Database Design

Relational Schemas

Database Design

1. *Requirements Analysis:* in this step, we must point out

2. *Conceptual Database Design:* develop a high-level description of the data to be stored in the database, along with the constraints that are known to hold on this data.

3. *Logical Database Design*: convert the conceptual database design into a database schema within the data model of the chosen DBMS.

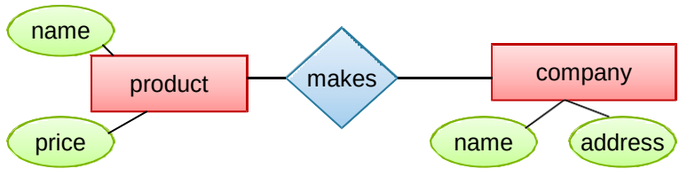
4. *Schema Refinement*: the schemas developed in step 3 are analyzed for potential problems, then are *normalized*.

5. *Physical Database Design:* potential workloads and access patterns are simulated to identify potential weaknesses in the conceptual database. This will often cause the creation of additional indices and/or clustering relations.

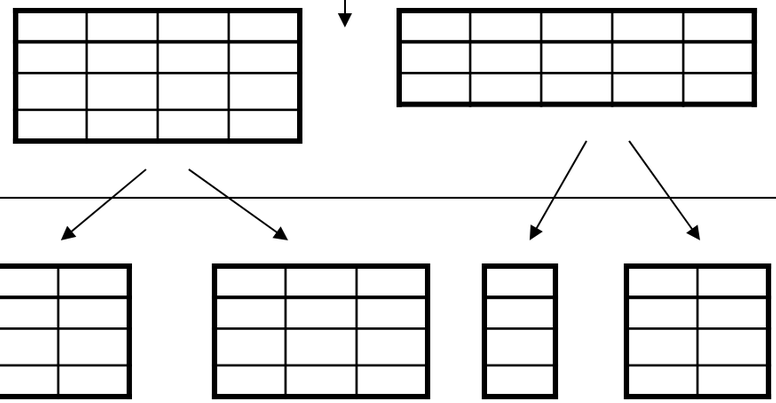
6. *Security Design:* Different user groups are identified and their different roles are analyzed so that access patterns to the data can be defined.

Database Design Process

Conceptual Model:



Relational Model:
Tables + constraints
And also functional dep.

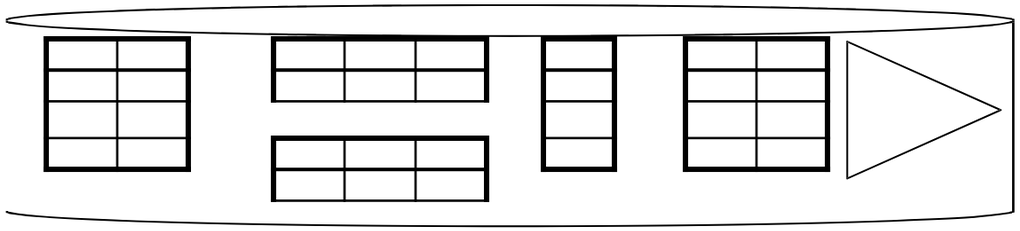


Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

Physical Schema



Relational Data model

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.

Relational Data model

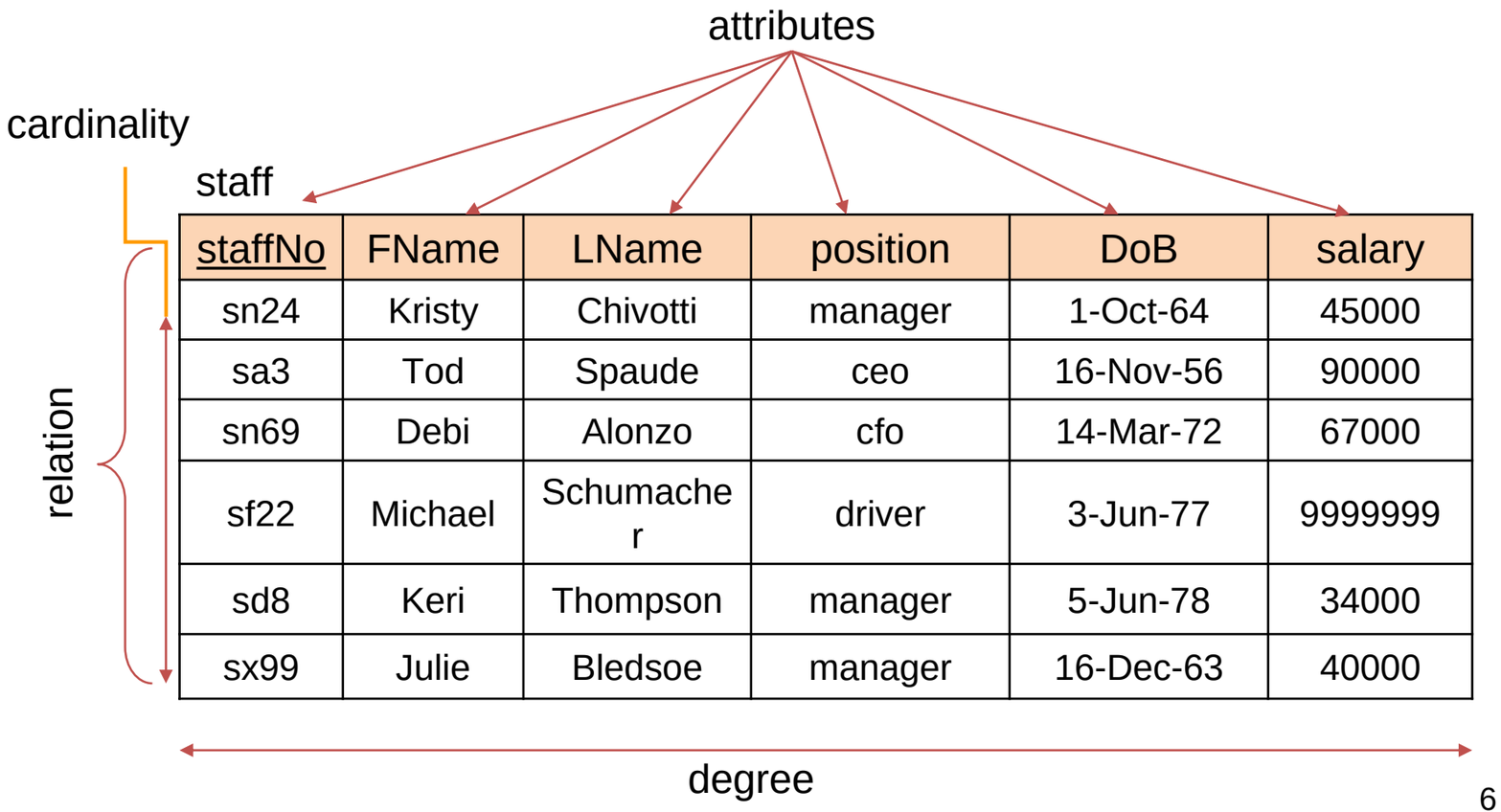
Tuple: A tuple is a row of a relation. Tuples can appear in any order in a relation and the relation will remain the same, and therefore convey the same meaning.

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 of Relation



staff

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

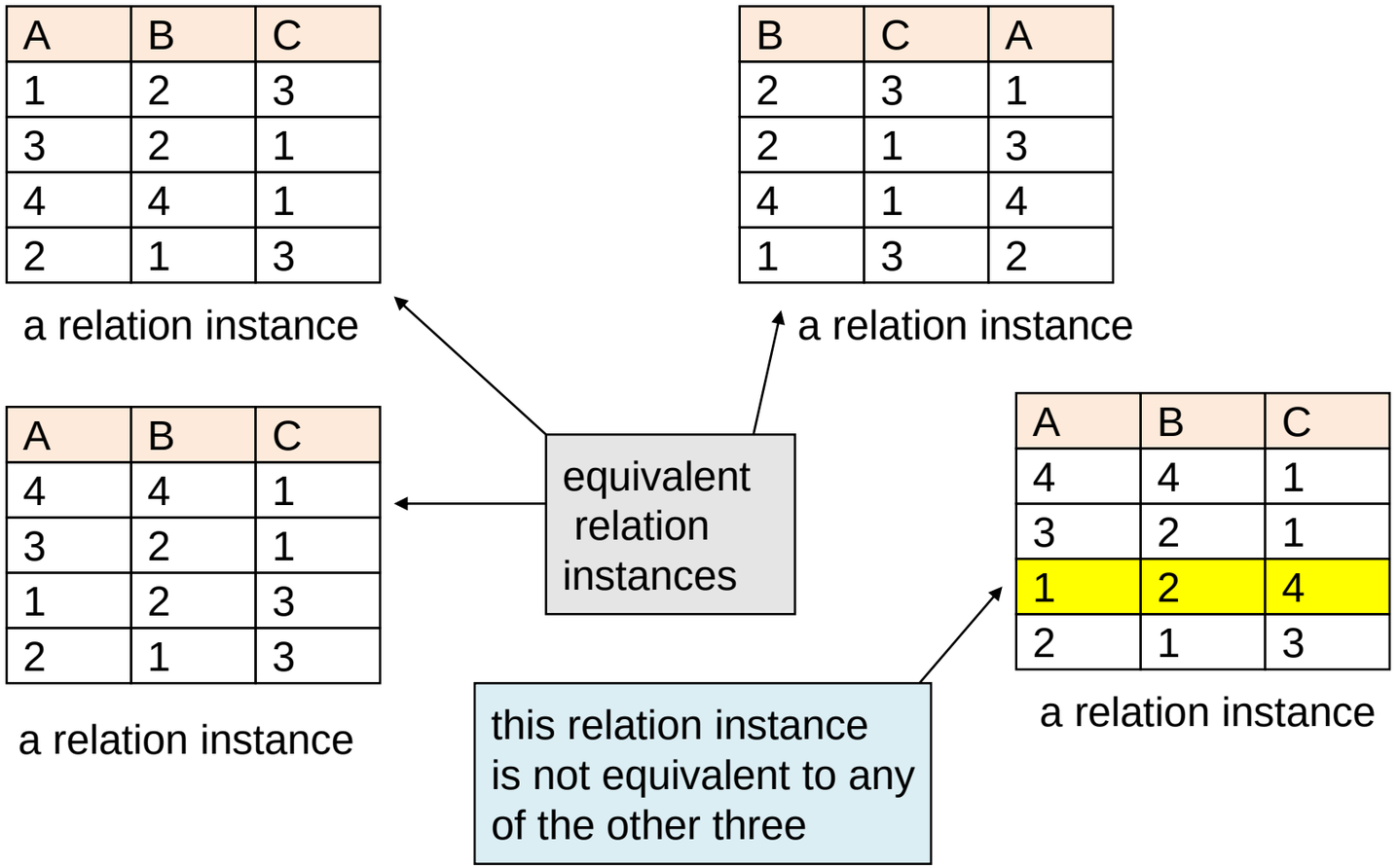
Example Domain Definitions

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}

Alternative Terminology for Relational model

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

Equivalent Relations



Logical Design

During logical design you transform the conceptual design into relational database schemas.

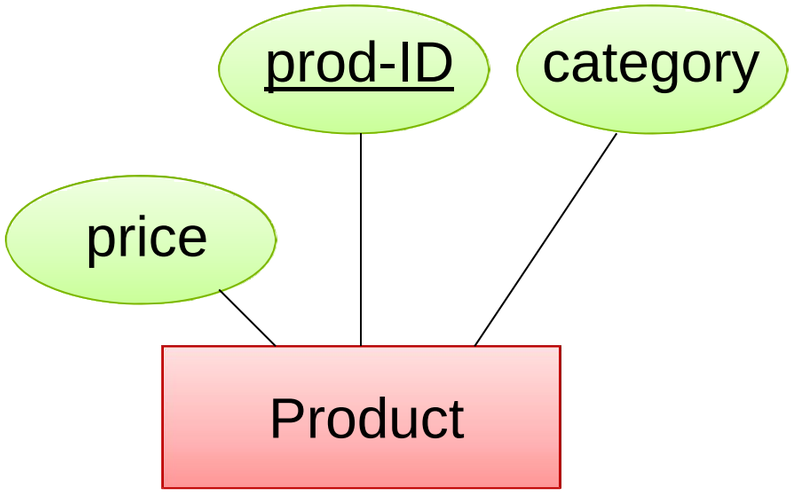
The inputs to the process are the E-R diagrams and the outputs are the relational schemas.

Mapping the E-R diagrams to relations is a relatively straightforward process with a well-defined set of rules.

Mapping Strong Entity Sets

- 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 Strong Entity Sets



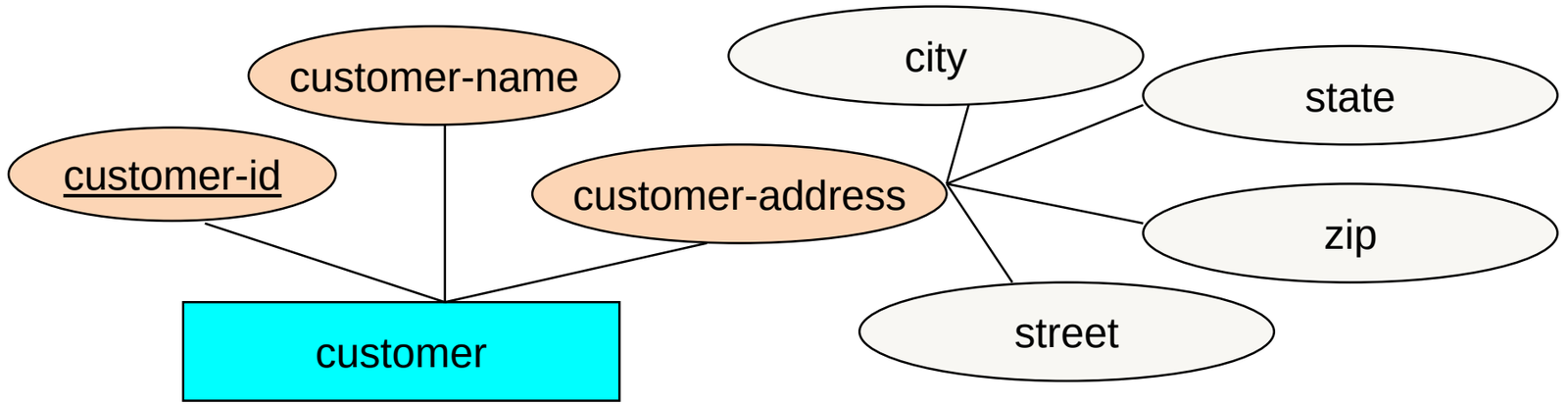
Product(prod-ID, category, price)

prod-ID	category	price
Gizmo55	Camera	99.99
Pokemn19	Toy	29.99

Mapping 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.

E-R diagram



Relation

customer

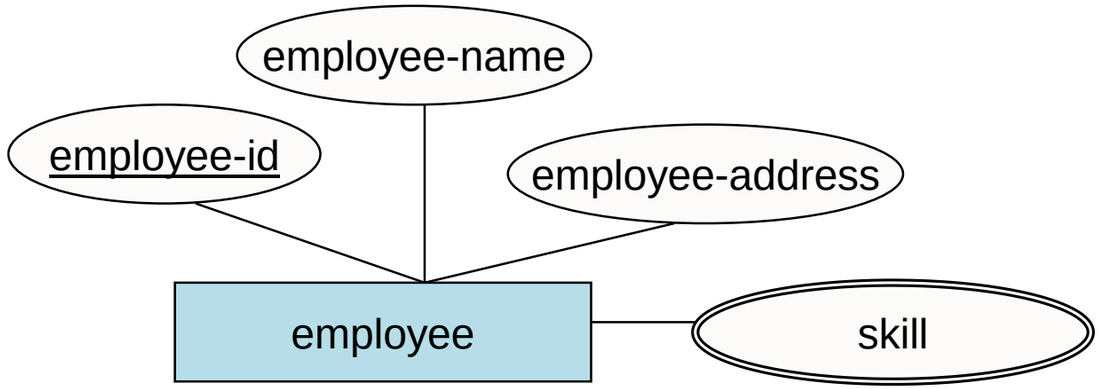
customer-id	customer-name	street	city	state	zip

Mapping Multi-valued Attributes

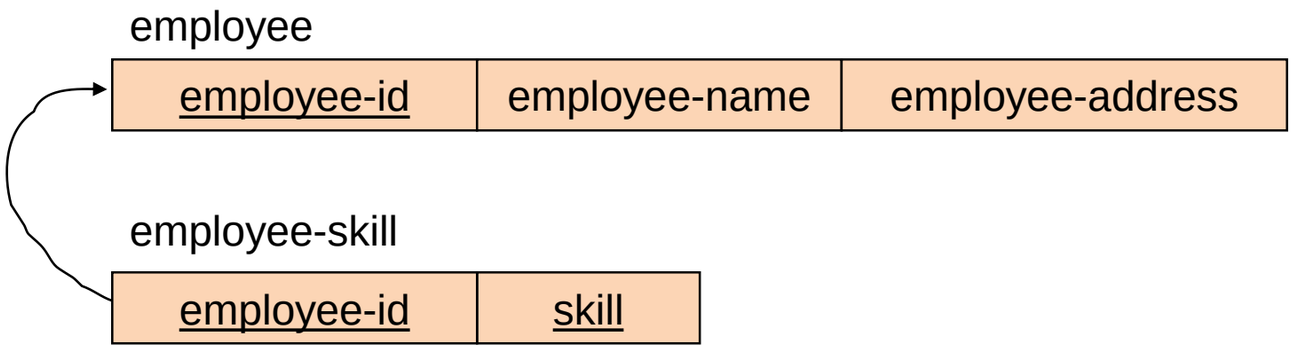
- When a regular entity type contains a multi-valued attribute, two new relation schemas (rather than one) are created.
- The first relation schema contains all of the attributes of the entity type except the multi-valued attribute.
- The second relation schema contains two attributes that form the primary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

Mapping Multi-valued Attributes

E-R diagram



Resulting relation schemas

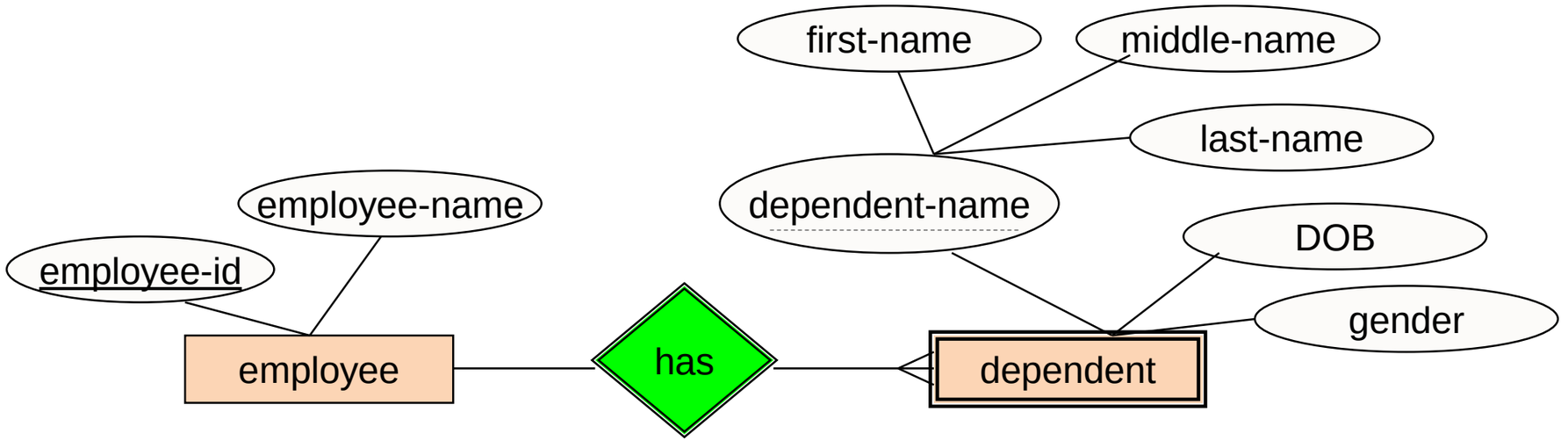


Mapping Weak Entity Sets

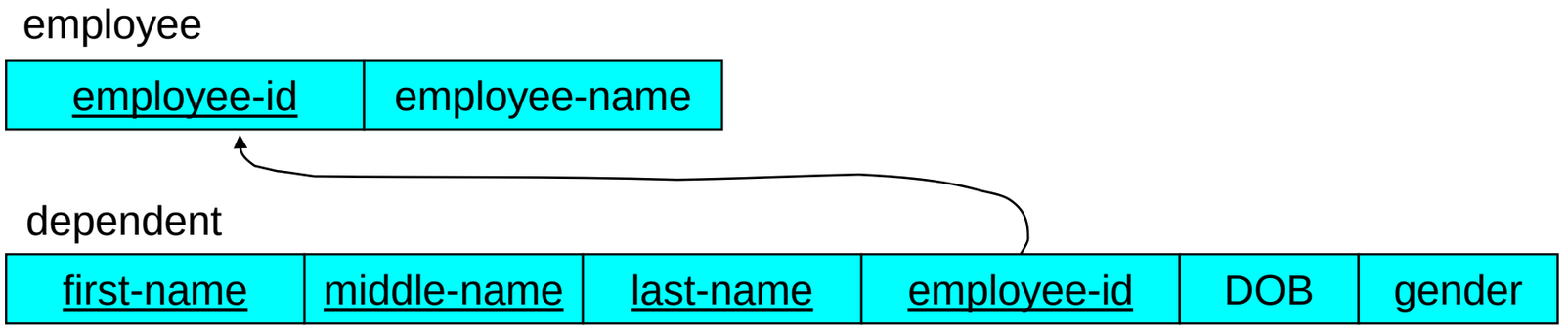
- A weak entity does not have a complete identifier, but must have an attribute called a partial identifier that permits distinguishing the various occurrences of the weak entity for each owner entity instance.
- For each weak entity type, create a new relation schema and include all of the simple attributes (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 Weak Entity Sets

E-R diagram



Resulting relation schemas



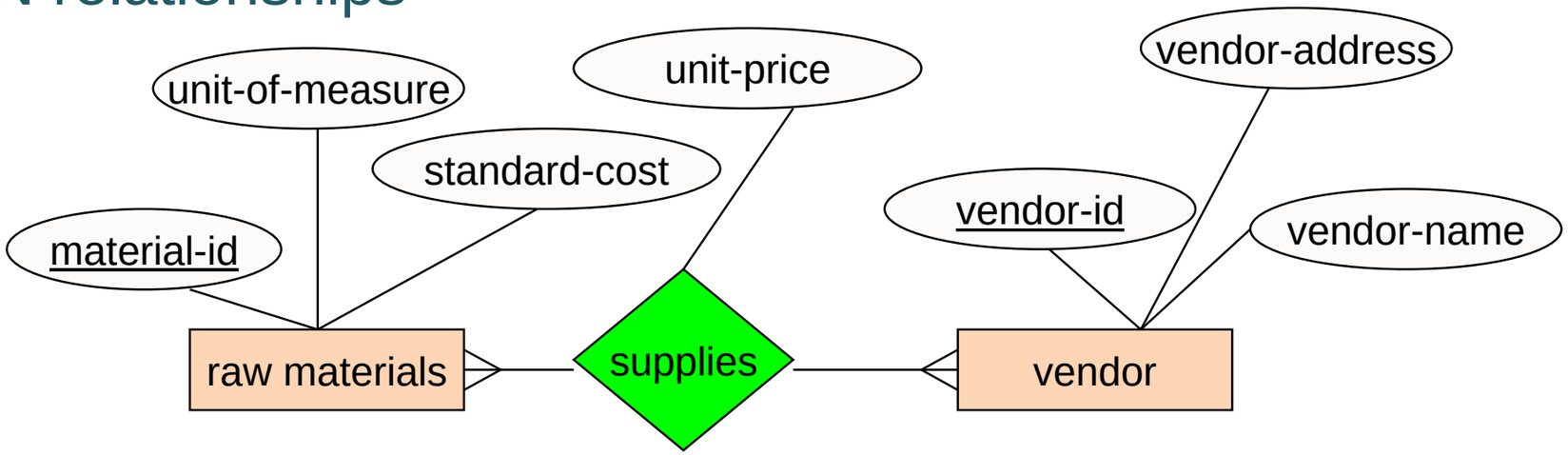
Mapping Binary Relationship

- Binary M:N 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.

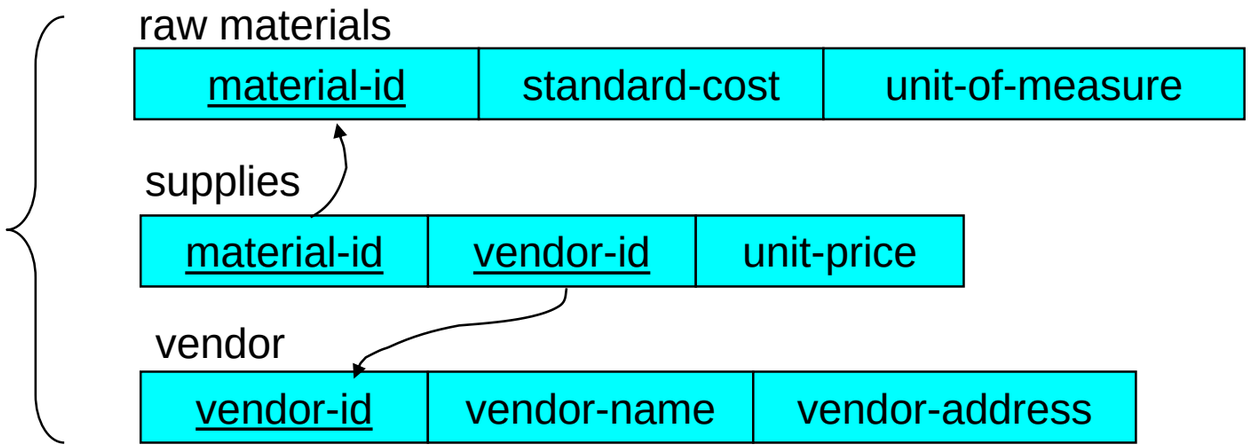
Mapping Binary Relationship

- Binary M:N relationships

E-R diagram



Resulting relation schemas



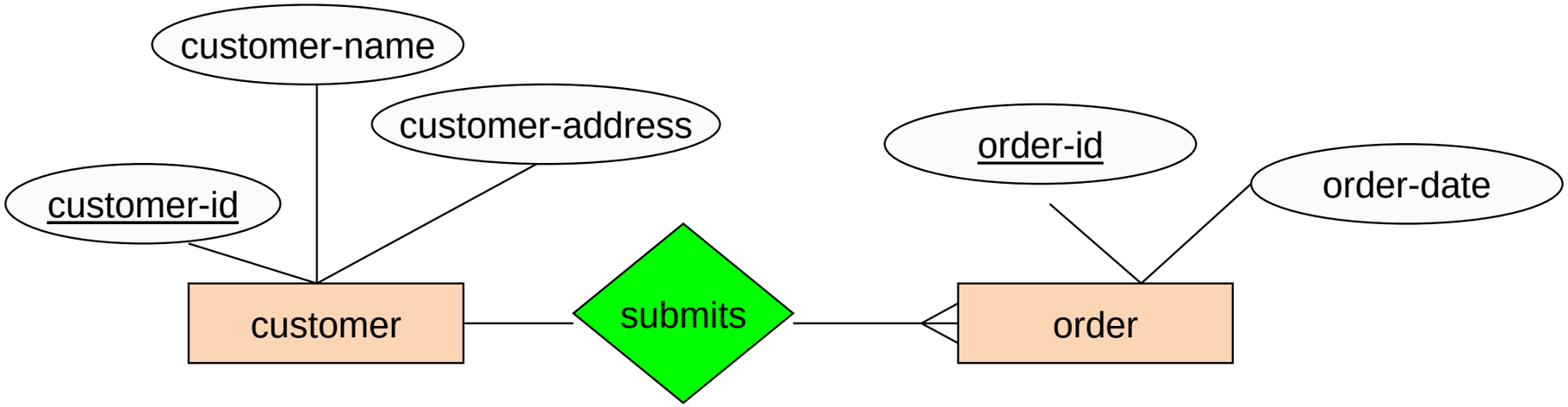
Mapping Binary Relationship

- Binary 1:M relationships
- First, create a relation schema for each of the two entity types participating in the relationship using the procedure from previous sections.
- 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.)

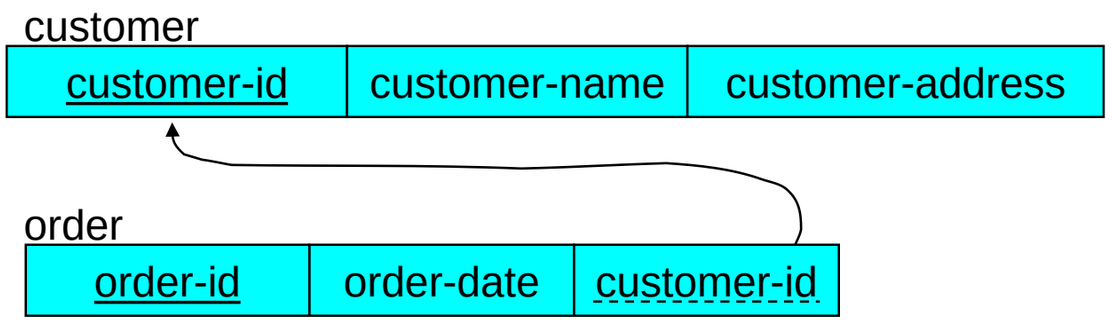
Mapping Binary Relationship

- Binary 1:M relationships

E-R diagram



Resulting relation schemas



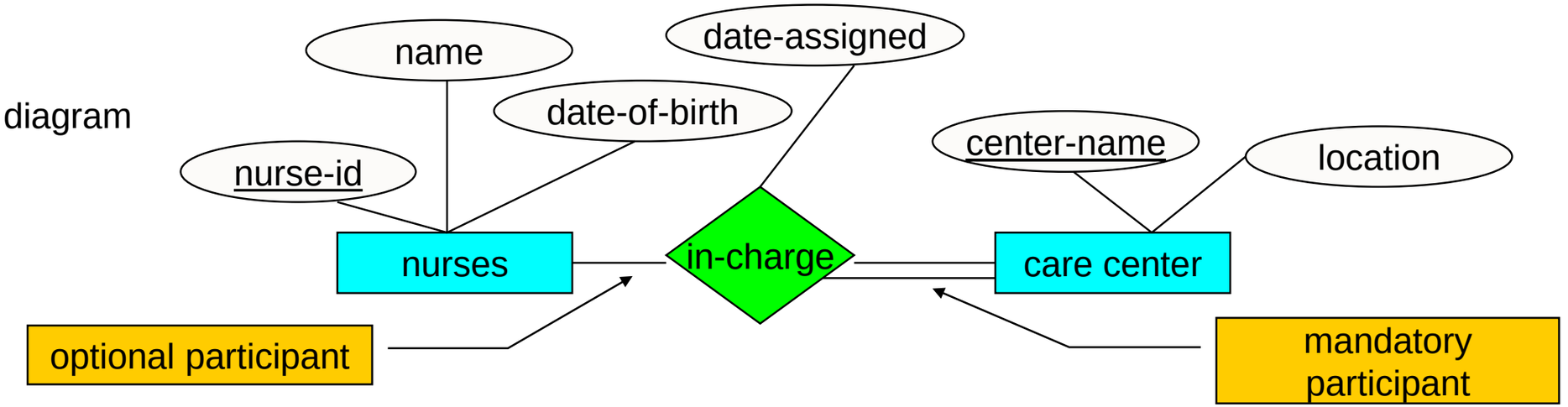
Mapping Binary Relationship

- Binary 1:1 relationships
- The process of mapping such a relationship onto relation schemas requires two steps.
  - Two relations are created, one for each of the participating entity types.
  - 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.

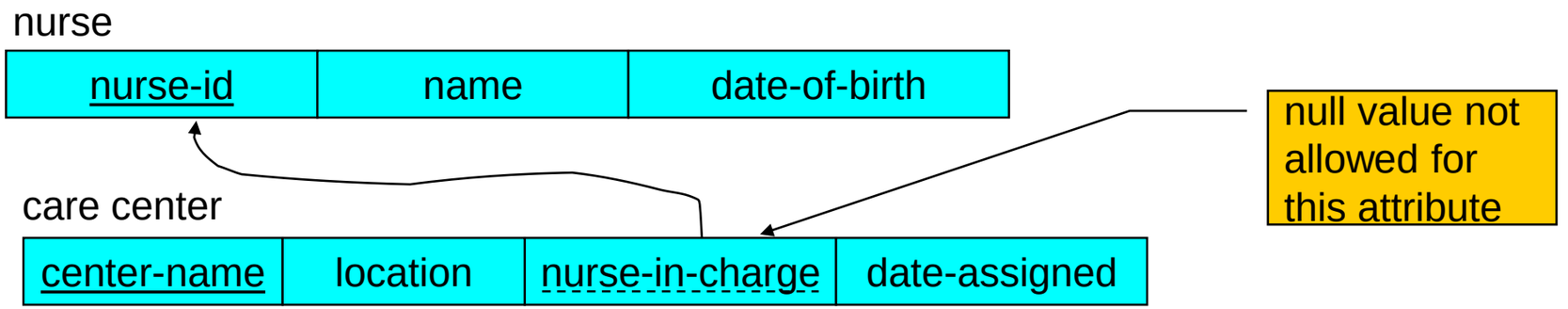
Mapping Binary Relationship

- Binary 1:1 relationships

E-R diagram



Resulting relation schemas



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:
- 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.
- 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:

1. An identifier was not assigned.
2. An identifier was assigned.

Mapping Associative Entities

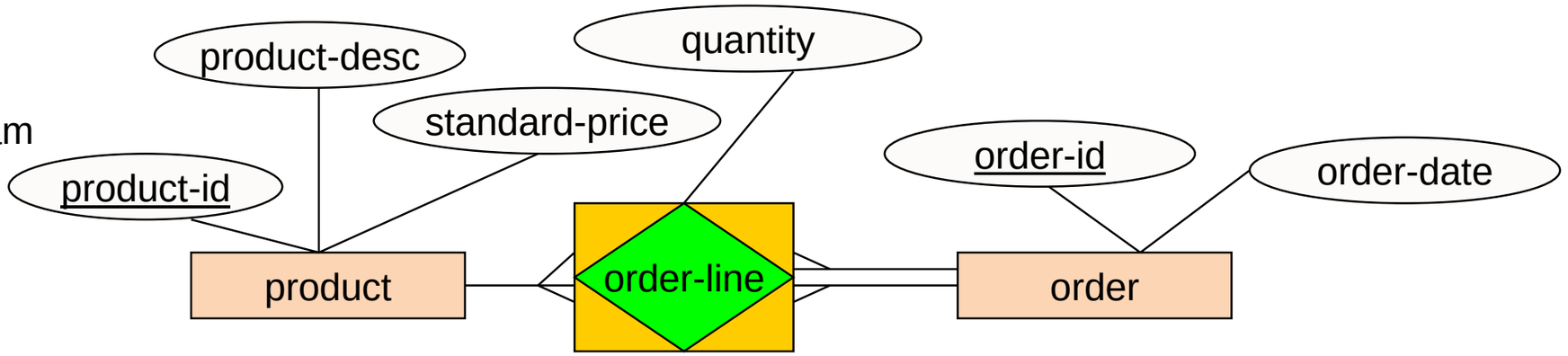
1. No identifier assigned:

- 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.

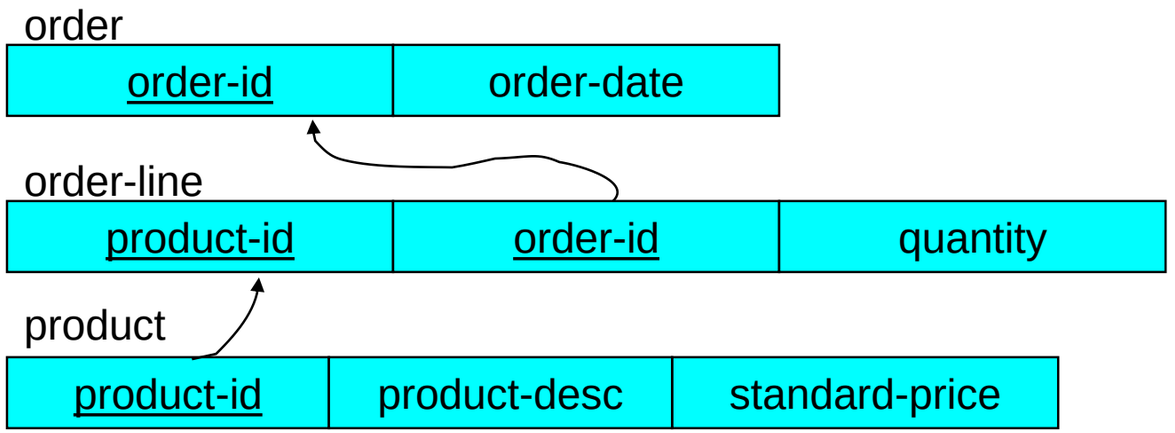
Mapping Associative Entities

1. No identifier assigned:

E-R diagram



Resulting relation schemas



Mapping Associative Entities

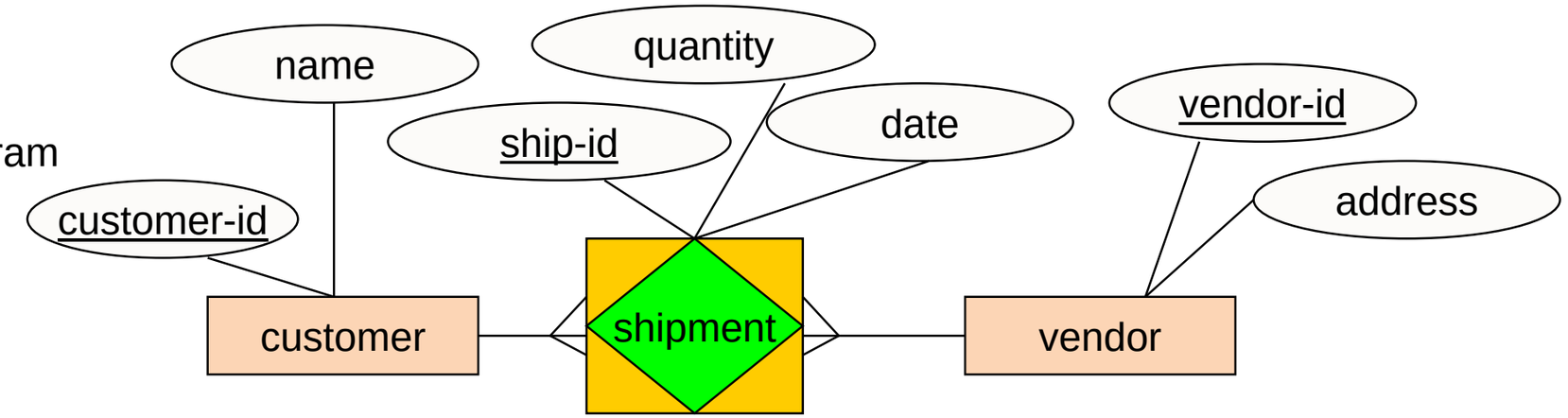
2. Identifier assigned:

- 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.
- 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.

Mapping Associative Entities

2. Identifier assigned:

E-R diagram



Resulting relation schemas

customer

customer-id	name	(other attributes)

shipment

shipment-id	customer-id	vendor-id	date	quantity

vendor

vendor-id	address	(other attributes)

Mapping Unary (recursive) Relationships

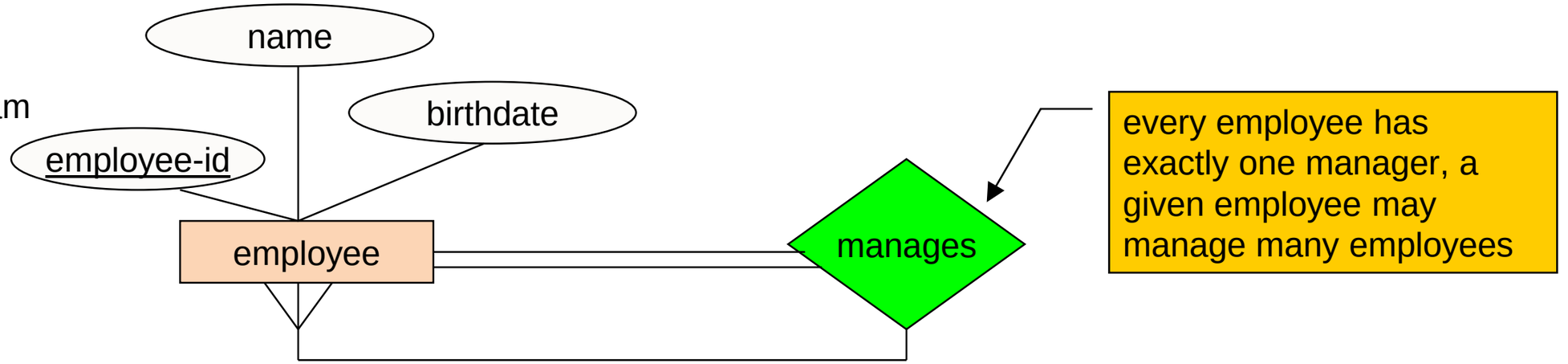
1:M cases

- 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.

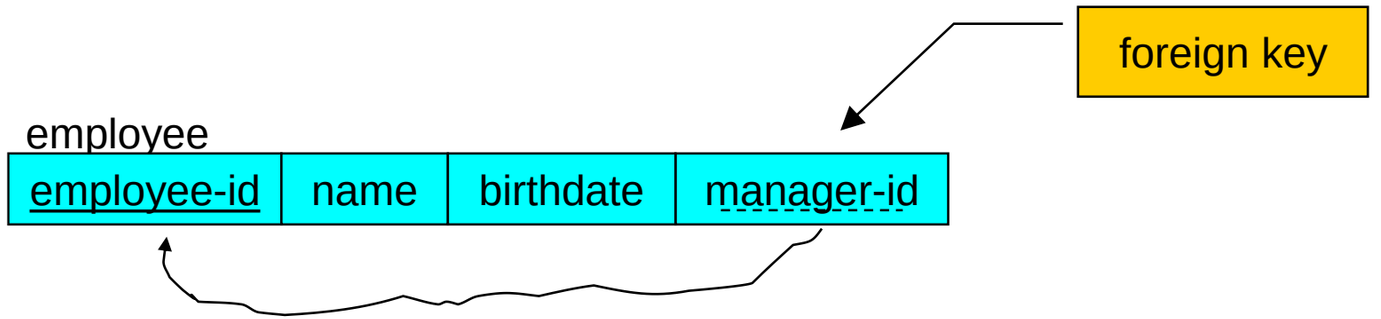
Mapping Unary (recursive) Relationships

1:M cases

E-R diagram



Resulting relation schema



Mapping Unary (recursive) Relationships

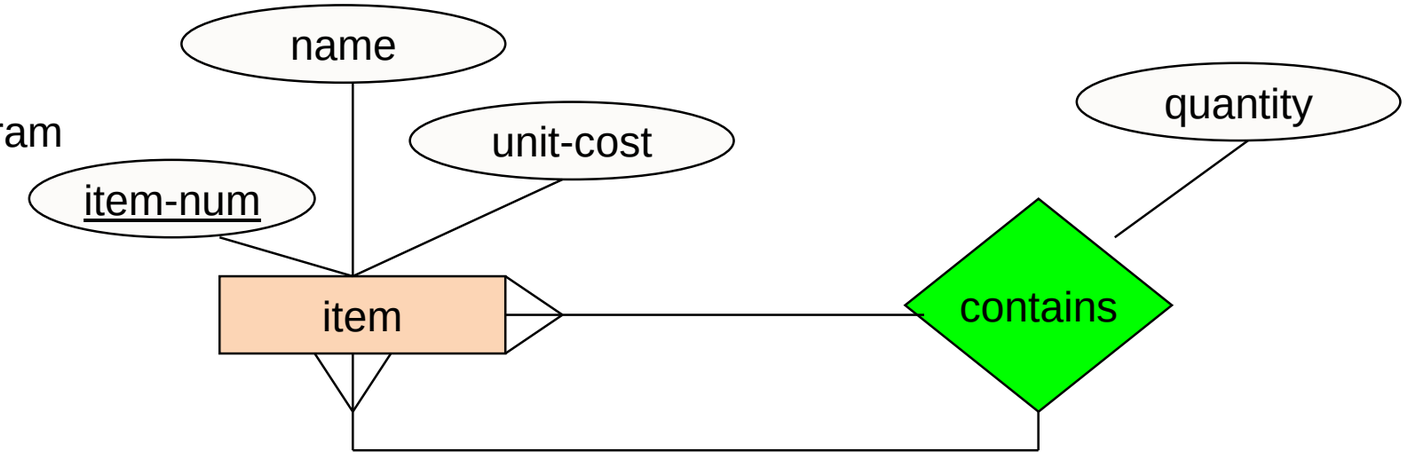
M:M cases

- 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.

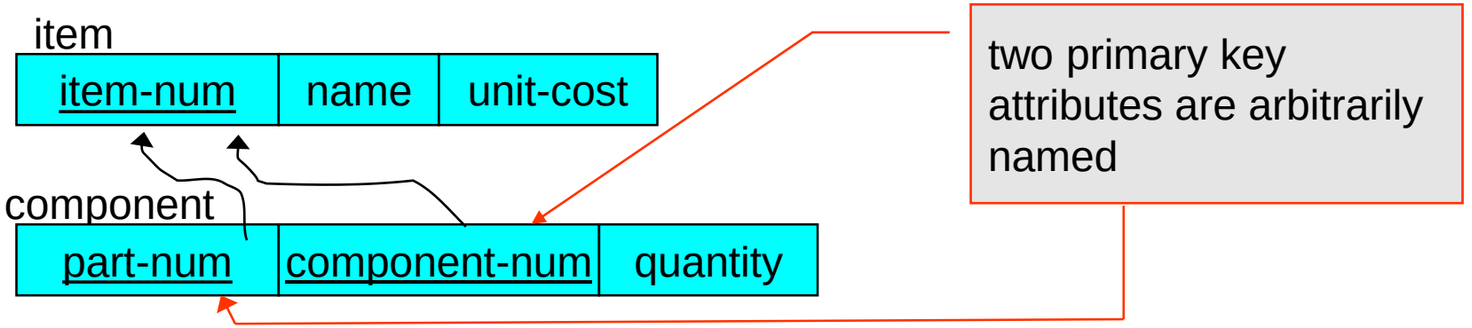
Mapping Unary (recursive) Relationships

M:M cases

E-R diagram

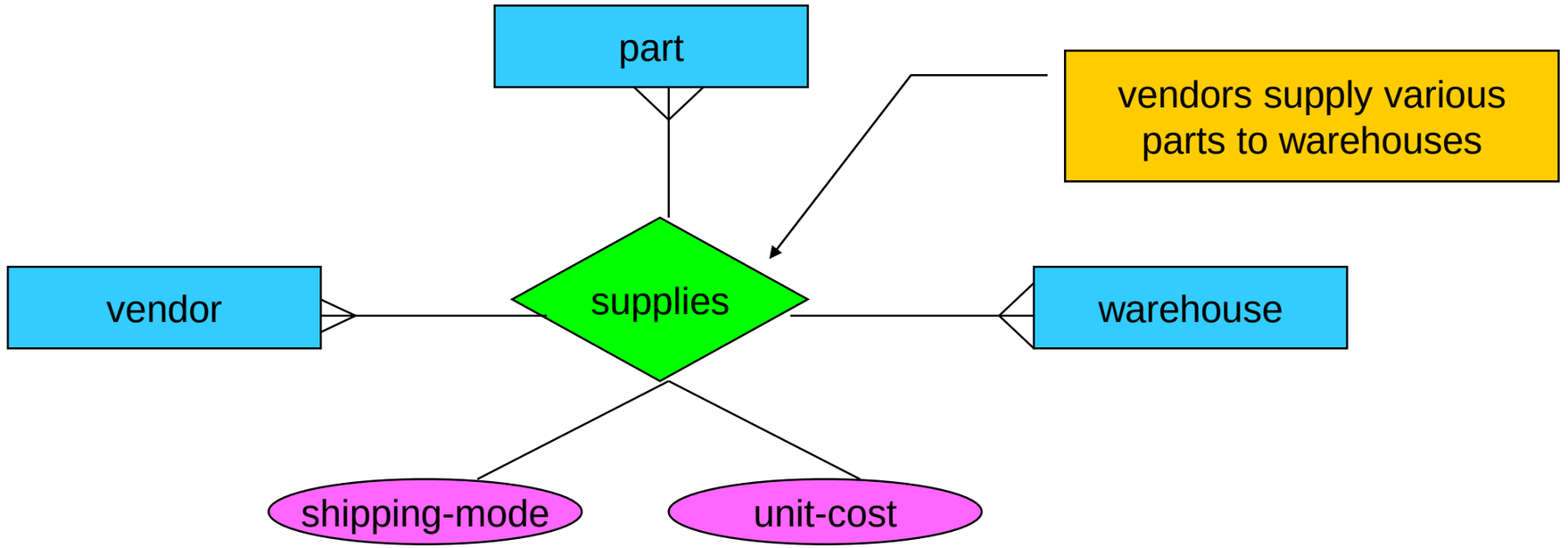


Resulting relation schema



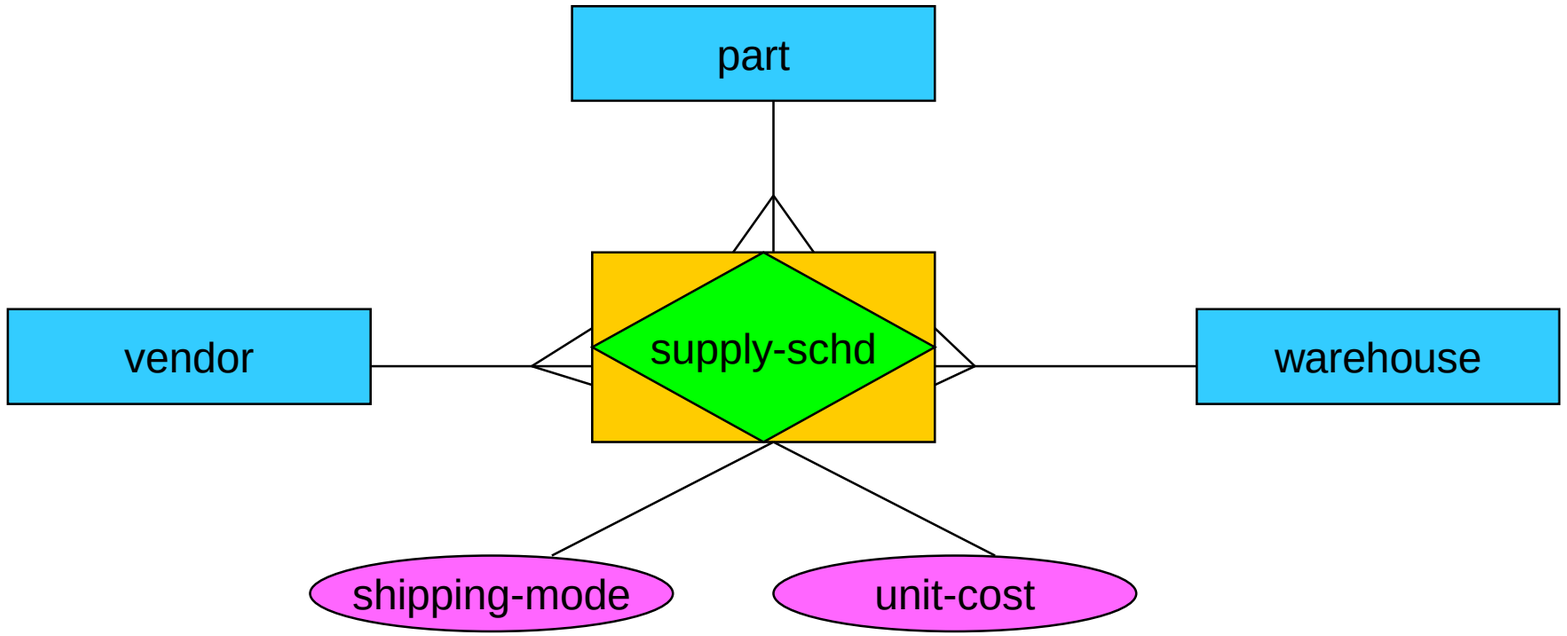
Mapping Ternary (n-ary) Relationships

- A ternary relationship is defined as a relationship among three entity types as shown below.



Mapping Ternary (n-ary) Relationships

- It is strongly recommended that all ternary (or higher) relationships be converted associative entities before proceeding further. An example is shown below which converts the ERD from the previous page into one with an associative entity:

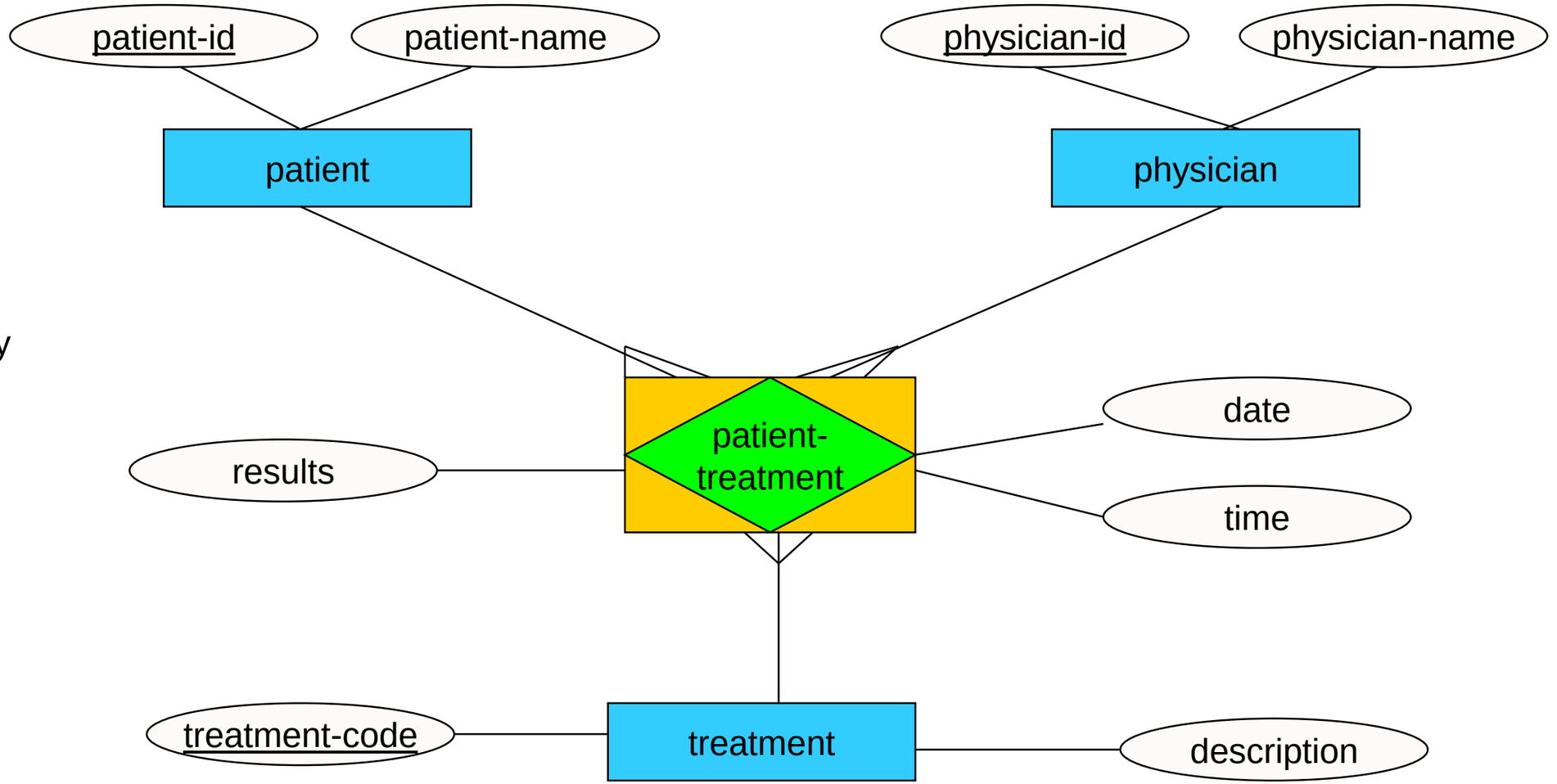


Mapping Ternary (n-ary) Relationships

- To map an associative entity set, 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.

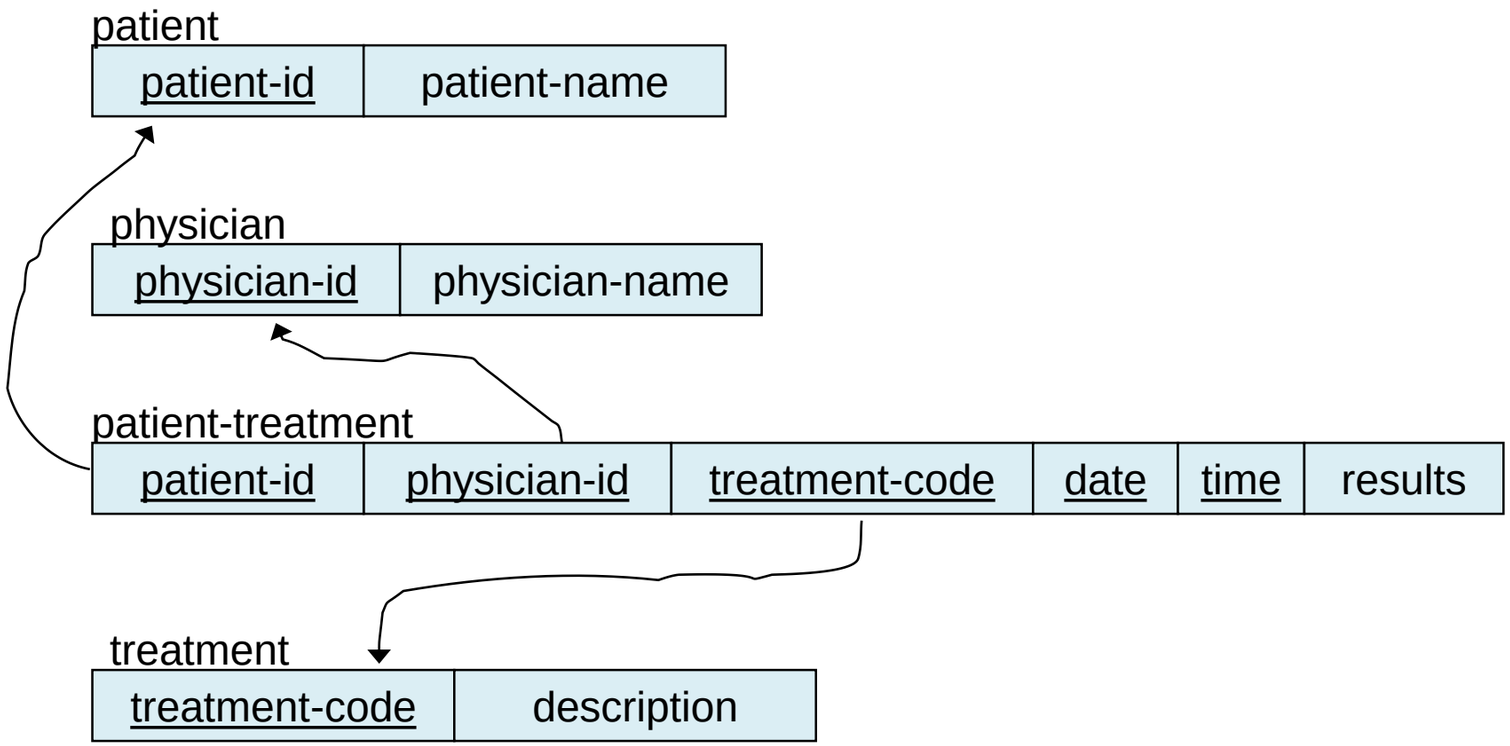
Mapping Ternary (n-ary) Relationships

ERD for ternary relationship with associative entity



Mapping Ternary (n-ary) Relationships

Resulting relational schemas



Mapping Supertype/Subtype Relationships

- Create a separate relation schema for the supertype and for each of its subtypes.
- Assign to the relation schema created for the supertype the attributes that are common to all members of the supertype, including the primary keys.
- Assign to the relation schema for each subtype the primary key of the supertype, and only those attributes that are unique to that subtype.
- Assign one (or more) attributes of the supertype to function as the subtype discriminator.

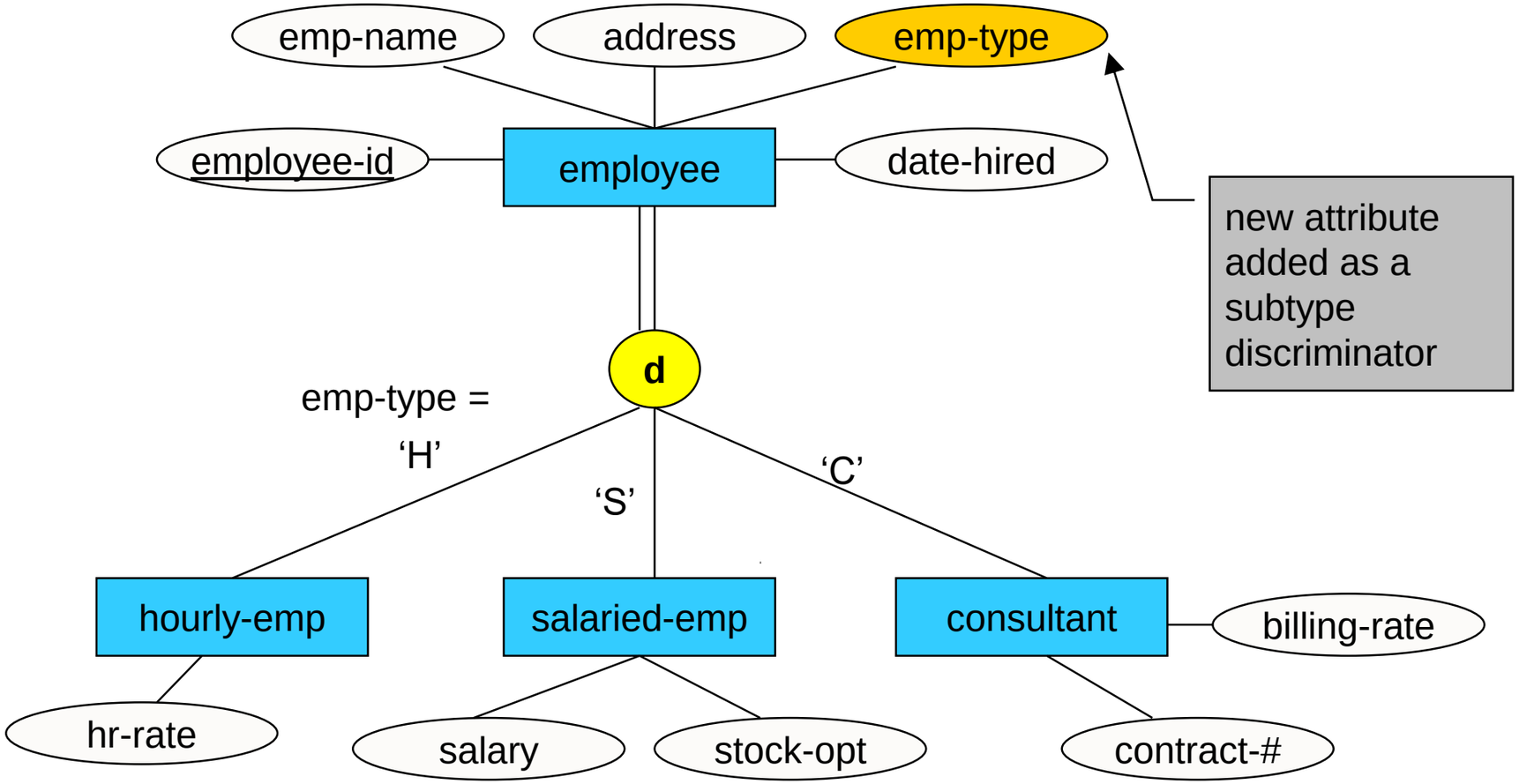
Mapping Supertype/Subtype Relationships

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.

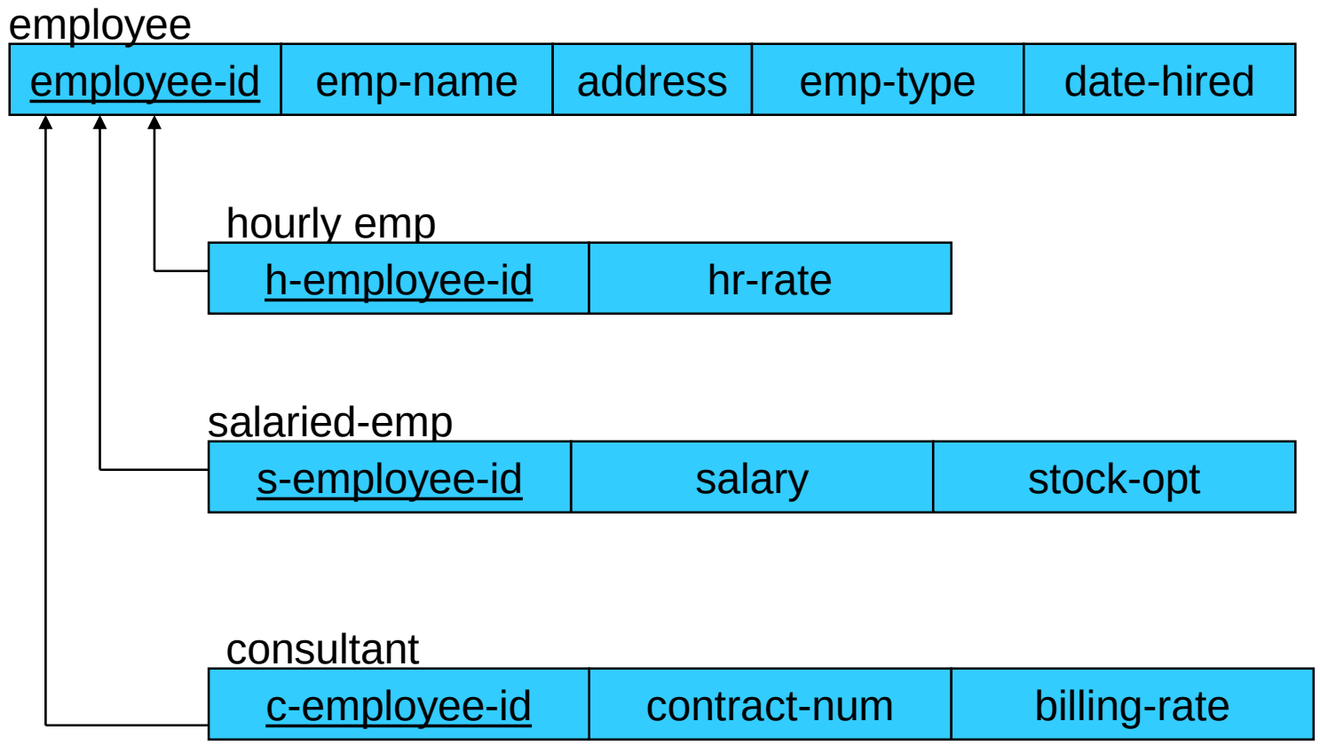
Mapping Supertype/Subtype Relationships

ERD showing disjoint subtype discriminator notation



Mapping Supertype/Subtype Relationships

Resulting relational schemas



Mapping ERD to Relational Schemas