Mapping Unary Relationships

- Map Unary Relationships - between the instances of a single entity type
- Also called recursive relationships
- The approach to mapping is different for the two types one-to-many and many-to-many

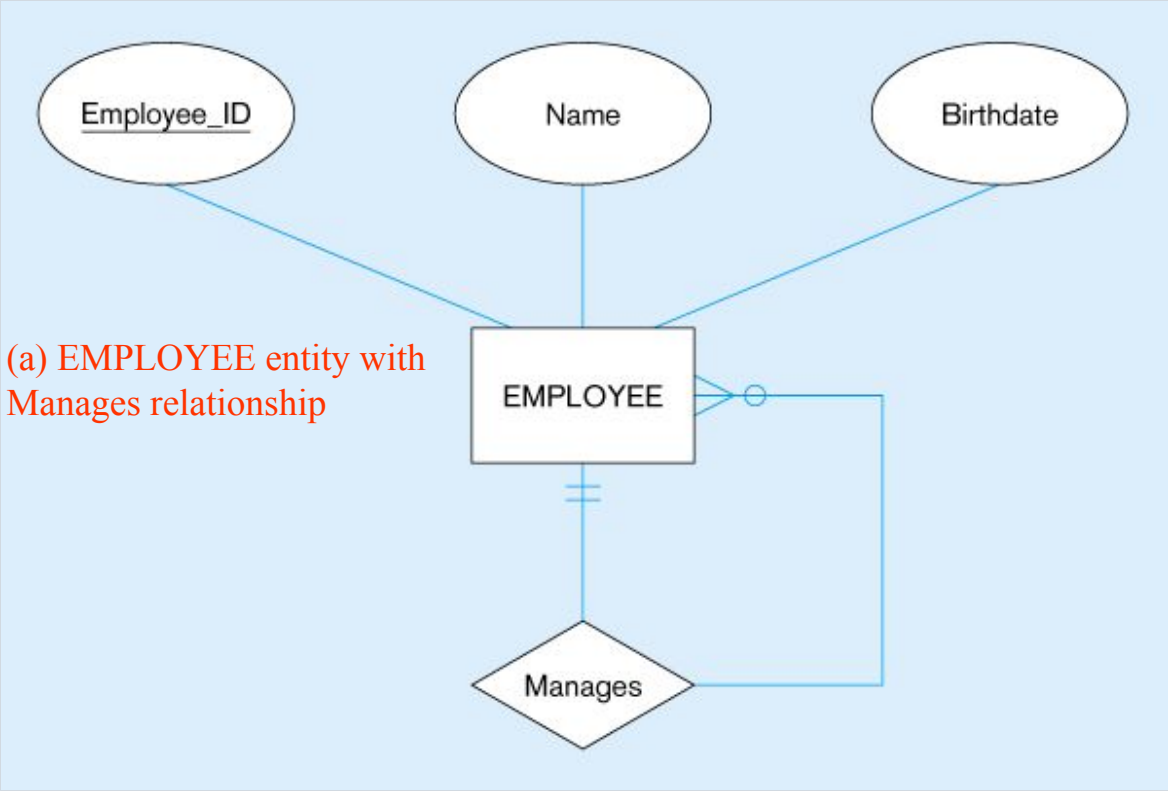
Unary one-to-many (1:N) relationships

- 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 the primary key values of that same relation
- The following Fig. shows a unary one-to-many relationship ‘Manages’ that associates each employee with another employee who is their manager. Each employee has exactly one manager, and a given employee may manage zero to many employees

Unary one-to-many (1:N) relationships

- The recursive foreign key in the relation is named Manager_ID
- This attribute has the same domain as the primary key Employee_ID
- Each row of this relation stores the following:
- Employee_ID, Birthdate and Manager_ID. Notice that as it is a foreign key, Manager_ID references Employee_ID

Mapping a unary 1:N relationship



(a) EMPLOYEE entity with Manages relationship

(b) EMPLOYEE relation with recursive foreign key

EMPLOYEE

Employee_ID	Name	Birthdate	Manager_ID

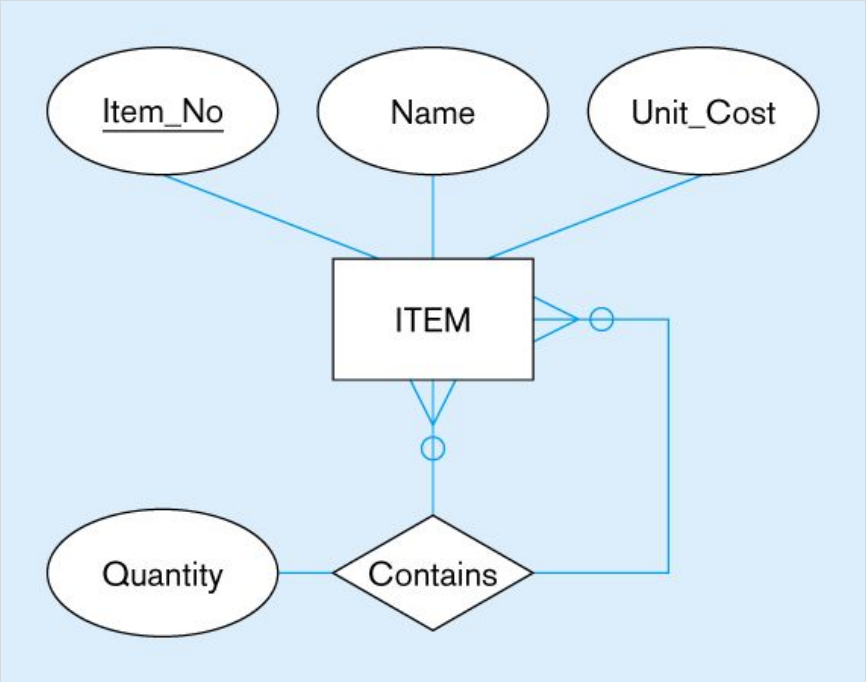
Unary many-to-many (M:N) relationships

- Here two relations are created, one to represent the entity type in the relationship and another representing the M:N relationship itself
- The primary key of the associative relation consists of two attributes, both taking their values from the primary key of the other relation
- Any non-key attribute of the relationship is included in the associative relation

Unary many-to-many (M:N) relationships

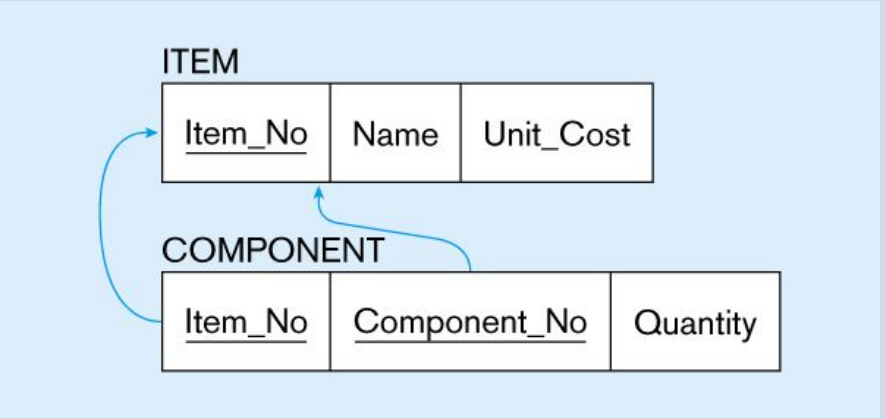
- Here a bill-of-materials relationship among items that are assembled from other items or components is shown.
- The relationship ‘Contains’ is M:N since a given item can contain numerous component items, and conversely an item can be used as a component in numerous other items

Mapping a unary M:N relationship



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

(b) ITEM and COMPONENT relations



Unary many-to-many (M:N) relationships

- The ITEM relation is mapped directly from the same identity type
- COMPONENT is an associative relation whose primary key consists of two attributes that are arbitrarily named Item_No and Component_No
- The attribute ‘Quantity’ is a nonkey attribute of this relation that for a given item records the quantity of a particular component used in the item
- Notice that both Item_No and Component_No reference the primary key (Item_No) of the ITEM relation

Unary many-to-many (M:N) relationships

We can easily query the above relation to determine the components of a given item

The following SQL query will list the immediate components (and their quantity) for an item number 100:

```
SELECT Component_No, Quantity
FROM COMPONENT
WHERE Item_No = 100
```

Map ternary (and n-ary) relationships

- It is best to convert a ternary relationship to an associative entity in order to represent participation constraints more accurately. Firstly, we create a new associative relation. The default primary key of this relation consists of the three primary key attributes for the participating entities (sometimes 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 of the new relation

Map ternary (and n-ary) relationships

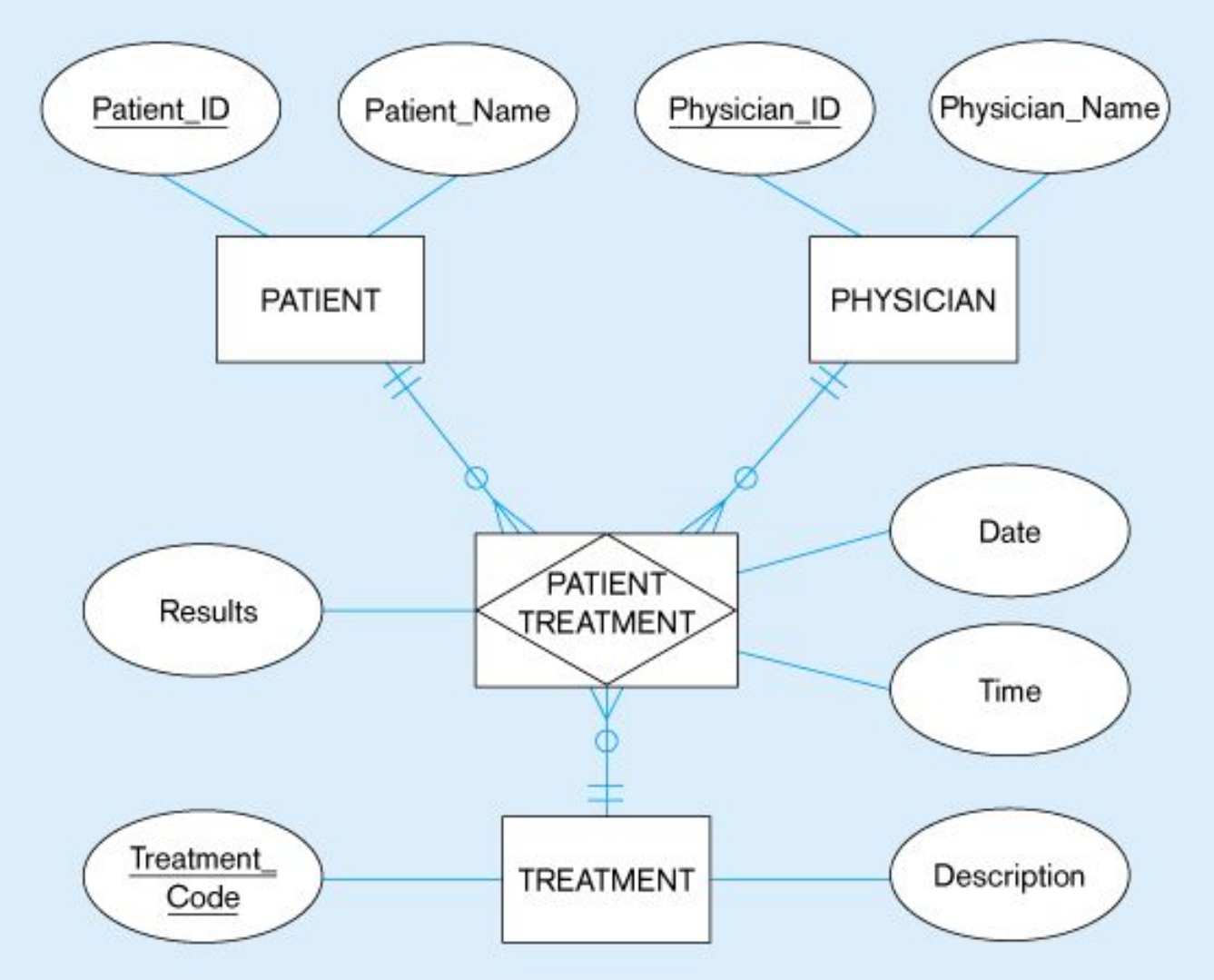
- The following Fig. Represents a PATIENT receiving a TREATMENT from a PHYSICIAN
- The associative entity type PATIENT_Treatment has the attributes Date, Time and Results and values of these are recorded for each instance of patient treatment
- The primary key attributes Patient_ID, Physician_ID and Treatment_Code become foreign keys in PATIENT_TREATMENT – these are components of its primary key but do not uniquely identify a given treatment, since a patient may receive the same treatment from the same physician on more than one occasion

Map ternary (and n-ary) relationships

- Does including the attribute ‘Date’ as part of the primary key (along with the other 3 attributes) result in a primary key?
- This would be so if a patient only receives one treatment from a given physician on a given date
- If this is not the case, we include Time as part of the primary key, which now consists of five attributes: Patient_ID, Physician_ID, Treatment_Code, Date and Time

Mapping a ternary relationship

Ternary relationship with associative entity



Mapping the ternary relationship