TRANSFORMING EER DIAGRAMS INTO RELATIONS

During logical design, you transform the E-R (and EER) diagrams that were developed during conceptual design into relational database schemas.

Review:

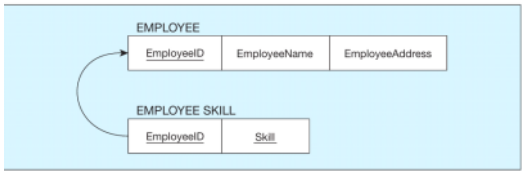
1. Regular entities are entities that have an independent existence and generally represent real-world objects, such as persons and products. Regular entity types are represented by rectangles with a single line.
   - Regular run of the mill entities like customer, employee, etc..
2. Weak entities are entities that cannot exist except with an identifying relationship with an owner (regular) entity type. Weak entities are identified by a rectangle with a double line.
   - Would be something like a dependent of an EMPLOYEE
3. Associative entities (also called gerunds) are formed from many-to-many relationships between other entity types. Associative entities are represented by a rectangle with rounded corners.
   - Would be something that is derived from a mulivalued attribute like say skill for EMPLOYEE

Step 1: Map Regular Entities

- Make each regular entity into a relation table
- The name of relation should be the same as the entity
- Make sure it has a unique id attribute (primary key)
- Composite Attributes
  - When your regular entity has a composite value they will turn into simple attribute components in the relation table
    - Note that when I say entity Im referring to the EER diagram represention and I am talking in reference to that diagram and when I say relation table I am referring to the relational tables we are now creating
- Multivalued attributes
  - When a regular entity contains a multivalued attribute (like say EMPLOYEE {skill}) you will create a new table for it (skill) and do the whole foreign key mapping as well as maybe a composite key to make up the primary key
  - Remember that the DBMS that you are using limits your choices on how you wanna go about tackling multivalued attributes, some will let you house them in the same relation as an attribute and some will make you create a new relation table for them

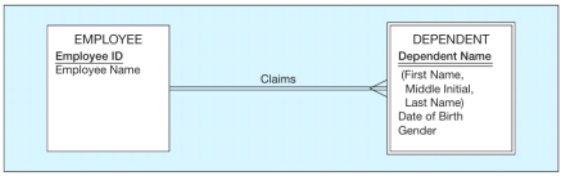
Notice the naming difference between a multivalued attribute vs a weak entity relation

-



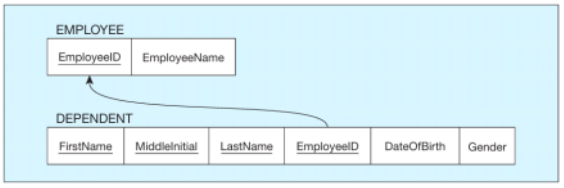
Step 2: Map Weak Entities

Recall that a weak entity type does not have an independent existence but exists only through an identifying relationship with another entity type called the owner.



Notice the difference in naming between the two relation tables here and in the mulitivalued one above. DEPENDENT isnt named EMPLOYEE DEPENDENT its just named DEPENDENT

Notice that DEPENDENT has many composite keys that make up its primary key

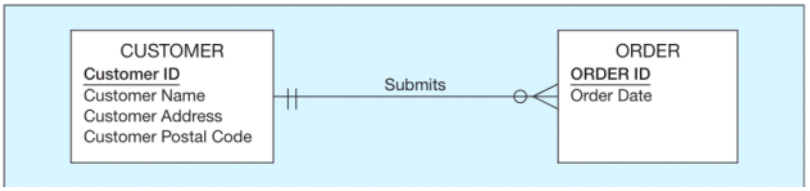


- Here we can make the case for a surrogate key: A serial number or other system assigned primary key for a relation.
  - In this example, we could create a new attribute in DEPENDENT called DependentID instead of using a bunch of composite attributes
    - Notice that this solution will ensure unique identification for each dependent (even for those of George Foreman!).
  - There are certain situations where a surrogate key is needed and you can look it up if need be

Step 3: Map Binary Relationships

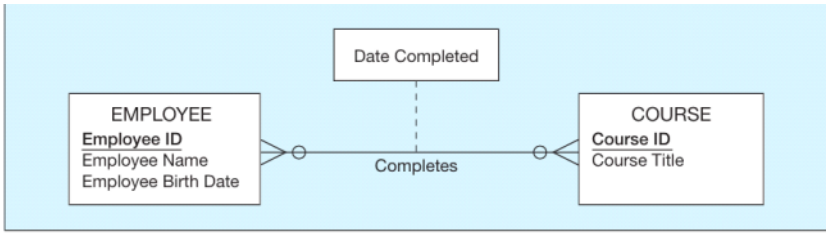
The procedure for representing relationships depends on both the degree of the relationships (unary, binary, or ternary) and the cardinalities of the relationships.

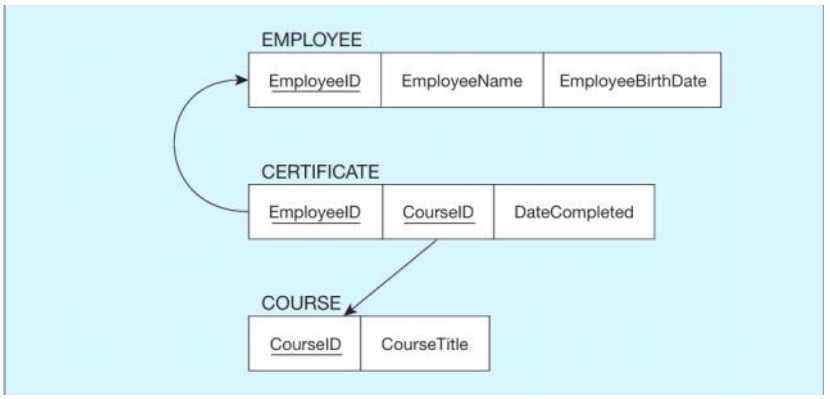
MAP BINARY ONE-TO-MANY RELATIONSHIPS:



"The primary key migrates to the many side."

MAP BINARY MANY-TO-MANY RELATIONSHIPS



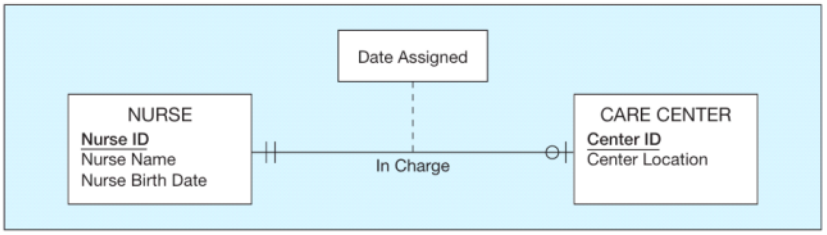


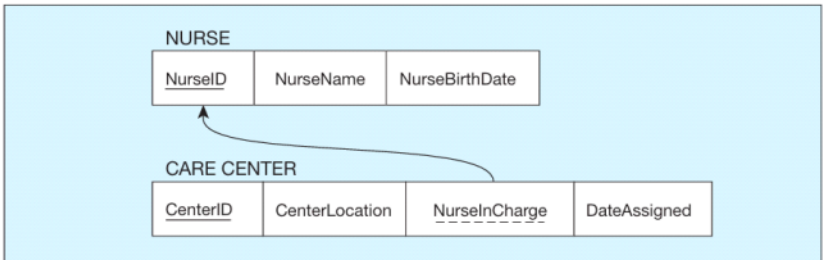
"Suppose that there is a binary many to-many (M:N) relationship between two entity types, A and B. For such a relationship, create a new relation, C. Include as foreign key attributes in C the primary key for each of the two participating entity types. These attributes together become the primary key of C. Any nonkey attributes that are associated with the M:N relationship are included with the relation C."

Notice how we create a new relation table called certificate. The relationship would read something like An employee exists who has a certificate for some course.

Also note it is wise to create a surrogate key for CERTIFICATE instead of having a composite key

MAP BINARY ONE-TO-ONE RELATIONSHIPS





"The process of mapping such a relationship to relations requires two steps. First, two relations are created, one for each of the participating entity types. Second, the primary key of one of the relations is included as a foreign key in the other relation."
"In a 1:1 relationship, the association in one direction is nearly always an optional one, whereas the association in the other direction is a mandatory one. "

In this example a care center must be assigned a nurse but a nurse does not necessarily have to be assigned a care center

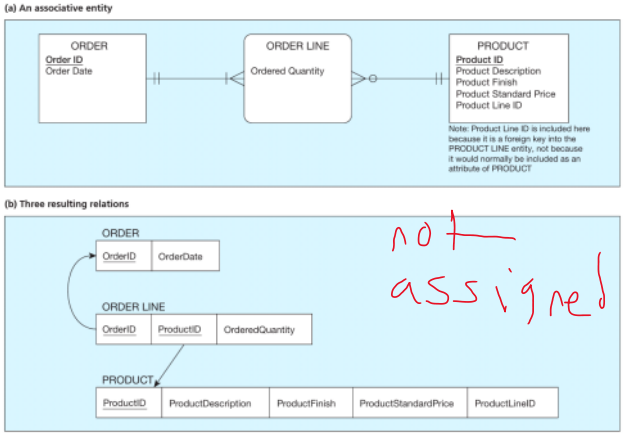
- This means if you tried to look up a specific nurse if in care center you might not find the nurse if they arent assigned a care center

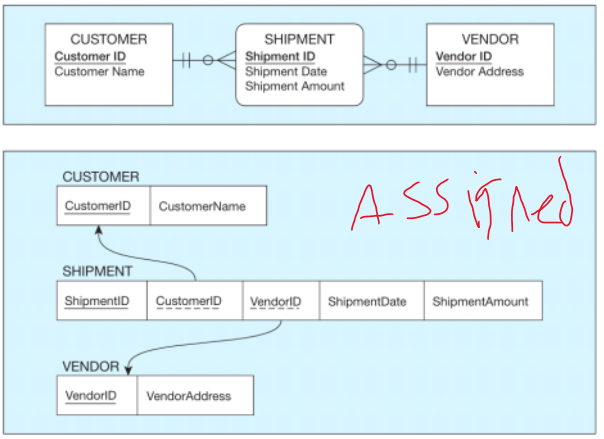
Step 4: Map Associative Entities

Mapping the associative entity involves essentially the same steps as mapping an M:N relationship, as described in Step 3.

- The first step is to create three relations: one for each of the two participating entity types and a third for the associative entity
- The second step then depends on whether on the E-R diagram an identifier was assigned to the associative entity
    - Identifier not assigned: the default primary key will be a composite key of two primary key attributes from the two other rel ations
    - Identifier Assigned:
        - This happens if the identifier is unique in some way or if the composite attributes could have duplicates when combining to m ake primary key
            - If duplicate composite keys are possible than a surrogate key is needed

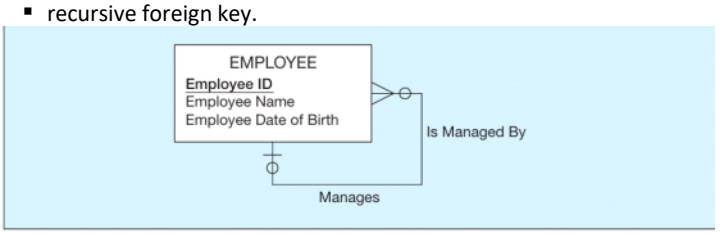
□ If duplicate composite keys are possible than a surrogate key is needed



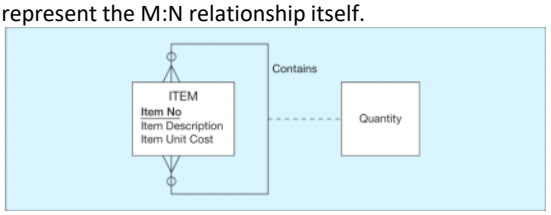


Step 5: Map Unary Relationships

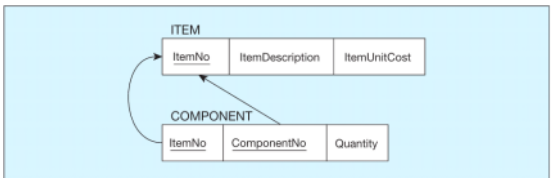
we defined a unary relationship as a relationship between the instances of a single entity type. Unary relationships are also called recursive relationships. The two most important cases of unary relationships are one-to-many and many-to-many relationships.

- Do step 1
- UNARY ONE-TO-MANY RELATIONSHIPS
  - Next, a foreign key attribute is added to the same relation; this attribute references the primary key values in the same relation.
    - recursive foreign key.
  - 

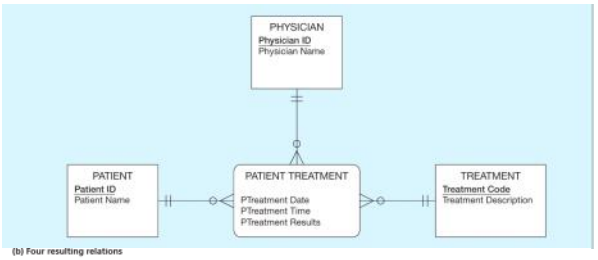
"shows a unary one-to-many relationship named Manages that associates each employee of an organization with another employee who is his or her manager. Each employee may have one manager; a given employee may manage zero to many employees."

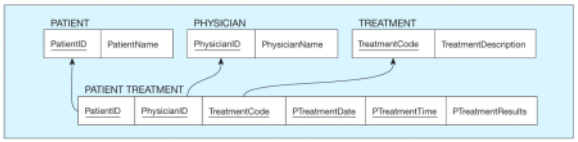
- UNARY MANY-TO-MANY RELATIONSHIPS
  - With this type of relationship, two relations are created: one to represent the entity type in the relationship and an associative relation to represent the M:N relationship itself.
  - 

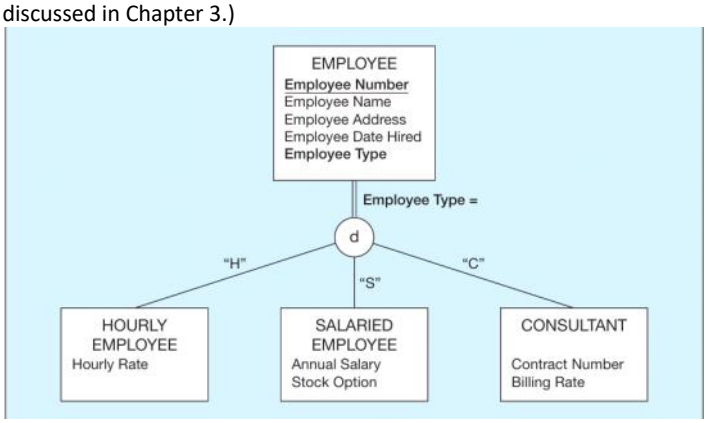
"The relationship (called Contains) is M:N because a given item can contain numerous component items, and, conversely, an item can be used as a component in numerous other items."

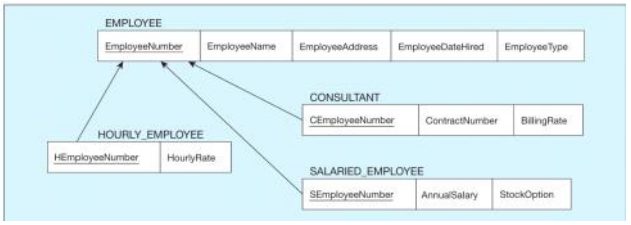


- **Step 6: Map Ternary (and n-ary) Relationships**
  is a relationship among three entity types.
  - we recommended that you convert a ternary relationship to an associative entity to represent participation constraints more accurately.
  - To map an associative entity type that links three regular entity types, you create a new associative relation.

- 

(b) Four resulting relations
- 

- **Step 7: Map Supertype/Subtype Relationships**
  - 1. Create a separate relation for the supertype and for each of its subtypes.
  - 2. Assign to the relation created for the supertype the attributes that are common to all members of the supertype, including the primary key.
    3. Assign to the relation for each subtype the primary key of the supertype and only those attributes that are unique to that subtype.
  - 4. Assign one (or more) attributes of the supertype to function as the subtype dis criminator. (The role of the subtype discriminator was discussed in Chapter 3.)
- 



SUMMARY

TABLE 4-2 Summary of EER-to-Relational Transformations

EER Structure	Relational Representation (Sample Figure)
Regular entity	Create a relation with primary key and nonkey attributes (Figure 4-8).
Composite attribute	Each component of a composite attribute becomes a separate attribute in the target relation (Figure 4-9).
Multivalued attribute	Create a separate relation for multivalued attribute with composite primary key, including the primary key of the entity (Figure 4-10).
Weak entity	Create a relation with a composite primary key (which includes the primary key of the entity on which this entity depends) and nonkey attributes (Figure 4-11).
Binary or unary 1:*M* relationship	Place the primary key of the entity on the one side of the relationship as a foreign key in the relation for the entity on the many side (Figure 4-12; Figure 4-17 for unary relationship).
Binary or unary *M:N* relationship or associative entity without its own key	Create a relation with a composite primary key using the primary keys of the related entities plus any nonkey attributes of the relationship or associative entity (Figure 4-13, Figure 4-15 for associative entity, Figure 4-18 for unary relationship).
Binary or unary 1:1 relationship	Place the primary key of either entity in the relation for the other entity; if one side of the relationship is optional, place the foreign key of the entity on the mandatory side in the relation for the entity on the optional side (Figure 4-14).
Binary or unary *M:N* relationship or associative entity with its own key	Create a relation with the primary key associated with the associative entity plus any nonkey attributes of the associative entity and the primary keys of the related entities as foreign keys (Figure 4-16).
Ternary and *n*-ary relationships	Same as binary *M:N* relationships above; without its own key, include as part of primary key of relation for the relationship or associative entity the primary keys from all related entities; with its own surrogate key, the primary keys of the associated entities are included as foreign keys in the relation for the relationship or associative entity (Figure 4-19).
Supertype/subtype relationship	Create a relation for the superclass, which contains the primary and all nonkey attributes in common with all subclasses, plus create a separate relation for each subclass with the same primary key (with the same or local name) but with only the nonkey attributes related to that subclass (Figure 4-20 and 4-21).