CGS 2545: Database Concepts
Spring 2012

Chapter 4 – Logical Database Design And The Relational Data Model (Part 1)

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cgs2545/spr2012

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

The Relational Data Model

- The relational data model is based on the concept of mathematical relations.
- Codd (the guy who proposed the relational model) was a trained mathematician and he used terminology taken from this discipline, primarily set theory and predicate logic.

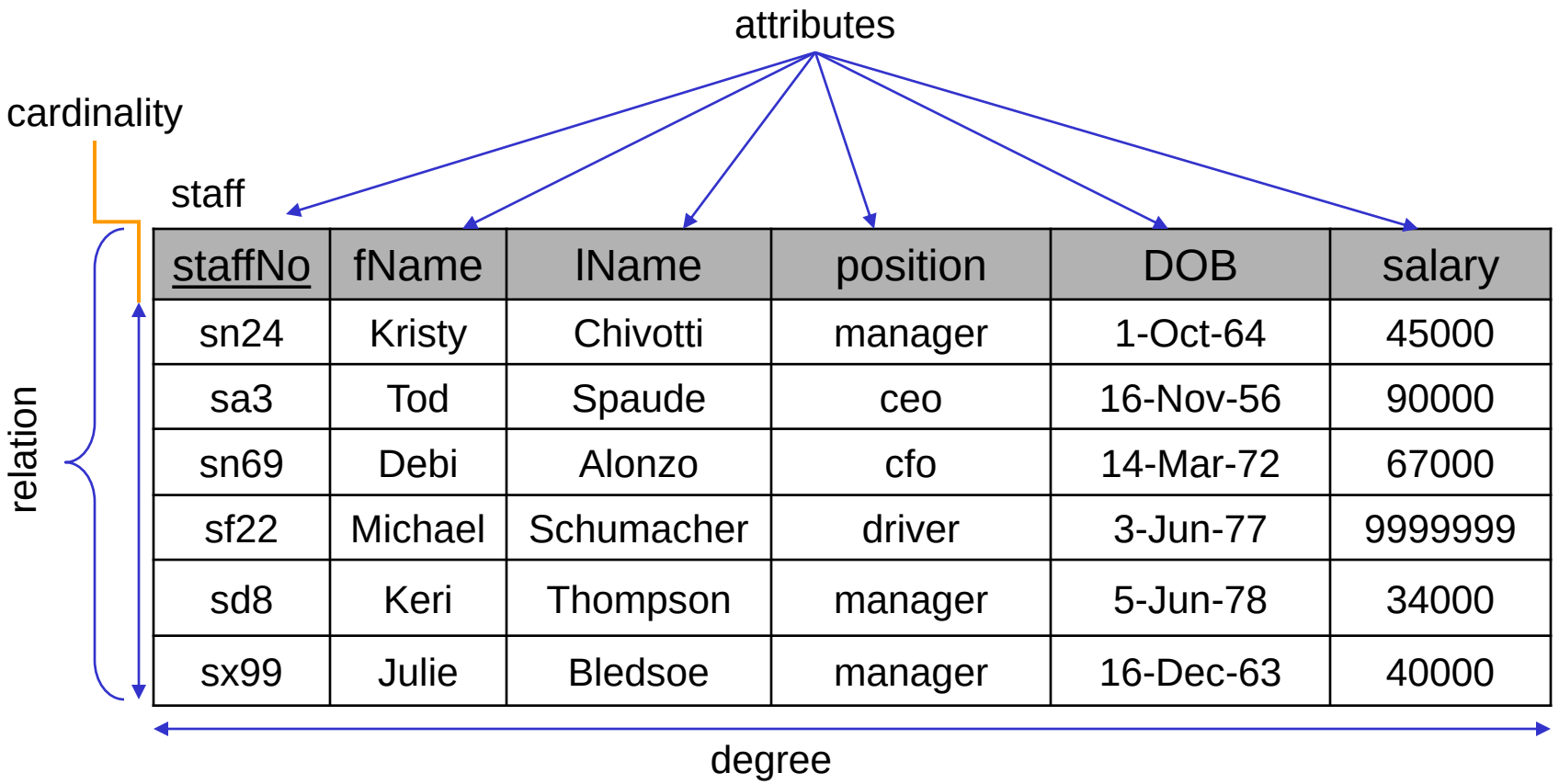
The Relational Data Model (cont.)

- Relation: A relation is a table (matrix) with rows and columns. Relations hold information about the objects modeled in the db.
- Attribute: An attribute is a named column of a relation. An attribute is some characteristic of an entity (or relationship) that is modeled in the database. Attributes can appear in any order in a relation.
- Domain: A domain is the set of allowable values for one or more attributes. Every attribute is defined on some domain. Domains may be distinct for each attribute, or two or more attributes may be defined on the same domain.

The Relational Data Model (cont.)

- Tuple: A tuple is a row of a relation. Tuples can appear in any order in a relation and the relation will remain the same, and therefore convey the same meaning.
- Degree: The degree of a relation is the number of attributes it contains.
- Cardinality: The cardinality of a relation is the number of tuples it contains.
- Relational database: A collection of normalized relations with distinct relation names.

An Example Relation



staffNo	fName	lName	position	DOB	salary
sn24	Kristy	Chivotti	manager	1-Oct-64	45000
sa3	Tod	Spaude	ceo	16-Nov-56	90000
sn69	Debi	Alonzo	cfo	14-Mar-72	67000
sf22	Michael	Schumacher	driver	3-Jun-77	9999999
sd8	Keri	Thompson	manager	5-Jun-78	34000
sx99	Julie	Bledsoe	manager	16-Dec-63	40000

Example Domain Definitions

Attribute	Domain Name	Meaning	Domain Definition
staffNo	staffnumbers	set of all possible staff numbers	character: size 4, must begin with letter s.
fName, lName	name	set of all possible person names	character: size 20
DOB	date	date person was born	date: range from 1-Jan-20, format: dd-mmm-yy
salary	salaries	possible values of staff salaries	monetary: 7 digits, range 10,000-9,999,999
position	alljobs	set of all possible positions	select one from set: {ceo, cfo, coo,manager, asst. manager, driver, secretary}

Alternate Terminology for Relational Model

Formal Term	Alternative 1	Alternative 2
relation	table	file
tuple	row	record
attribute	column	field

What is a Relation

- To understand the true meaning of the term relation, we need to review some basic math concepts:
- Given two sets D_1 and D_2 where

  $D_1 = \{2, 4\}$ and $D_2 = \{1, 3, 5\}$

- The Cartesian product of these two sets, written $D_1 \times D_2$, is the set of all ordered pairs such that the first element is a member of D_1 and the second element is a member of D_2.
  - $D_1 \times D_2 = \{(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)\}$
- Any subset of this Cartesian product is a relation.
  - Thus, we could produce relation R such that: $R = \{(2, 3), (4, 3)\}$
- We can specify some condition which will select elements from $D_1 \times D_2$ to be included in R, such as:
  - $R = \{(x, y) \mid x \in D_1, y \in D_2, \text{and } y = 3\}$

What is a Relation (cont.)

- Given three sets D_1, D_2, and D_3 where

  $D_1 = \{2, 4\}$, $D_2 = \{1, 3\}$, and $D_3 = \{3, 6\}$

- The Cartesian product of three sets, written $D_1 \times D_2 \times D_3$, is the set of all ordered triples such that the first element is a member of D_1, the second element is a member of D_2, and the third element is a member of D_3.
  - $D_1 \times D_2 \times D_3 = \{(2, 1, 3), (2, 1, 6), (2, 3, 3), (2, 3, 6)$

    $(4, 1, 3), (4, 1, 6), (4, 3, 3), (4, 3, 6)\}$
- Any subset of this Cartesian product is a relation.
- In general, if D_1, D_2, ... D_n are *n* sets. Their Cartesian product is defined as: $D_1 \times D_2 \times \cdots \times D_n = \{(d_1, d_2, \cdots, d_n) \mid d_1 \in D_1, d_2 \in D_2, \cdots, d_n \in D_n\}$

  and generally written as: $\mathop{X}_{i=1}^{n} D_i$

What is a Relation (cont.)

- A relational schema is a named relation defined by a set of attribute and domain name pairs.
  - $R_i = \{A_1:d_1, A_2:d_2, ..., A_n:d_n \mid d_1 \in D_1, d_2 \in D_2, ..., d_n \in D_n\}$
- A relational database schema is a set of relation schemas, each with a distinct name.
  - $R = \{R_1, R_2, ..., R_n\}$

What is a Relation (cont.)

A relation has the following properties:

1. The relation has a name that is distinct from all other relation names in the relational schema.
2. Each cell (attribute) contains exactly one atomic value.
3. Each attribute has a distinct name.
4. The values of an attribute are all from the same domain.
5. Each tuple is distinct; there are no duplicate tuples.
6. The order of the attributes has no significance.
7. The order of the tuples has not significance, theoretically. (However, in practice, the order may affect the efficiency of accessing tuples. Much more on this later.)

Relation Schemas vs. Relation Instances

- There is an important distinction to be made between a relation schema and a relation instance.
- The schema is the name and attributes for the relation and is relatively immutable.
- An instance is a set of tuples for that relation, and the instance may change frequently. Indeed most updates and certainly every insert and deletion will change the instance.
  - A snapshot database models the current "state" of the real world which is captured in the database. At any given moment in time it is modeling the current "instance" of the real world. If the real world state changes, so too must the database to maintain the representation of the current real world instance.

Equivalent Relations

A	B	C
1	2	3
3	2	1
4	4	1
2	1	3

a relation instance

B	C	A
2	3	1
2	1	3
4	1	4
1	3	2

a relation instance

A	B	C
4	4	1
3	2	1
1	2	3
2	1	3

a relation instance

equivalent relation instances

A	B	C
4	4	1
3	2	1
1	2	4
2	1	3

a relation instance

this relation instance is not equivalent to any of the other three

Logical Design

- During logical design you transform the conceptual design into relational database schemas.
  - The inputs to the process are the E-R diagrams and the outputs are the relational schemas.
- Mapping the E-R diagrams to relations is a relatively straightforward process with a well-defined set of rules. In fact many CASE tools (Computer Aided Software Engineering tools) can automatically perform many of the conversion steps. However, it is important that you understand the steps in this process for three reasons:

1. CASE tools often cannot model more complex data relationships such as ternary relationships and superclass/subclass relationships. These steps will need to be done manually.
2. There are some legitimate alternatives where you must manually chose an alternative.
3. You need to be prepared to perform a quality check on the results obtained with the CASE tool.

Logical Design (cont.)

- In the steps that we'll need to follow to map E-R diagrams into relational schemas, it will be helpful to remember that we've defined three basic types of entities which are summarized below:
  - Regular (strong) entities are entities that have an independent existence and generally represent real-world objects such as persons or products. Represented in ERDs by rectangles with a single line.
  - Weak entities are entities that cannot exist except with an identifying relationship with an owner (strong) entity type. Weak entities are identified by a rectangle with a double line.
  - Associative entities (also sometimes called gerunds) are formed from many-to-many relationships between other entity types. Associative entities are represented by a rectangle with a single line that enclosed the diamond relationship symbol.

Mapping E-R Diagrams to Relational Schemas

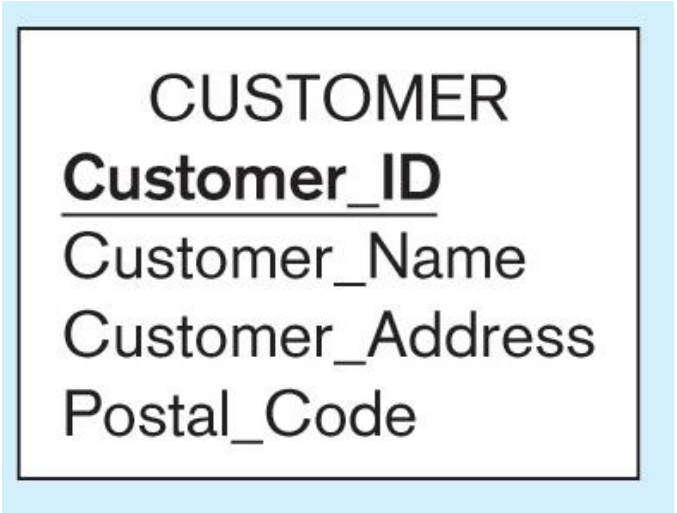
STEP 1: Mapping Regular (Strong) Entities

- Each regular entity in an ERD is transformed into a relation schema.
- The name given to the relation is generally the same as the entity type.
- Each simple attribute of the entity type becomes an attribute of the relation schema.
- The identifier becomes the primary key of the corresponding relation.

Mapping E-R Diagrams to Relational Schemas (cont.)

STEP 1: Map Regular (Strong) Entities - EXAMPLE

Customer entity type ER diagram



Customer relation (table)

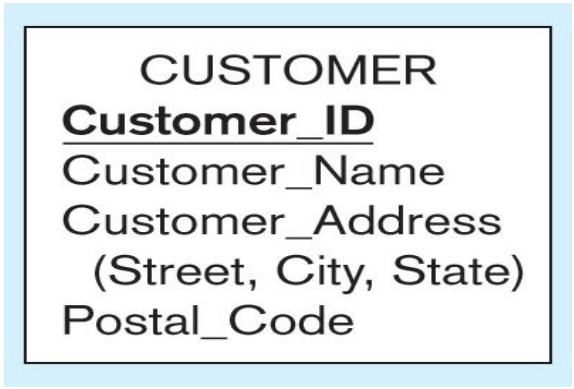
Customer_ID	Customer_Name	Customer_Address	Postal_Code

Mapping E-R Diagrams to Relational Schemas (cont.)

Composite Attributes:

- When a regular entity type has a composite attribute, only the simple component attributes of the composite attribute are included in the new relation schema.

ERD



Note that the composite attribute has disappeared – replaced by its components.

Customer Relation

Customer_ID	Customer_Name	Street	City	State	Postal_Code

Mapping E-R Diagrams to Relational Schemas (cont.)

Multi-valued Attributes:

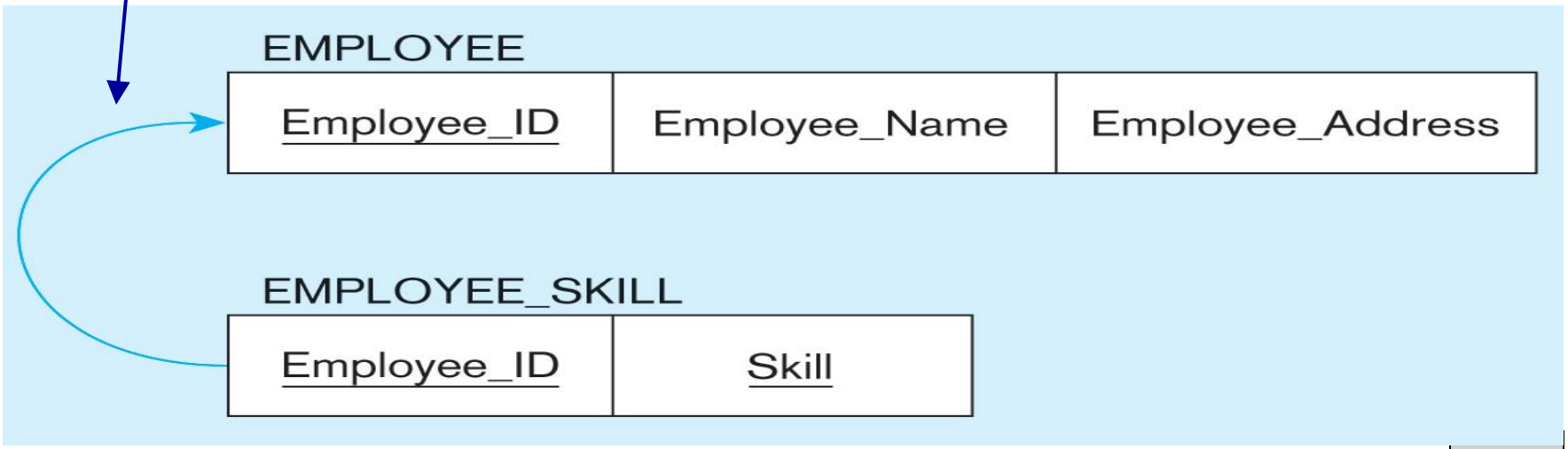
- When a regular entity type contains a multi-valued attribute, two new relation schemas (rather than one) are created.
- The first relation schema contains all of the attributes of the entity type except the multi-valued attribute. The second relation schema contains two attributes that form the primary key of the second relation schema. The first of these attributes is the primary key of the first relation schema, which becomes a foreign key in the second relation. The second is the multi-valued attribute.
- The name of the second relation should capture the semantics of the multi-valued attribute.

Mapping E-R Diagrams to Relational Schemas (cont.)

Multi-valued Attributes Example:

Arrow represents a referential integrity constraint. In table employee-skill the attribute employee-id is a foreign key, i.e., it is a primary key in another table. The arrow links the attribute to the table where it is the primary key.

ERD



Mapping E-R Diagrams to Relational Schemas (cont.)

Multi-valued Attributes:

- Notice in the previous relational schemas constructed due to the multi-valued attribute *skill*, that the resulting relation schema *employee-skill* has only key attributes.
- Each tuple simply records the fact that a given employee possesses a certain skill.
- This provides the database designer the opportunity to suggest to the users that new attributes can be added to this relation.
  - For example, the attributes *years-experience* and/or *certification-date* might be appropriate new values to add to this relation.

Mapping E-R Diagrams to Relational Schemas

STEP 2: Mapping 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.
- A weak entity does not have a complete identifier, but must have an attribute called a partial identifier that permits distinguishing the various occurrences of the weak entity for each owner entity instance.
- The following procedure assumes that you have already created a relation schema corresponding to the identifying entity type. If you have not done this – do it now before proceeding.

Mapping E-R Diagrams to Relational Schemas

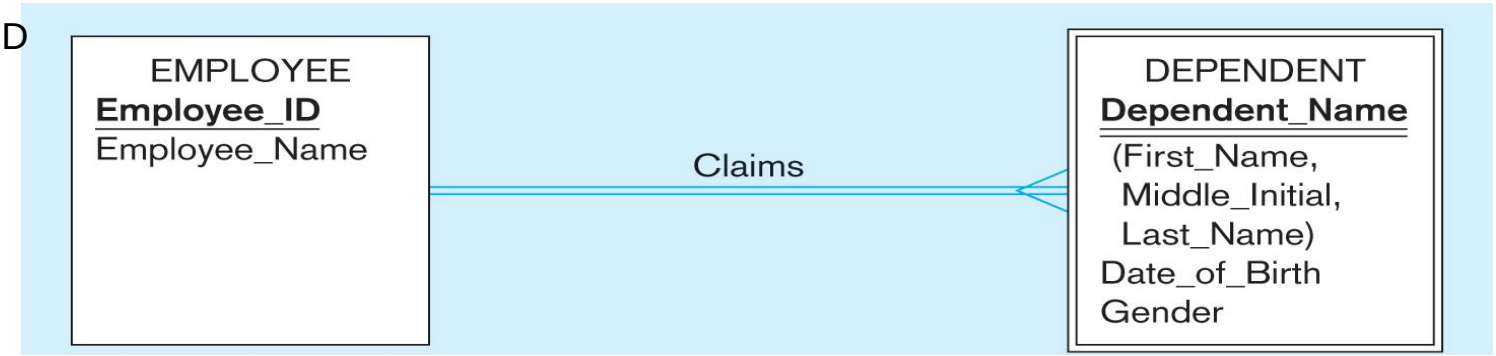
STEP 2: Mapping Weak Entities - continued

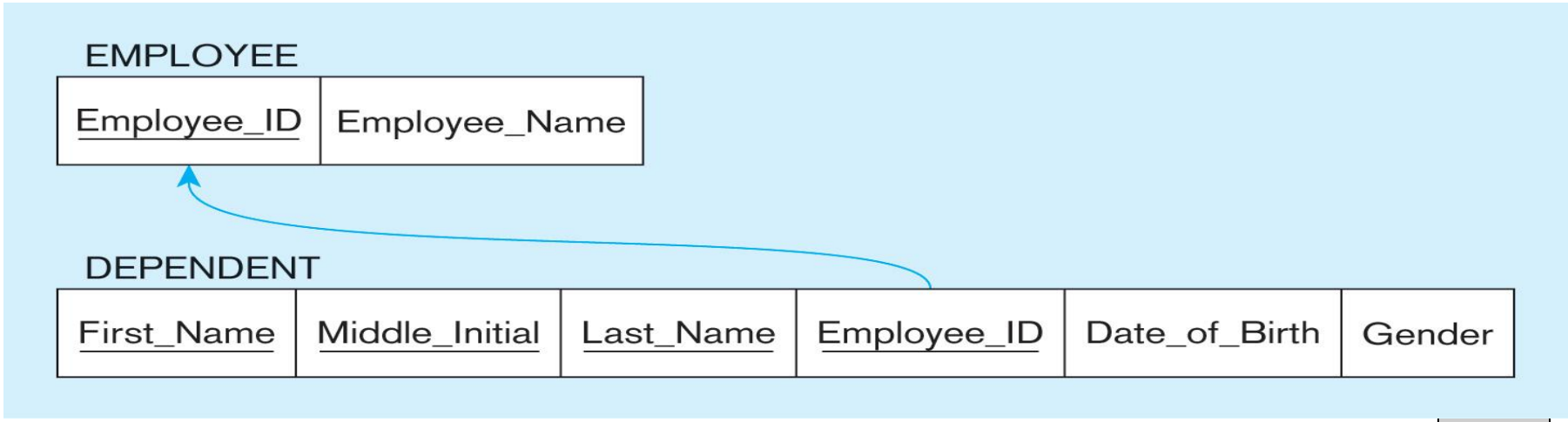
- For each weak entity type, create a new relation schema and include all of the simple attributes (or simple components of composite attributes) as attributes of this relation schema.
- Then include the primary key of the identifying relation as a foreign key attribute in this new relation schema.
- The primary key of the new relation schema is the combination of this primary key of the identifying relation and the partial identifier of the weak entity type.

Mapping E-R Diagrams to Relational Schemas

STEP 2: Mapping Weak Entities - EXAMPLE

ERD





Mapping E-R Diagrams to Relational Schemas

STEP 3: Mapping Binary Relationships

- The procedure for mapping relationships into the relational model depends on both the degree of the relationship (unary, binary, ternary, etc.) and the cardinalities of the relationships.
- We'll look at the most common and important of these over the next several pages. Note that binary 1:M and binary M:1 relationships are symmetric.

Mapping E-R Diagrams to Relational Schemas

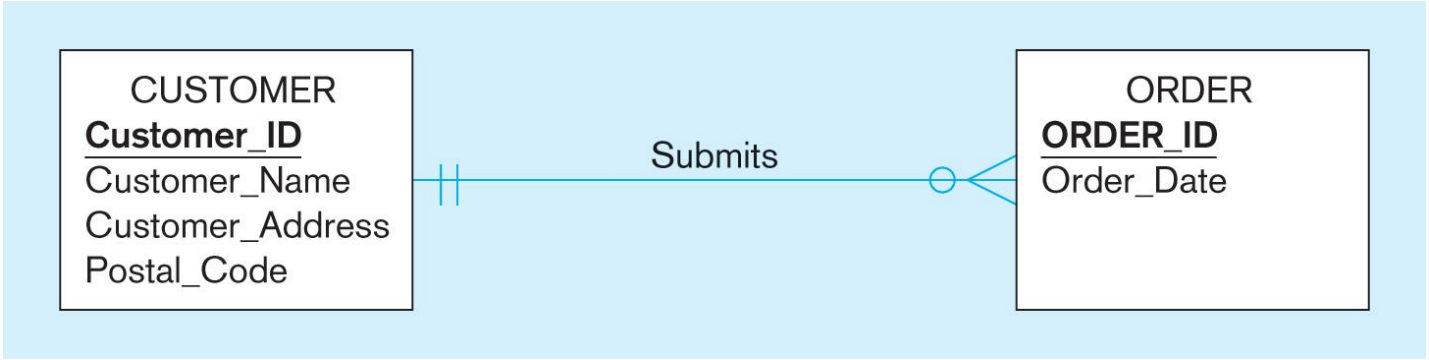
STEP 3: Binary 1:M Relationships

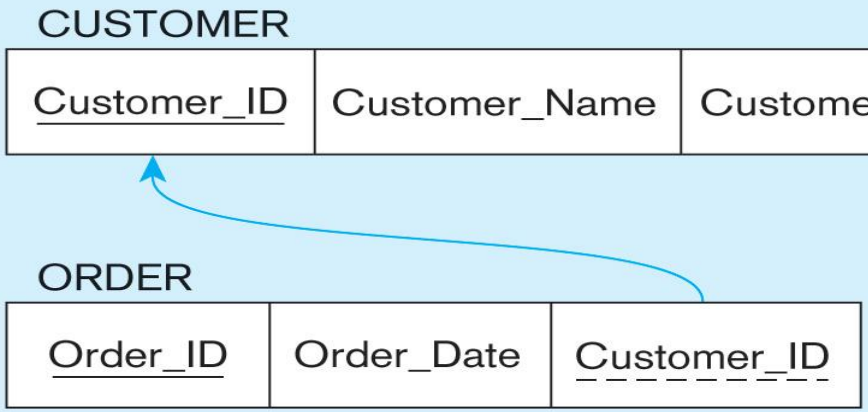
- For each binary 1:M relationship, first create a relation schema for each of the two entity types participating in the relationship using the procedure from Step 1.
- Next, include the primary key attribute (or attributes) of the entity on the one-side of the relationship as a foreign key in the relation that is on the many-side of the relationship. (The primary key migrates to the many-side.)

Mapping E-R Diagrams to Relational Schemas

STEP 3: Binary 1:M - EXAMPLE

ERD





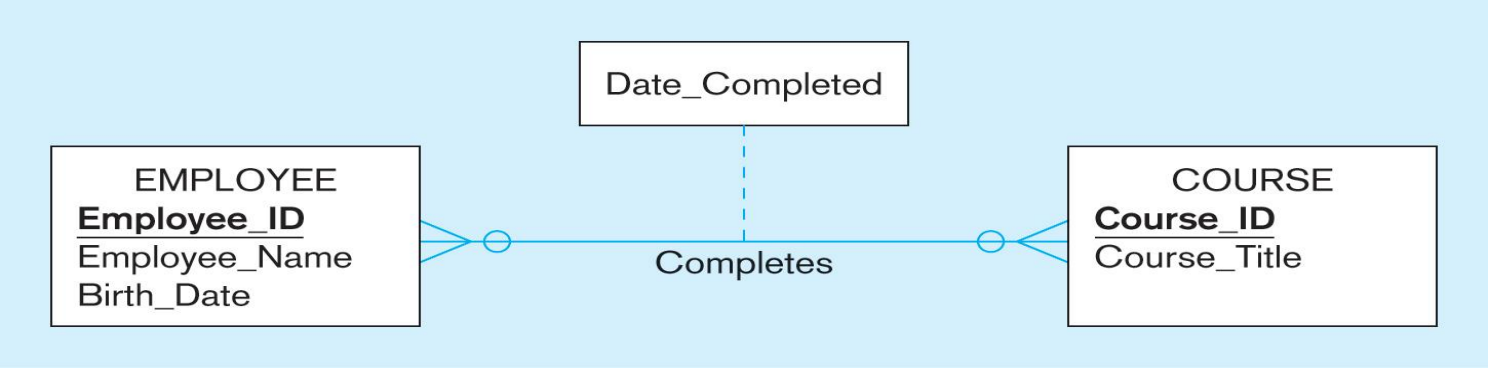
Mapping E-R Diagrams to Relational Schemas

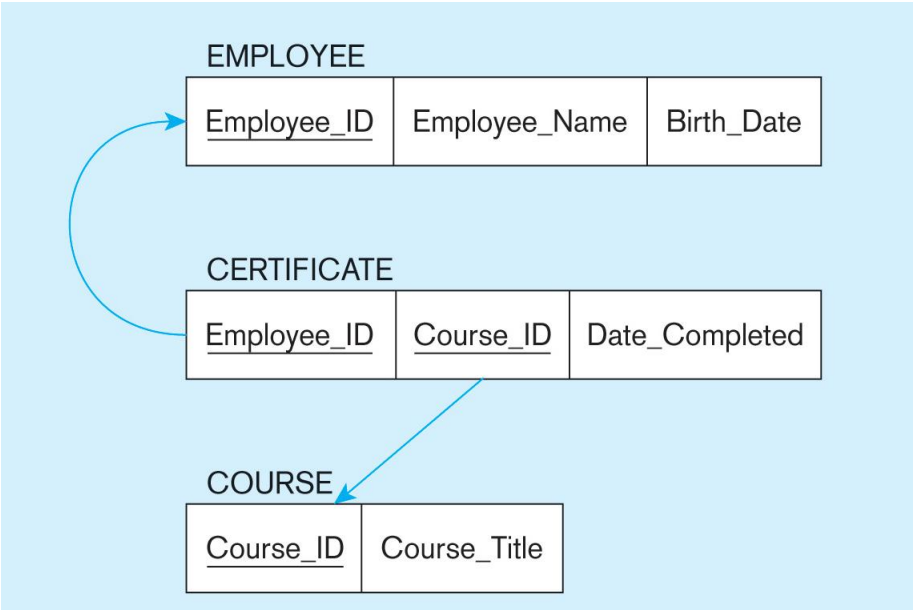
STEP 3: Binary M:M Relationships

- For each binary M:N relationship between two entity types A and B, first create a new relation schema C.
- Include as foreign key attributes in C the primary key for each of the two participating entity types A and B. These attributes becomes the primary key of relation schema C.
- Any non-key attributes that are associated with the M:N relationship between A and B are included in the relation schema C.

Mapping E-R Diagrams to Relational Schemas

STEP 3: Binary M:N - EXAMPLE





Mapping E-R Diagrams to Relational Schemas

STEP 3: Binary 1:1 Relationships

- The process of mapping such a relationship onto relation schemas requires two steps.

  1. Two relations are created, one for each of the participating entity types.
  2. The primary key of one of the relations is included as a foreign key in the other relation.

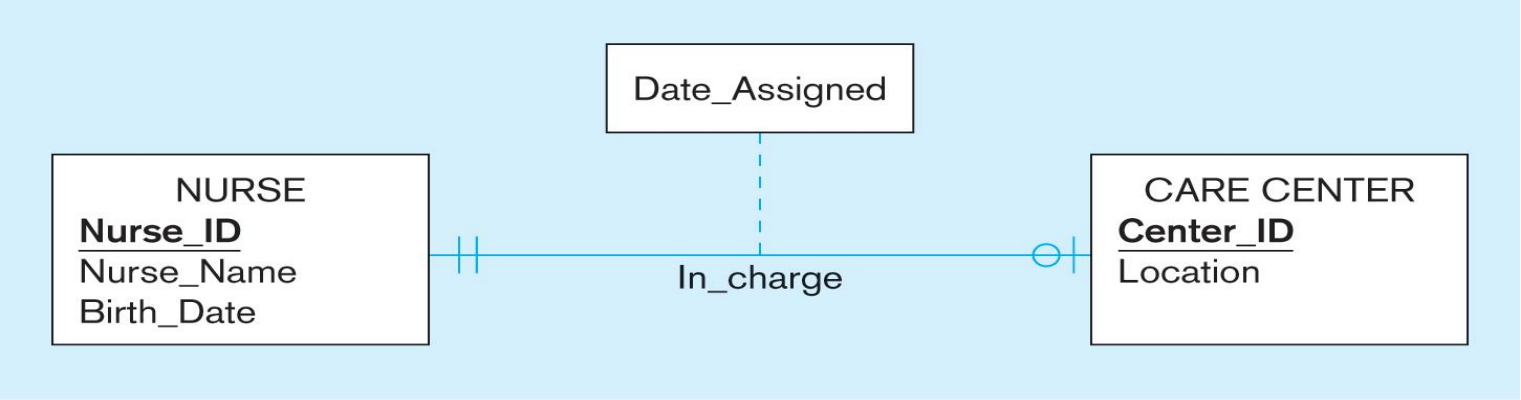
- In a 1:1 relationship, the association in one direction is nearly always an optional one, while the association in the other direction is mandatory (recall participation constraints).

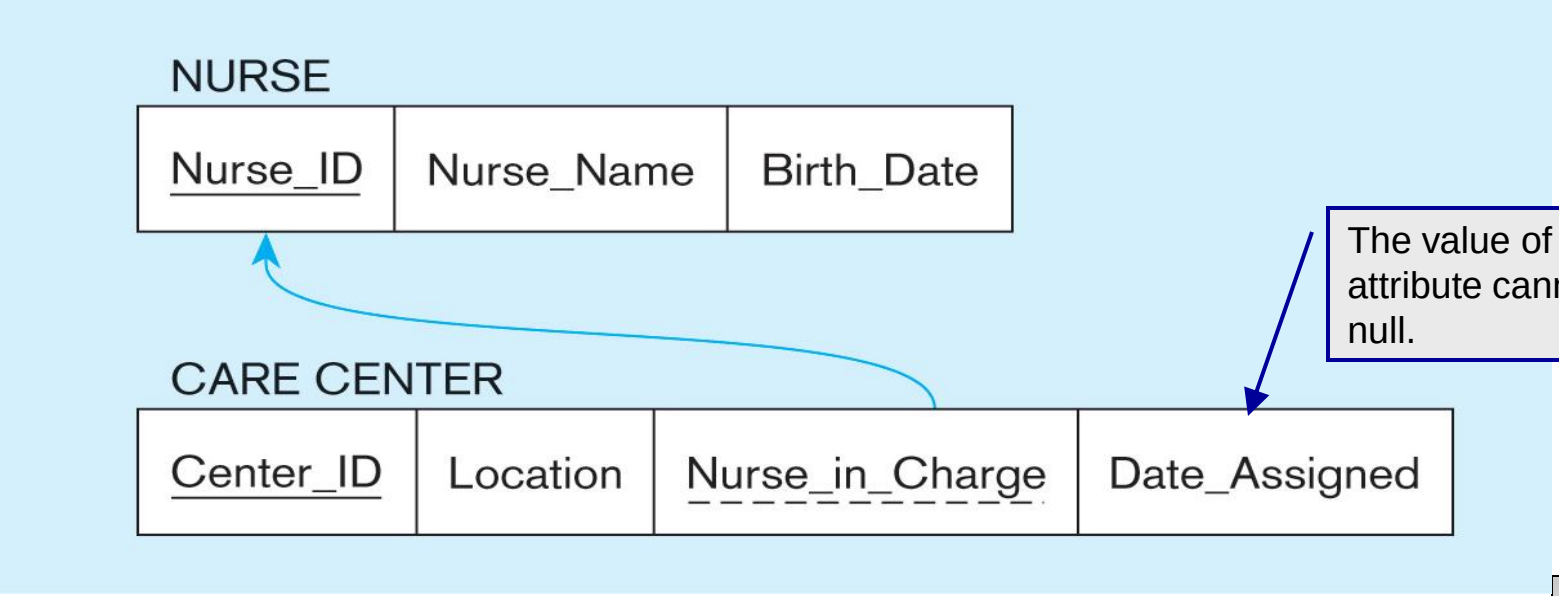
  – You should include in the relation on the optional side of the relationship the foreign key of the entity type that has the mandatory participation in the 1:1 relationship. This approach will avoid the need to store null values in the foreign key attribute.

- Any attributes associated with the relationship itself are also included in the same relation as the foreign key.

Mapping E-R Diagrams to Relational Schemas

STEP 3: Binary 1:1 - EXAMPLE





Mapping E-R Diagrams to Relational Schemas

STEP 4: Mapping Associative Entities

- Mapping an associative entity to a relation schema is similar to the procedure followed for mapping a M:N relationship. Two steps are required:

1. Create three relation schemas, one for each of the two participating entity types, and the third for the associative entity. The relation formed from the associative entity is called the *associative relation*.

2. The actions in this step depend on whether or not the associative entity was assigned an identifier in the E-R diagram. Two cases exist:

   – An identifier was not assigned.

   – An identifier was assigned.

We'll examine each case separately.

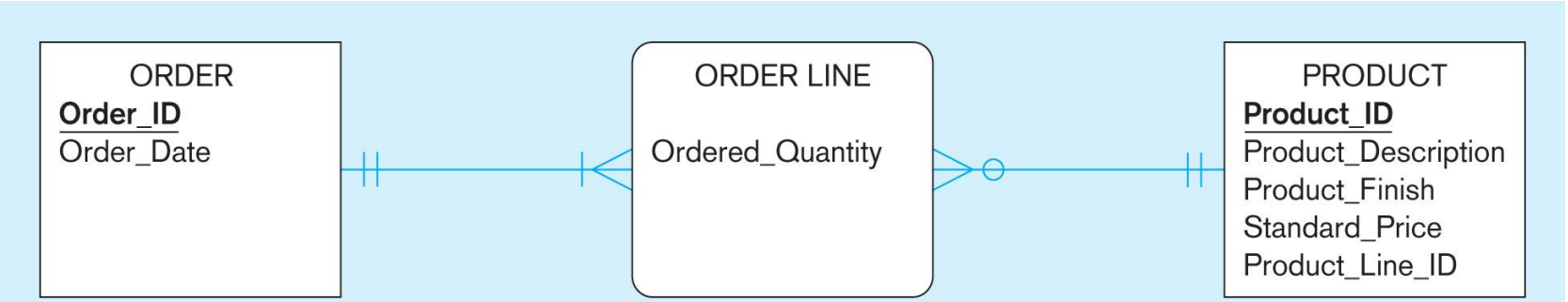
Mapping E-R Diagrams to Relational Schemas

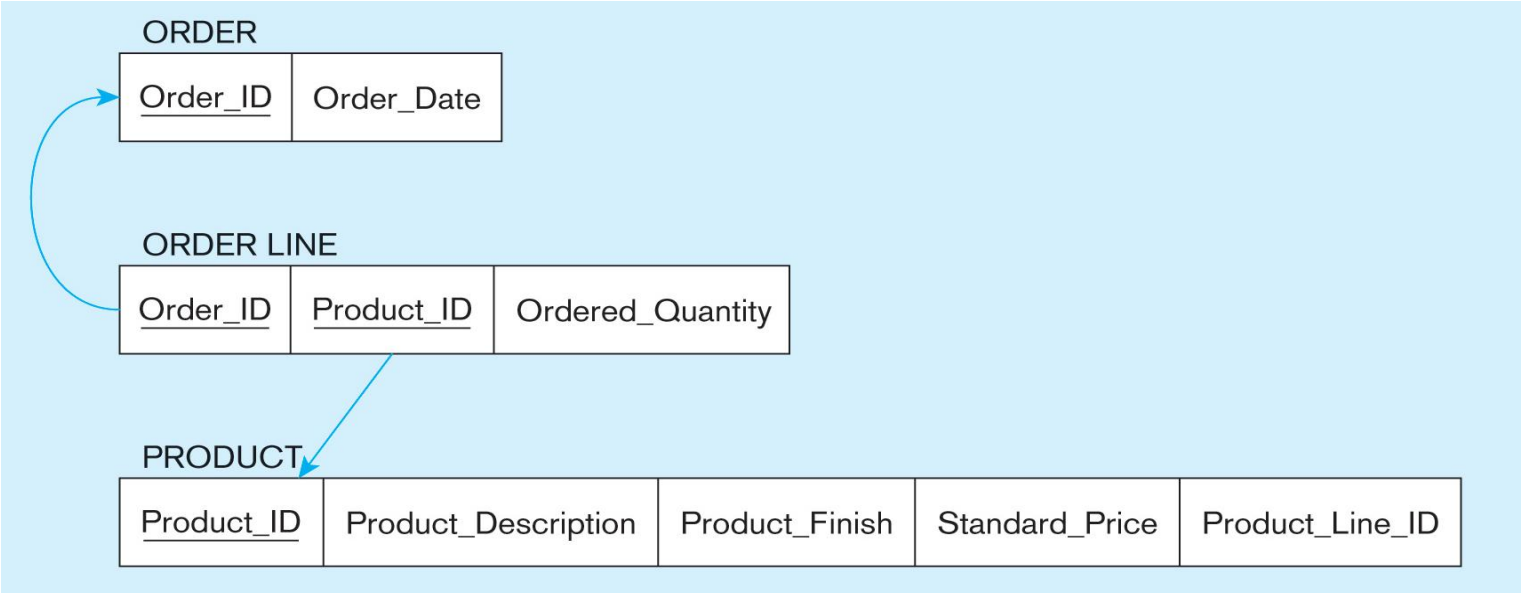
STEP 4: Mapping Associative Entities – No Identifier Assigned

- If an identifier was not assigned, the default primary key for the associative relation consists of the two primary key attributes from the other two relations.
- These attributes are then foreign keys that reference the other two relations.
- An example of this is shown on the next page, but note the similarity of this example to that of the M:N relationship case.

Mapping E-R Diagrams to Relational Schemas

STEP 4: Associative Entity: EXAMPLE – no identifier assigned





Mapping E-R Diagrams to Relational Schemas

STEP 4: Mapping Associative Entities – Identifier Assigned

- Sometimes the data modeler will assign an identifier (called a surrogate identifier or key) to the associative entity type on the ERD.
- There are two basic reasons this may occur:
  1. The associative entity type has a natural identifier that is familiar to end users.
  2. The default identifier (consisting of the identifiers for each of the participating entity types) may not uniquely identify instances of the associative entity.
- In either case, the process for mapping the associative entity is modified as follows:

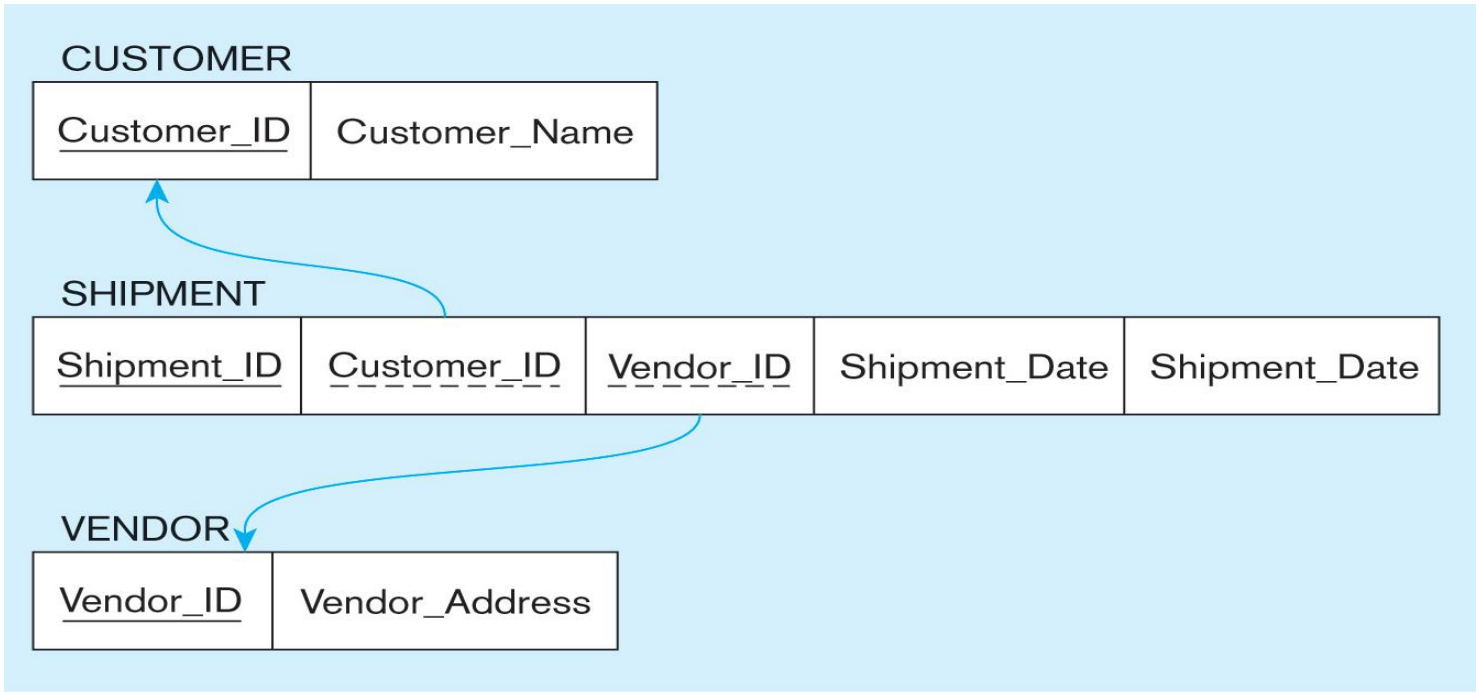
Mapping E-R Diagrams to Relational Schemas

STEP 4: Mapping Associative Entities – Identifier Assigned

- As before, a new associative relation is created to represent the associative entity.
- The primary key for the associative relation is the identifier assigned on the ERD (rather than the default key as in the previous case).
- The primary keys for the two participating entity types are then included as foreign keys in the associative relation.
- An example appears on the next page.

Mapping E-R Diagrams to Relational Schemas

STEP 4: Associative Entity: EXAMPLE – identifier assigned



Mapping E-R Diagrams to Relational Schemas

STEP 5: Mapping Unary (recursive) Relationships

- Recall that a recursive relationship is defined as a relationship between instances of a single entity type.
- The two most important cases of unary relationships are the 1:M and M:M cardinalities.
- We'll again look at these two cases separately as they are handled somewhat differently.

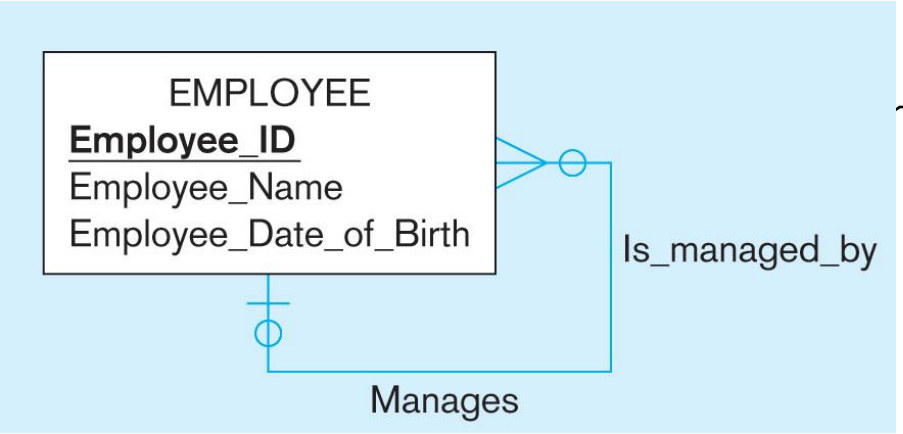
Mapping E-R Diagrams to Relational Schemas

STEP 5: Mapping Recursive Relationships – 1:M Case

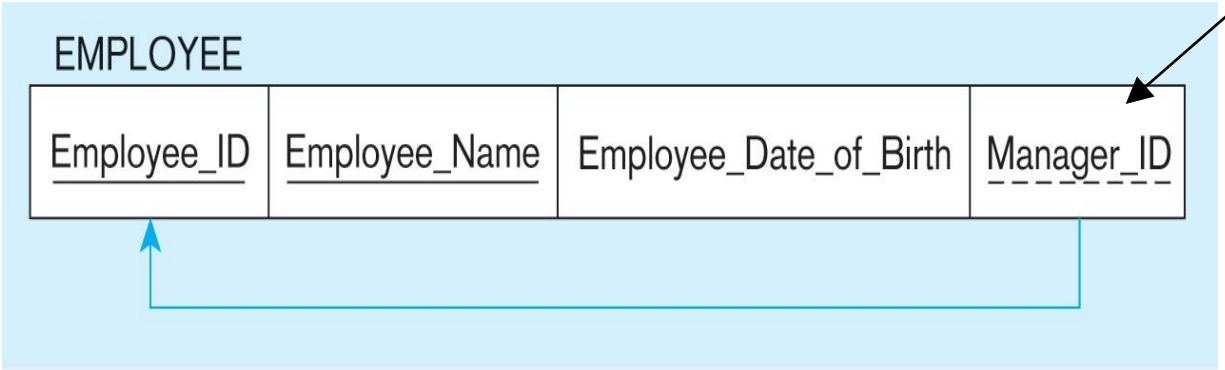
- The entity type in the unary relationship is mapped onto a relation schema using the procedure described in Step 1.
- Next, a foreign key attribute is added *within* the same relation that references the primary key values (this foreign key must have the same domain as the primary key).
- A recursive foreign key is a foreign key in a relation that references the primary key values of that same relation.

Mapping E-R Diagrams to Relational Schemas

STEP 5: Mapping Recursive Relationships: EXAMPLE – 1:M



every employee has exactly one manager, a given employee may manage many employees



foreign key

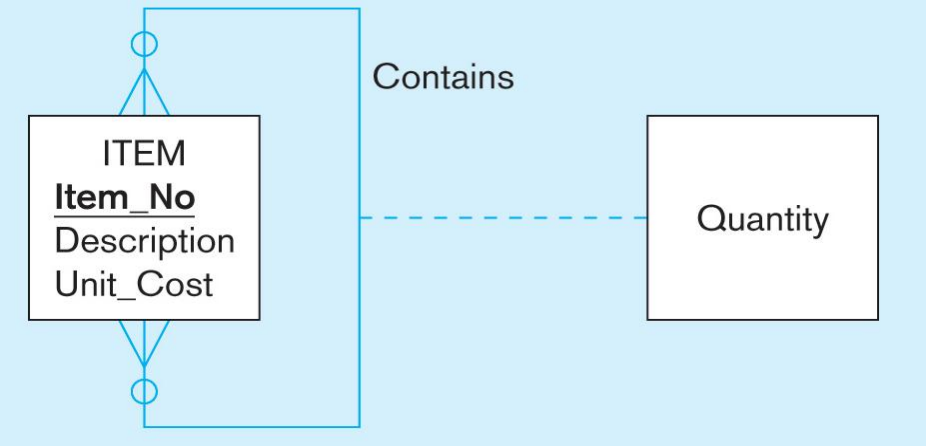
Mapping E-R Diagrams to Relational Schemas

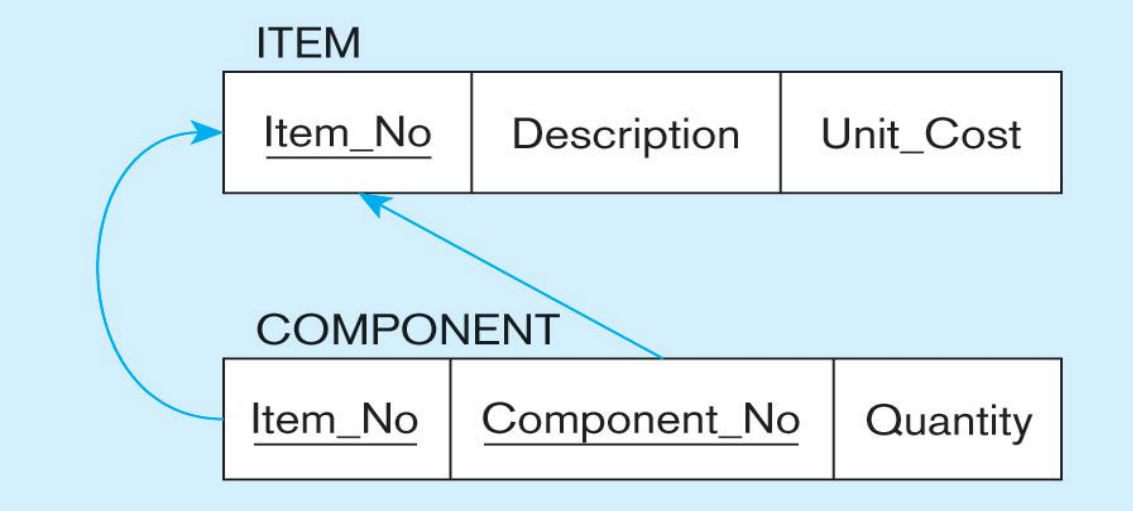
STEP 5: Mapping Recursive Relationships – M:N Case

- With this type of recursive relationship, two relation schemas are created: one to represent the entity type and the other an associative relation to represent the M:N relationship itself.
- The primary key of the associative relation consists of two attributes. These attributes (which do not necessarily have the same name) both take their values from the primary keys of the other relation.
- Any non-key attribute of the relationship is included in the associative relation.
- The example on the next page illustrates such a case representing a bill of materials relationship among items that are assembled from other items or components.

Mapping E-R Diagrams to Relational Schemas

STEP 5: Mapping Recursive Relationships: EXAMPLE – M:N





Mapping E-R Diagrams to Relational Schemas

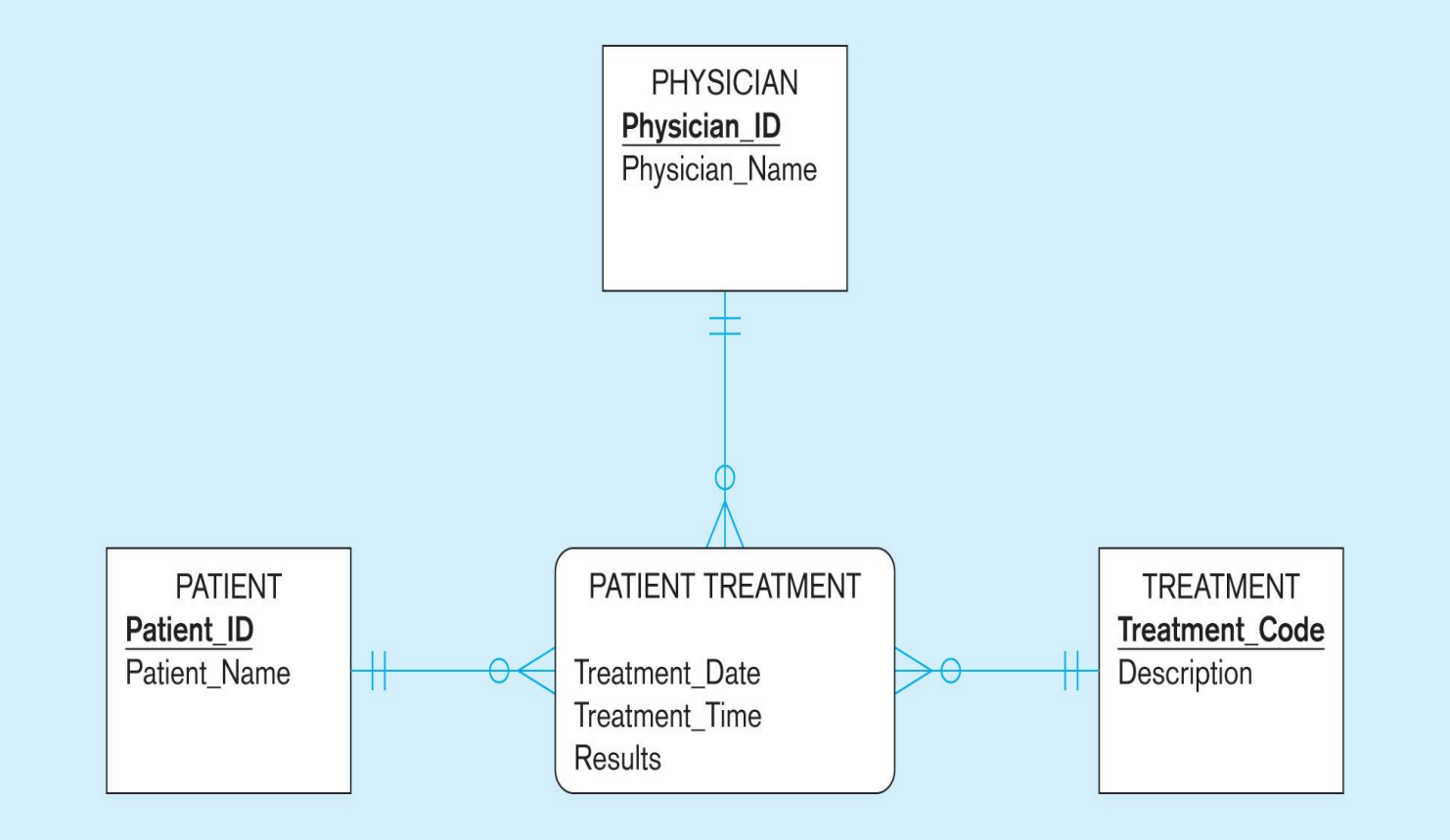
STEP 6: Mapping Ternary (and n-ary) Relationships

- It is strongly recommended that all ternary (or higher) relationships be converted associative entities before proceeding further
- To map an associative entity type that links three regular entity types, create a new associative relation.
- The default primary key of this relation consists of the three primary key attributes for the participating entity types (in some cases additional attributes are required to form a unique primary key). These attributes then act in the role of foreign keys that reference the individual primary keys of the participating entity types.
- Any attributes of the associative entity type become attributes in the new associative relation.

Mapping E-R Diagrams to Relational Schemas

STEP 6: Mapping Ternary (and n-ary) Relationships - EXAMPLE

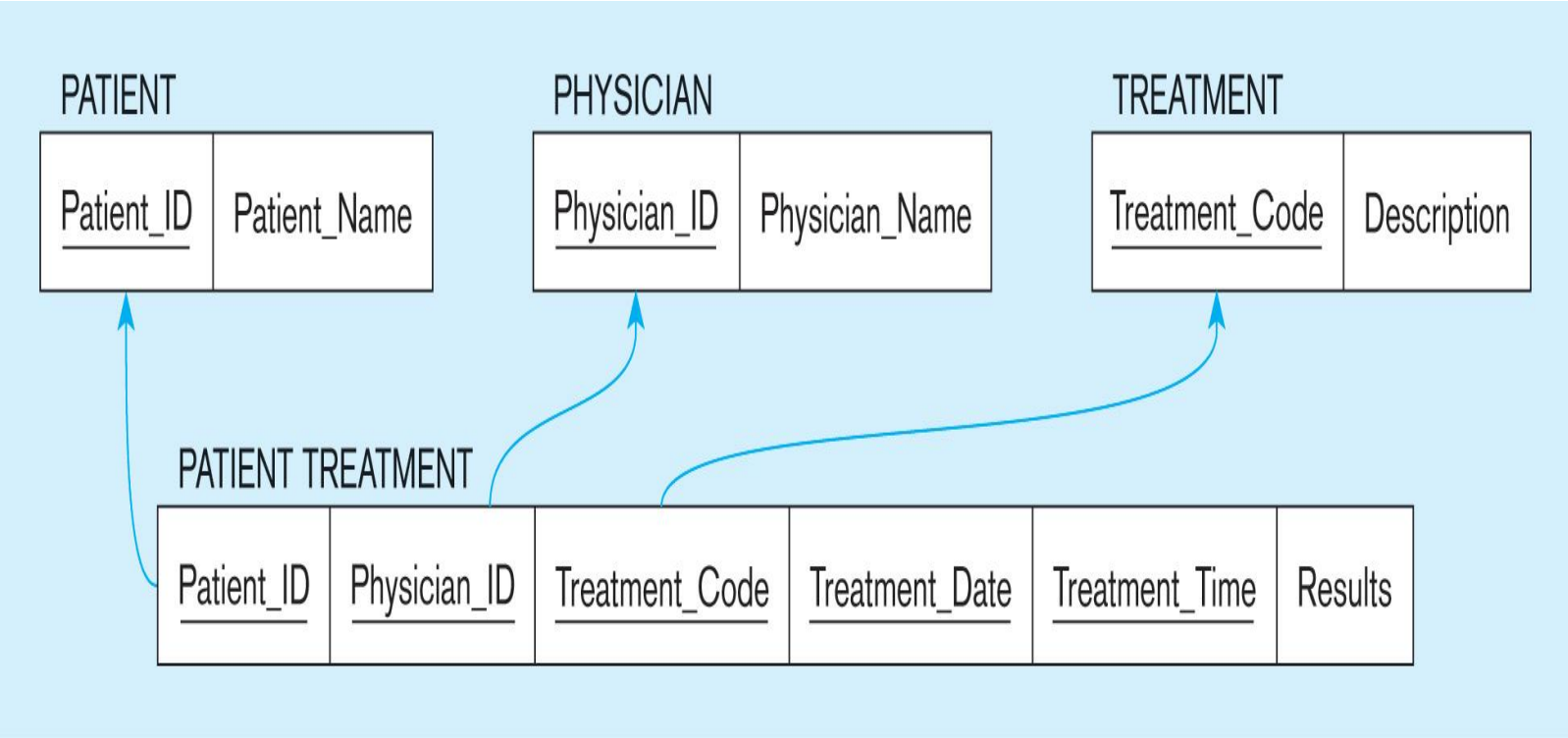
ERD for ternary relationship with associative entity



Mapping E-R Diagrams to Relational Schemas

STEP 6: Mapping Ternary (and n-ary) Relationships - EXAMPLE

Resulting relational schemas



Mapping E-R Diagrams to Relational Schemas

STEP 7: Mapping Supertype/Subtype Relationships

- The relational data model does not directly support supertype/subtype relationships. Fortunately, there are various strategies that database designers can use to represent these relationships with the relational data model.
- We'll examine one of the more common techniques that has been used for modeling supertype/subtype relationships.

Mapping E-R Diagrams to Relational Schemas (cont.)

STEP 7: Mapping Supertype/Subtype Relationships

1. Create a separate relation schema for the supertype and for each of its subtypes.
2. Assign to the relation schema created for the supertype the attributes that are common to all members of the supertype, including the primary keys.
3. Assign to the relation schema 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 discriminator.

Mapping E-R Diagrams to Relational Schemas (cont.)

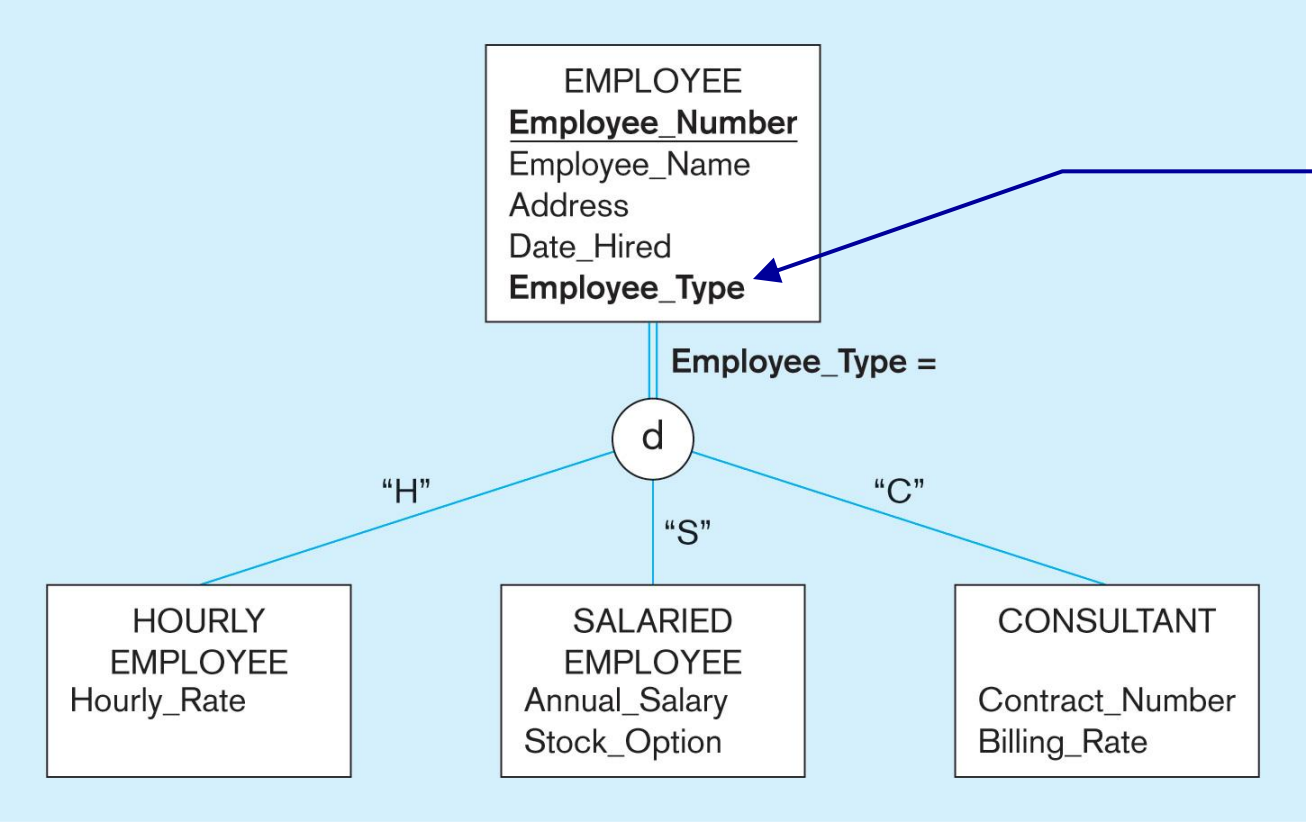
STEP 7: Mapping Supertype/Subtype Relationships

Defining subtype discriminators

- Given a supertype/subtype relationship, consider the problem of inserting a new instance of the supertype. Into which of the subtypes (if any) should this instance be inserted?
- A common approach uses a subtype discriminator. A subtype discriminator is an attribute of the supertype whose values determine the target subtype or subtypes (used when subclass membership is predicate based).
- Two cases arise: disjoint subtypes and overlapping subtypes.
  - There is no difference in the conversion technique – however, disjoint subtypes are typically differentiated by a simple attribute whereas overlapping types are typically differentiated by a composite (and/or) multi-valued attribute.

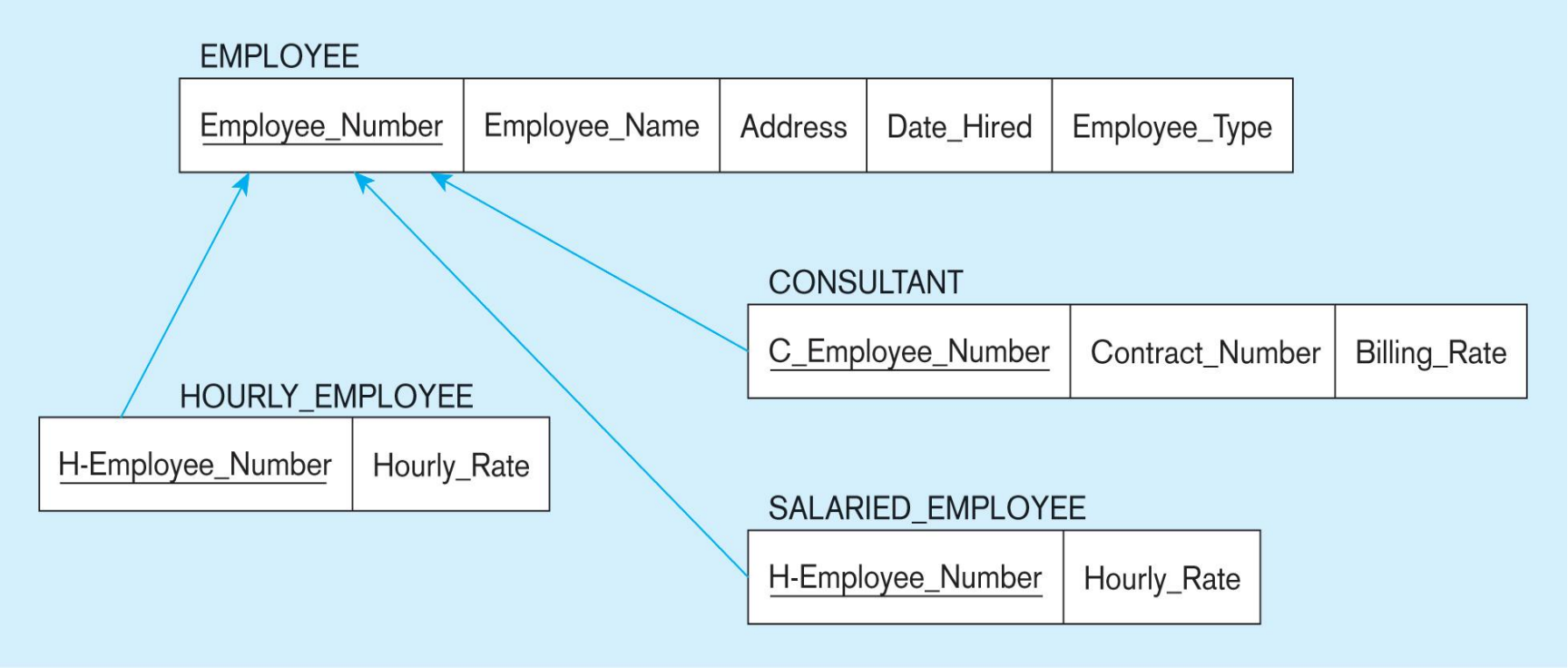
Mapping E-R Diagrams to Relational Schemas

EXAMPLE – Subtype discriminators in ERDs: disjoint subtypes



Mapping E-R Diagrams to Relational Schemas

EXAMPLE – Subtype discriminators in ERDs: disjoint subtypes



Summary of ERD To Relation Conversions

ER Structure	Relation Representation
Regular entity	Create a relation with primary key and non key attributes.
Composite attributes	Each component attribute of the composite becomes a separate attribute.
Multi-valued attributes	Create a separate relation for the multi-valued attribute with composite primary key, including primary key of the entity.
Weak entity	Create a relation with a composite primary key (which includes the primary key of the entity on which the weak entity depends) and nonkey attributes.
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.

Summary of ERD To Relation Conversions (cont.)

ER Structure	Relation Representation
Binary or Unary M:M relationship -or – -Associative entity without an identifier	Create a relation with composite primary key using the primary keys of the related entities plus any nonkey attributes of the relationship or associative entity.
Binary or Unary 1:1 relationship	Place the primary key of either entity in the relation for the other entity or do this for both entities; 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.
Binary or Unary M:M relationship or associative entity with an identifier	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.

Summary of ERD To Relation Conversions (cont.)

ER Structure	Relation Representation
Ternary and n-ary relationships	Same as binary M:M relationships described 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 key, the primary keys of the associated entities are included as foreign keys in the relation for the relationship or associative entity.
Supertype/Subtype relationship	Create a relation for the superclass, which contains the primary key and all nonkey attributes in common with all subclasses, plus create a separate relation for each subclass with the same primary key, but with only the nonkey attributes related to that subclass.