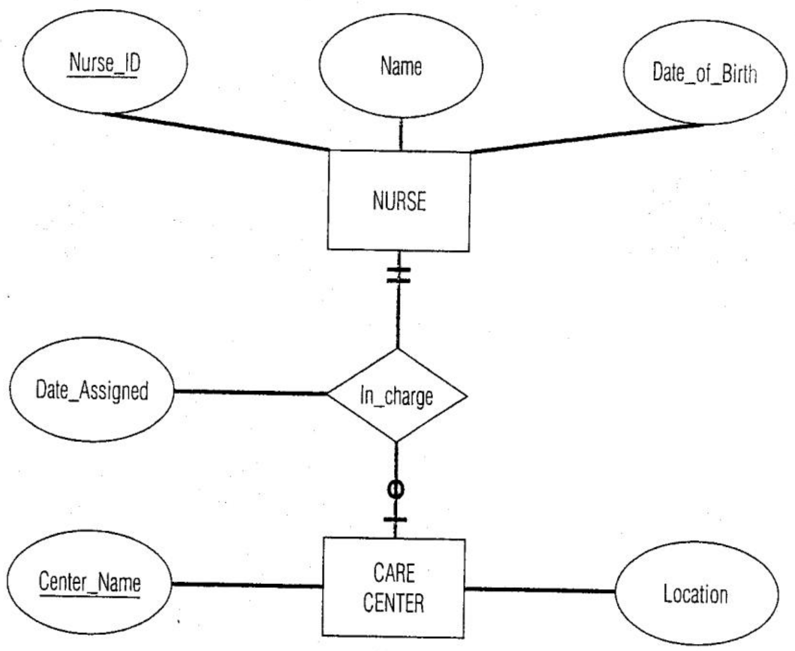


Figure 5
Mapping a binary 1:1 relationship

(a) Binary 1:1 relationship



(b) Resulting relations

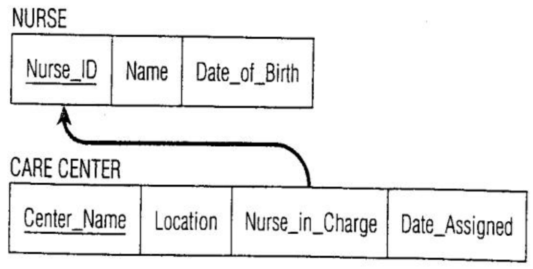
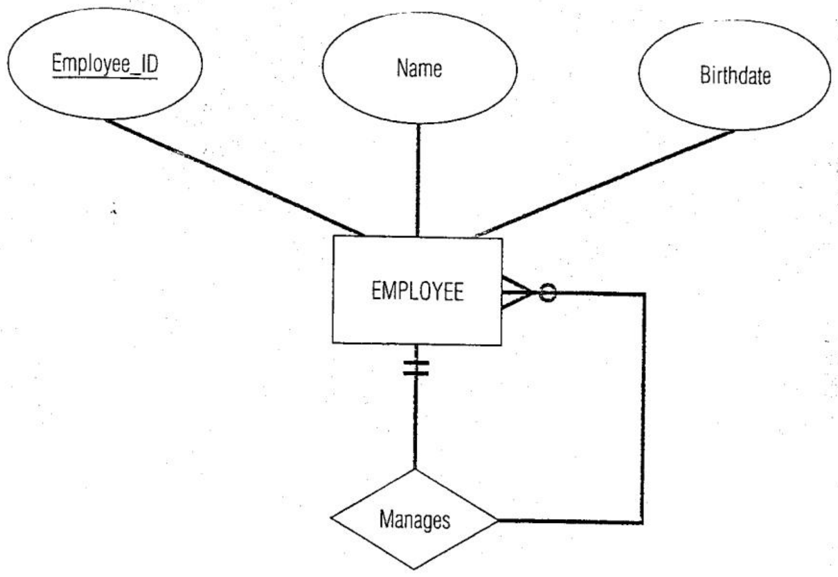


Figure 6
Mapping a unary 1:N relationship

(a) EMPLOYEE entity with Manages relationship



(b) EMPLOYEE relation with recursive foreign key

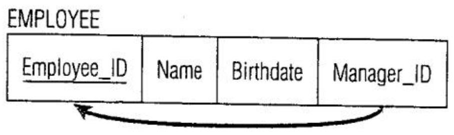
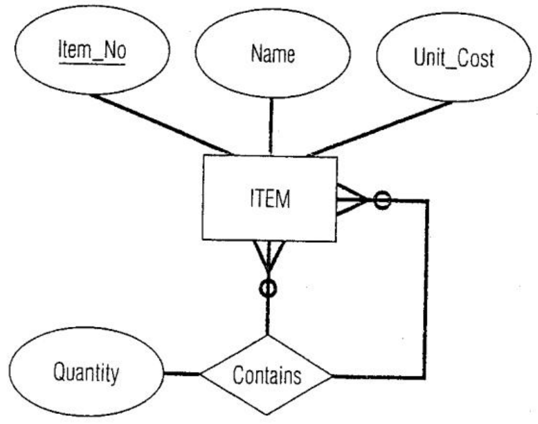
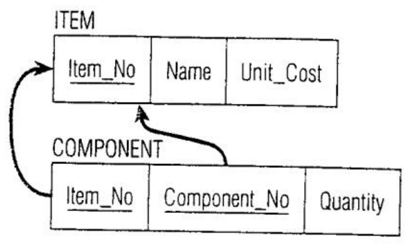


Figure 7
Mapping a unary M:N relationship

(a) Bill-of-materials relationships (M:N)



(b) ITEM and COMPONENT relations


ITEM
Item_No
Name
Unit_Cost
COMPONENT
Item_No
Component_No
Quantity