

The Entity-Relationship Model

NET3000
Database and SQL

Week 10

The Entity-Relationship model

- The E-R model is a detailed, logical representation of the data for an organisation or business area
- It should be understandable to both the user and to the IT technologist
- The model must be as 'open' as possible and not tied to any technology or to any particular business methodology
- It must be flexible enough so that it can be used and understood in practically any environment where information is modelled

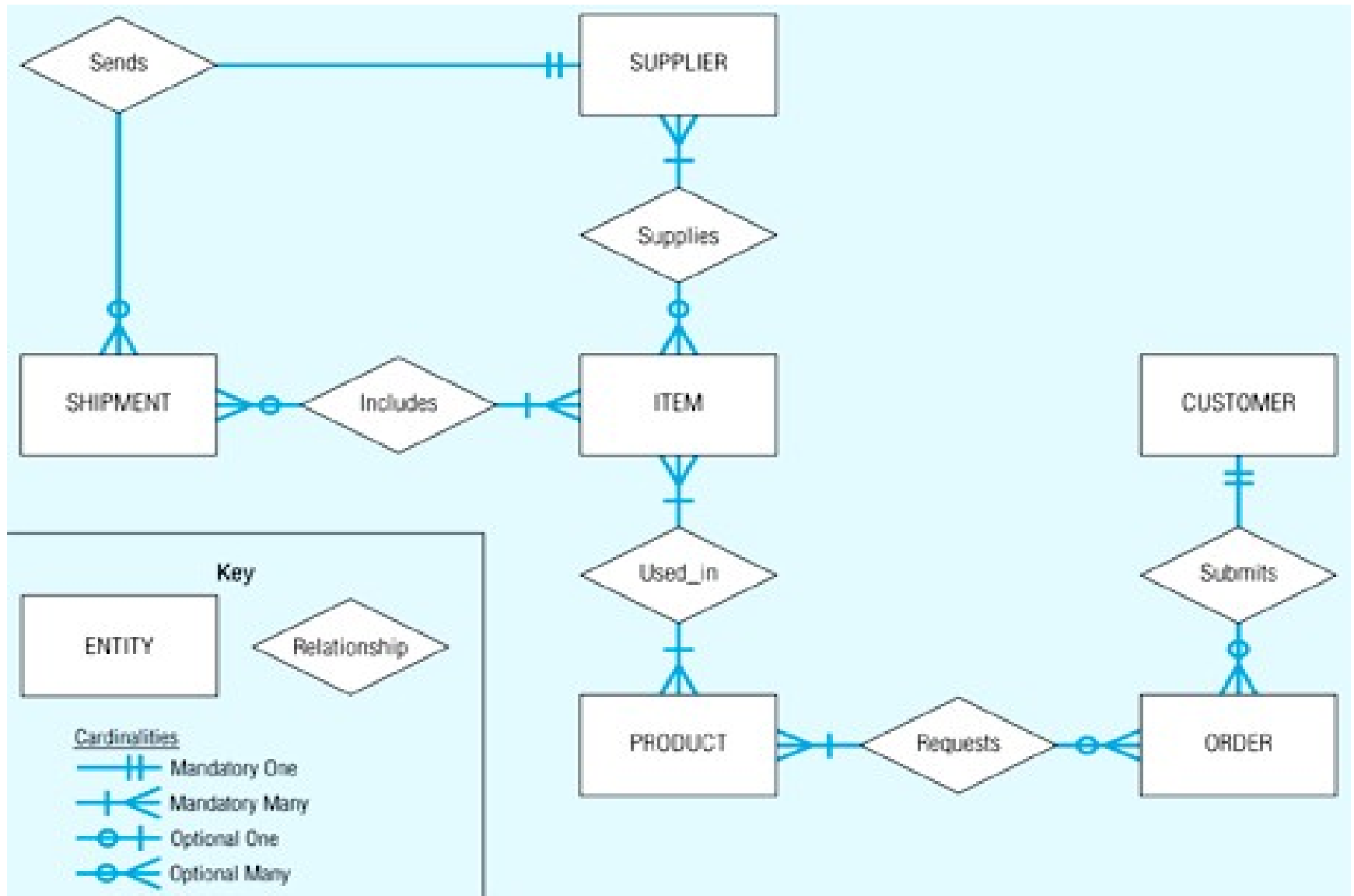
The ER model

- It is expressed in terms of entities in the business environment, the relationships (or associations) among those entities and the attributes (properties) of both the entities and their relationships
- The E-R model is usually expressed as an E-R diagram

E-R Model Constructs

- Entity - person, place, object, event, concept
- Entity Type - is a collection of entities that share common properties or characteristics. Each entity type is given a name, since this name represents a set of items, it is always singular. It is placed inside the box representing the entity type (see Fig. in next slide)
- Entity instance – is a single occurrence of an entity type. An entity type is described just once (using metadata) in a database, while many instances of that entity type may be represented by data stored in the database. e.g. – there is one EMPLOYEE entity type in most organisations, but there may be hundreds of instances of this entity stored in the database

Sample E-R Diagram



Strong versus Weak entity type

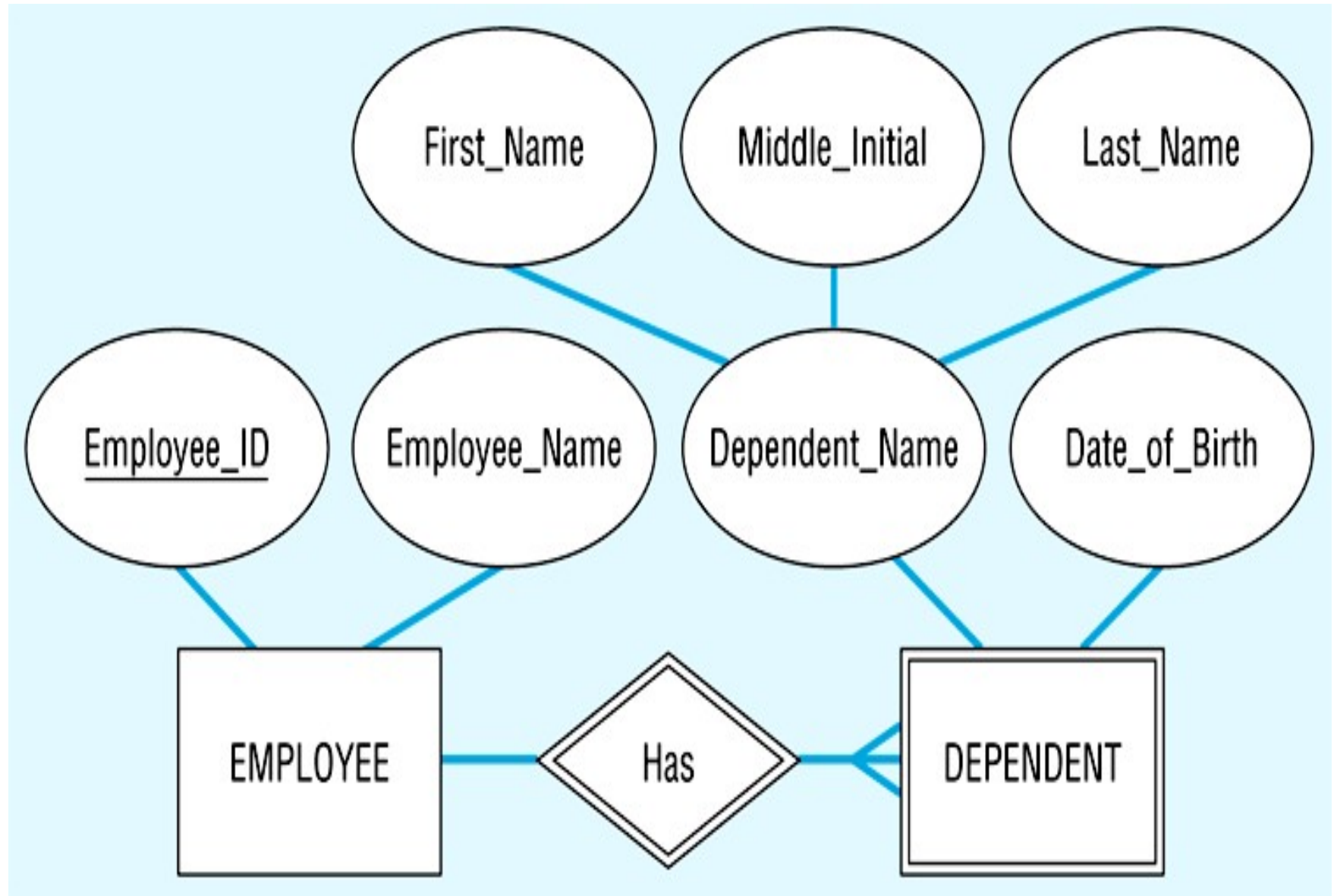
- Most of the basic entity types are classified as strong entity types [Rectangle] – one that exists independently from other entity types (such as EMPLOYEE)
- Always have a unique characteristic (identifier) – an attribute or combination of attributes that uniquely distinguish each occurrence of that identity
- A weak entity type [[Double Rectangle]] – existence depends on some other entity type. It has no meaning in the ER diagram without the entity on which it depends (such as DEPENDENT)
- The entity type on which the weak entity type depends is called the Identifying owner (or owner for short).

Strong versus Weak entity type

Identifying relationship is the relationship between a weak entity type and its owner (such as 'Has' in the following Fig.)

Weak entity identifier is its partial identifier (double underline) combined with that of its owner. During a later design stage dependent name will be combined with Employee_ID (the identifier of the owner) to form a full identifier for DEPENDENT.

Example of a weak entity



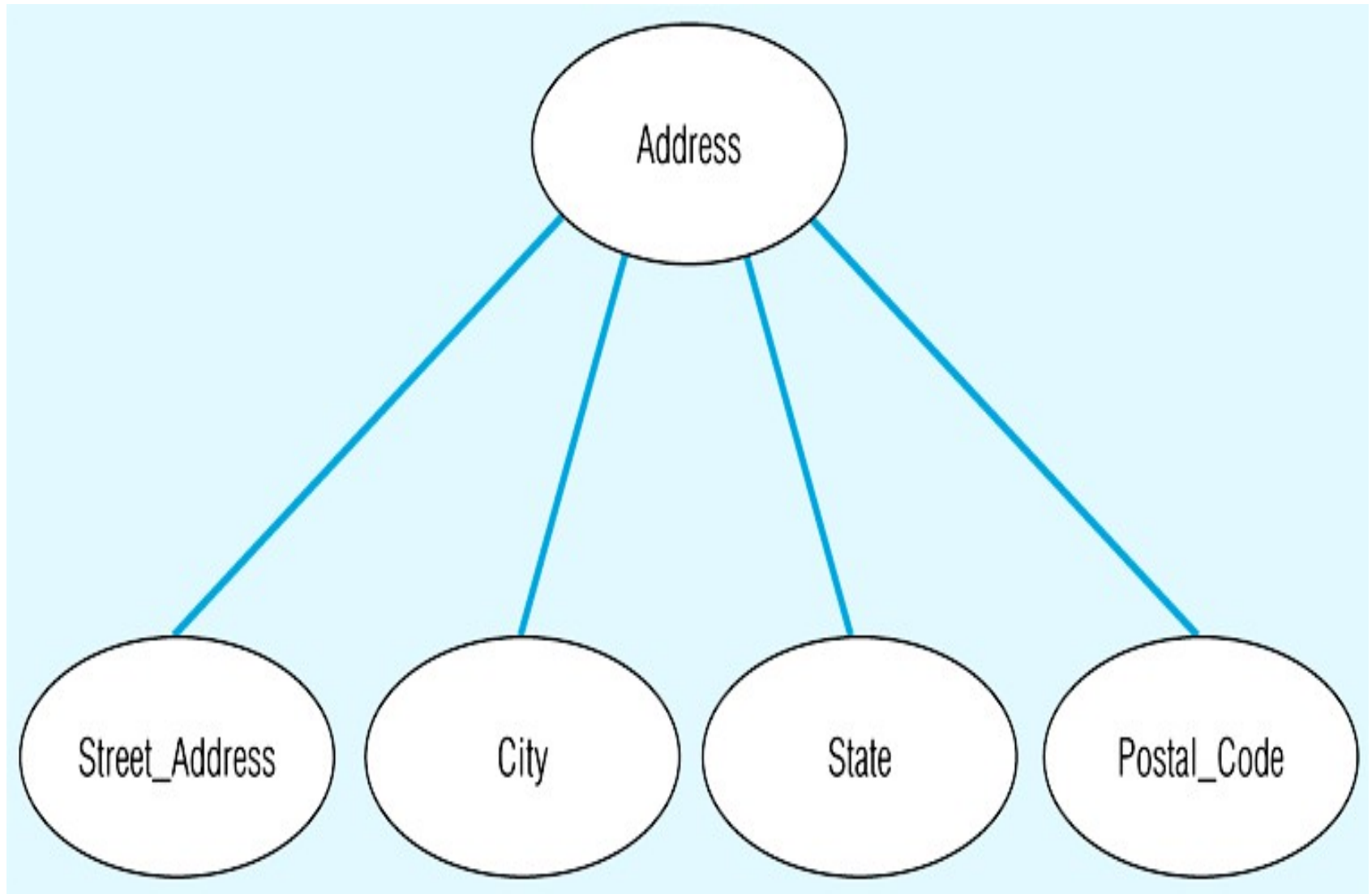
Attributes

- An attribute is a property or characteristic of an entity type, for example the entity EMPLOYEE may have attributes Employee_Name and Employee_Address.
- In ER diagrams place attributes name in an ellipse with a line connecting it to its associated entity
- Attributes may also be associated with relationships
- An attribute is associated with exactly one entity or relationship

Simple versus composite attributes (following Fig.)

- Some attributes can be broken down into meaningful component parts, such as Address, which can be broken down into Street_Address, City..etc.
- The component attributes may appear above or below the composite attribute on an ER diagram
- Provide flexibility to users, as can refer to it as a single unit or to the individual components
- A simple (atomic) attribute is one that cannot be broken down into smaller components

A composite attribute



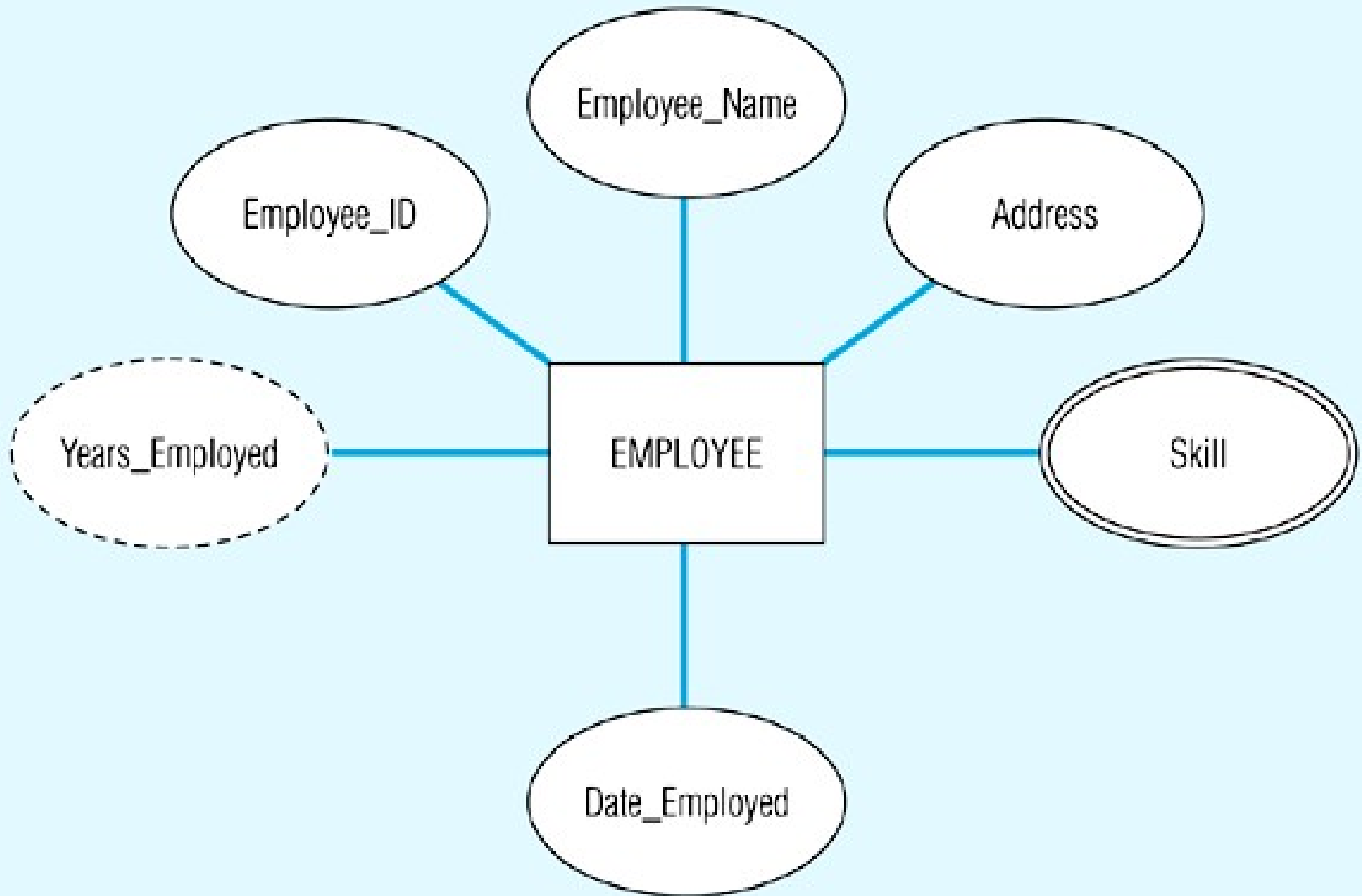
Single-Valued versus Multivalued Attribute

- It frequently happens that there is an attribute that may have more than one value for a given instance, e.g. EMPLOYEE may have more than one Skill.
- A multivalued attribute is one that may take on more than one value – it is represented by an ellipse with double lines

Stored versus Derived Attributes

- Some attribute values can be calculated or derived from others
- e.g., if Years_Employed needs to be calculated for EMPLOYEE, it can be calculated using Date_Employed and Today's_Date
- A derived attribute is one whose value can be calculated from related attribute values (plus possibly other data not in the database)
- A derived attribute is signified by an ellipse with a dashed line (see next Fig.)

Entity with a multivalued attribute (Skill)
and derived attribute (Years_Employed)

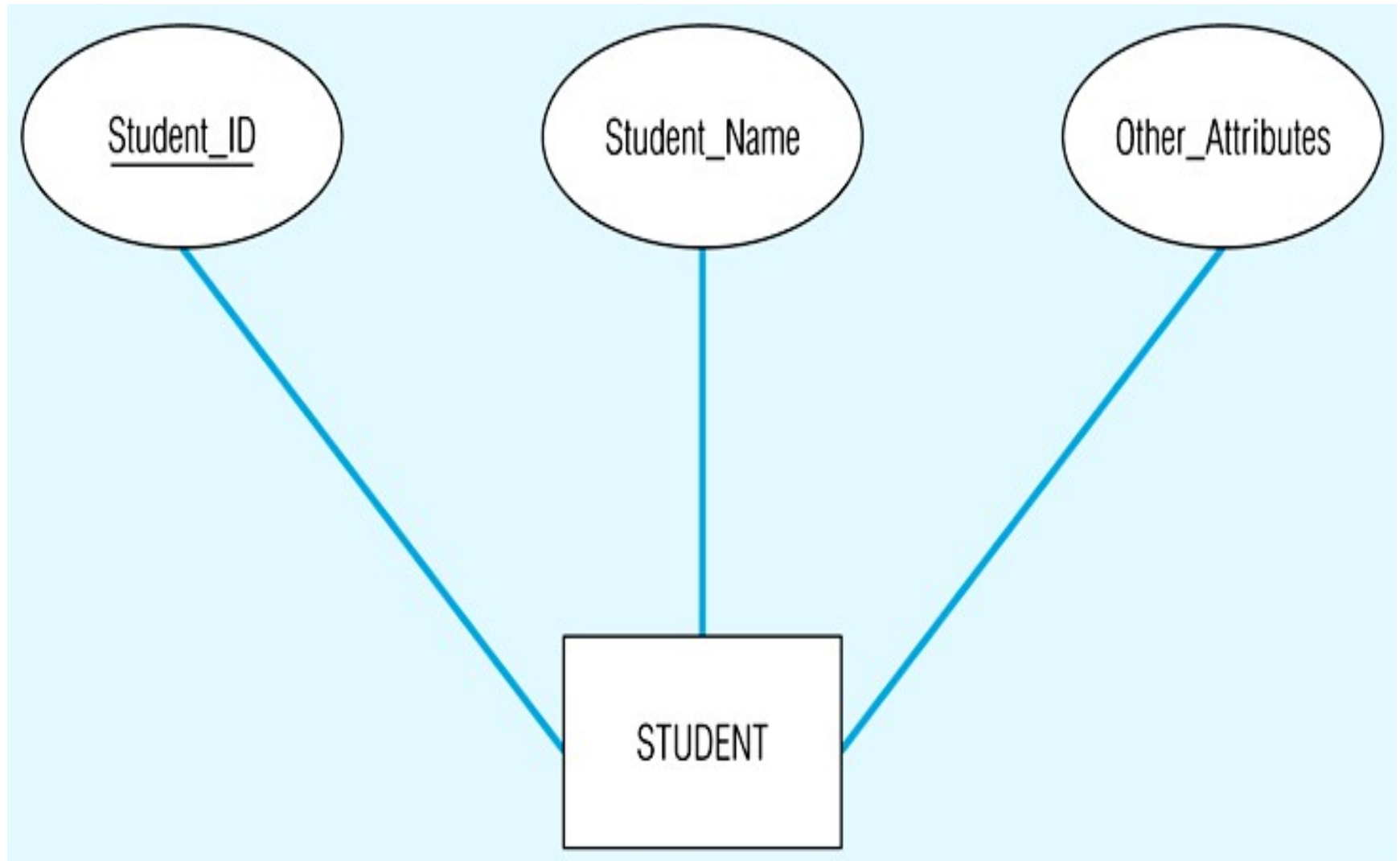


Identifier attribute

- Identifier attribute or Key is an attribute (or combination of attributes) that uniquely identifies individual instances of an entity type, such as Student_ID
- To be a candidate identifier, each entity instance must have a single value for the attribute, and the attribute must be associated with each entity
- The identifier attribute is underlined, such as Student_ID

Simple and composite key attributes

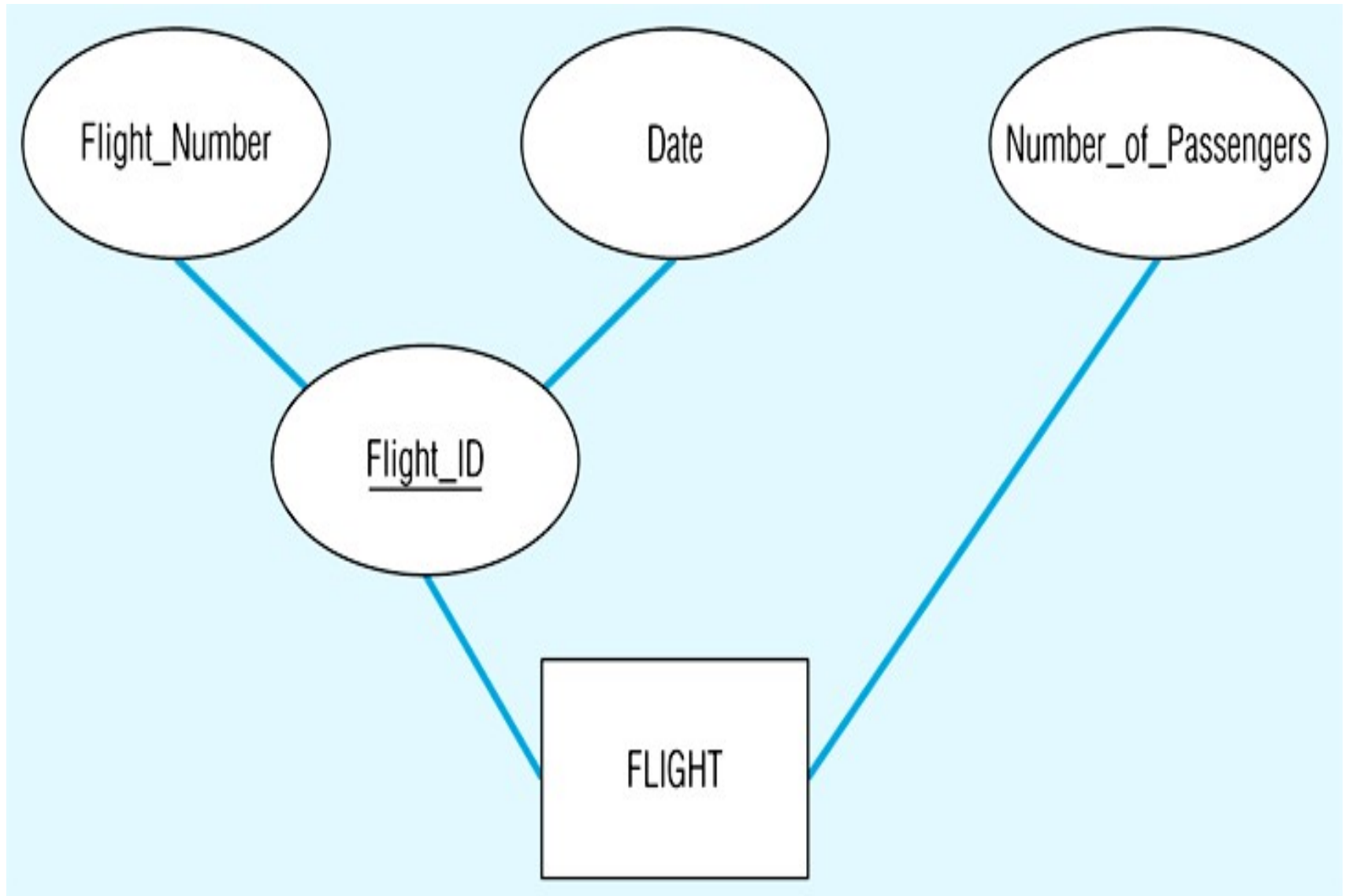
(a) Simple key attribute



Composite Identifier

- A Composite Identifier is when there is no single (or atomic) that can serve as an identifier
- Flight_ID is a composite identifier that has component attributes Flight_Number and Date – this combination is required to uniquely identify individual occurrences of Flight
- Flight_ID is underlined, whilst its components are not

(b) Composite key attribute



Criteria for selecting identifiers

Some entities have more than one candidate identifier, so the following criteria should be used:

Choose identifier that will not change in value over the life of each instance of the entity type

Choose identifier that is guaranteed to have valid values and
Will not be null (or unknown). If composite, make sure all parts will have valid values

Criteria for selecting identifiers

Avoid the use of intelligent identifiers whose structure indicates classifications, locations or people that might change. e.g. the first two digits of an identifier may indicate a warehouse location, but such codes are often changed as conditions change, which renders them invalid.

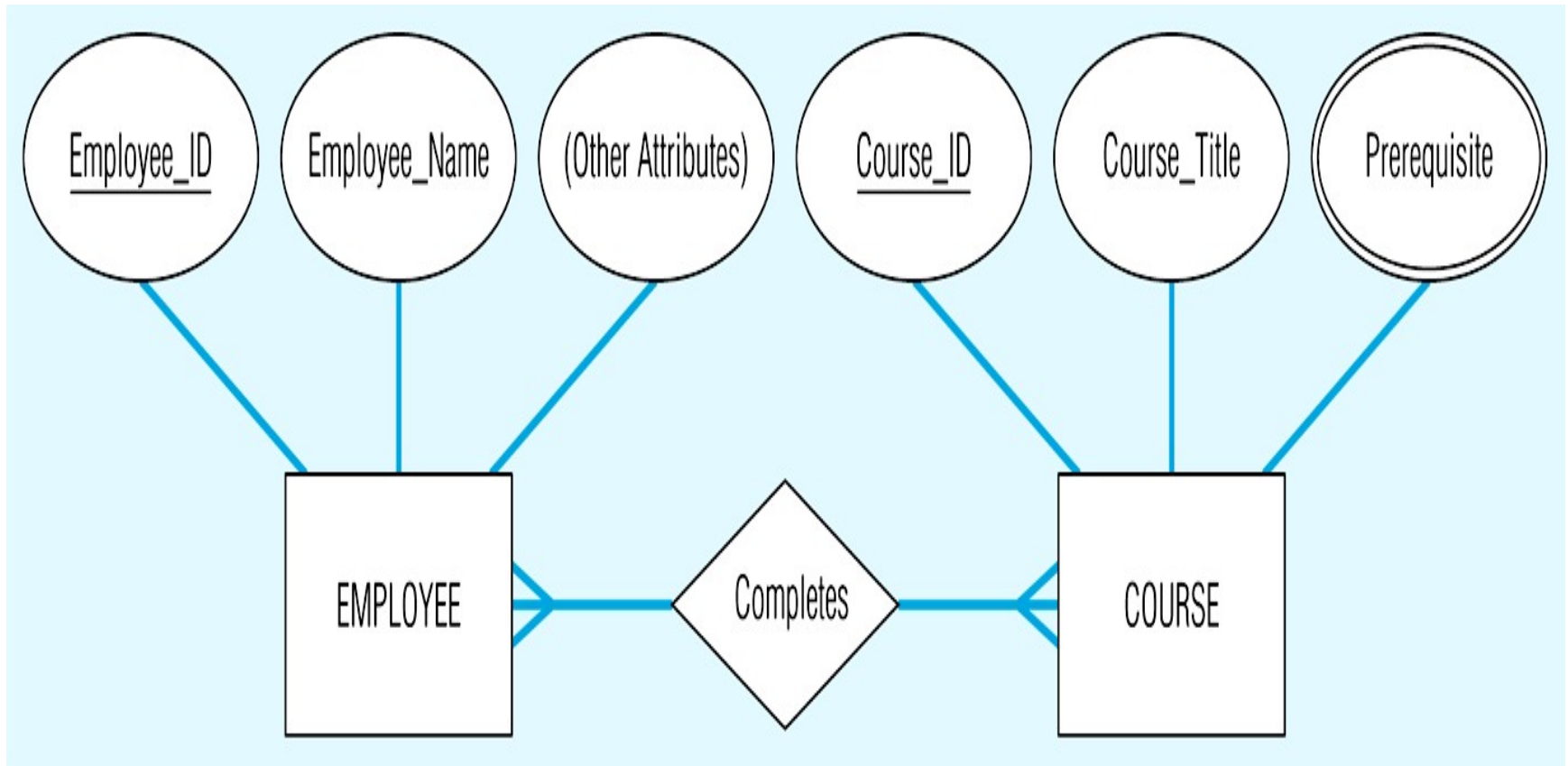
Consider substituting new, simple identifiers for long, composite ones, e.g. an attribute called Game_Number could be used for the entity type GAME instead of Home_Team and Away_Team

Relationships (following Fig.)

- A relationship is an association among the instances of one or more entity types that is of interest to the organisation
- Relationship Type is a meaningful association between (or among) entities – implying that the relationship allows us to answer questions that could not be answered given only the entity types. It is denoted by a diamond symbol

Relationship types and instances

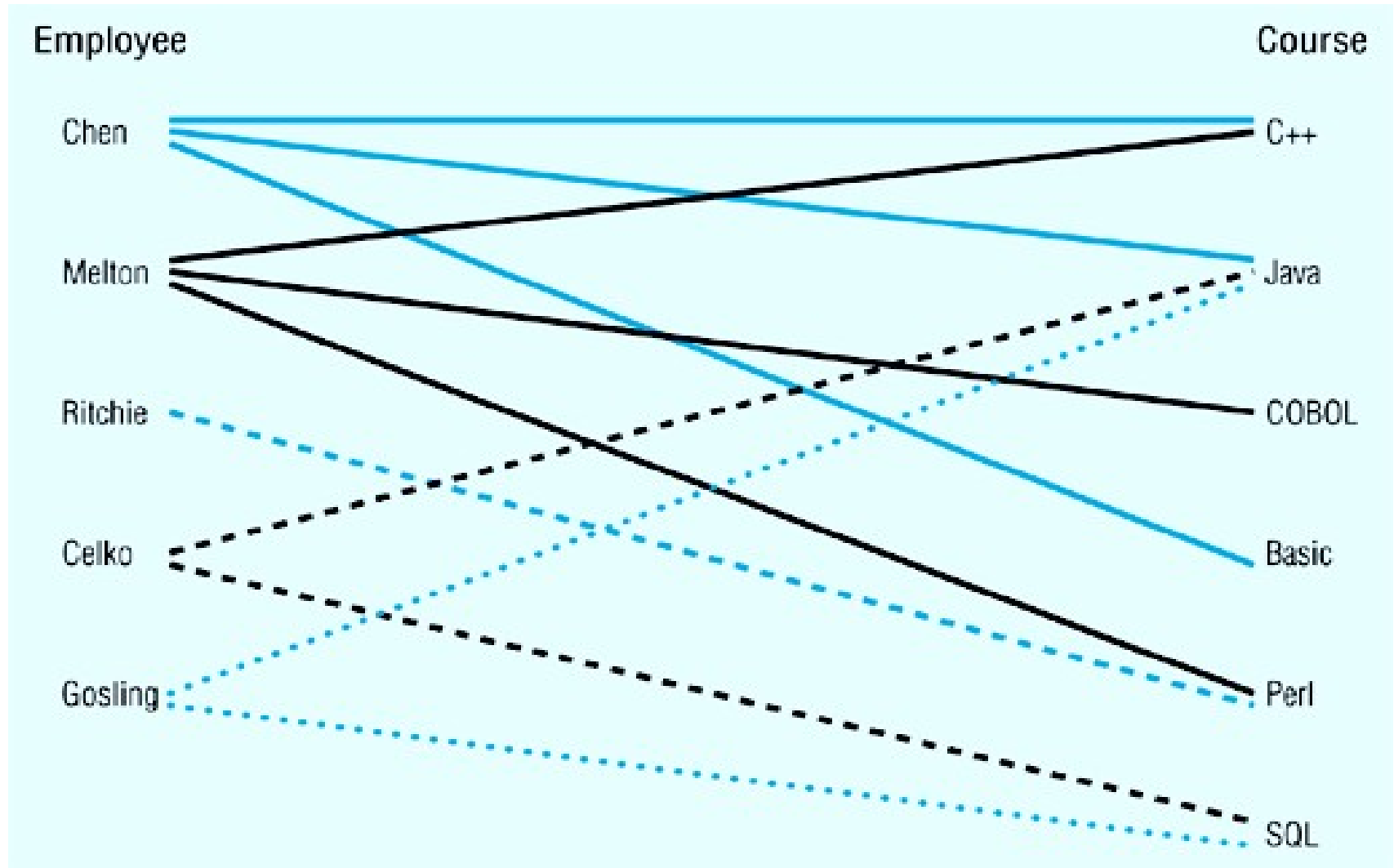
(a) Relationship type (Completes)



Relationship instance

- Is an association between (or among) entity instances, where each relationship includes exactly one entity from each participating entity type.
- For example, in the following figure each line represents a relationship instance between one employee and one course, indicating that the employee has completed that course

(b) Relationship instances

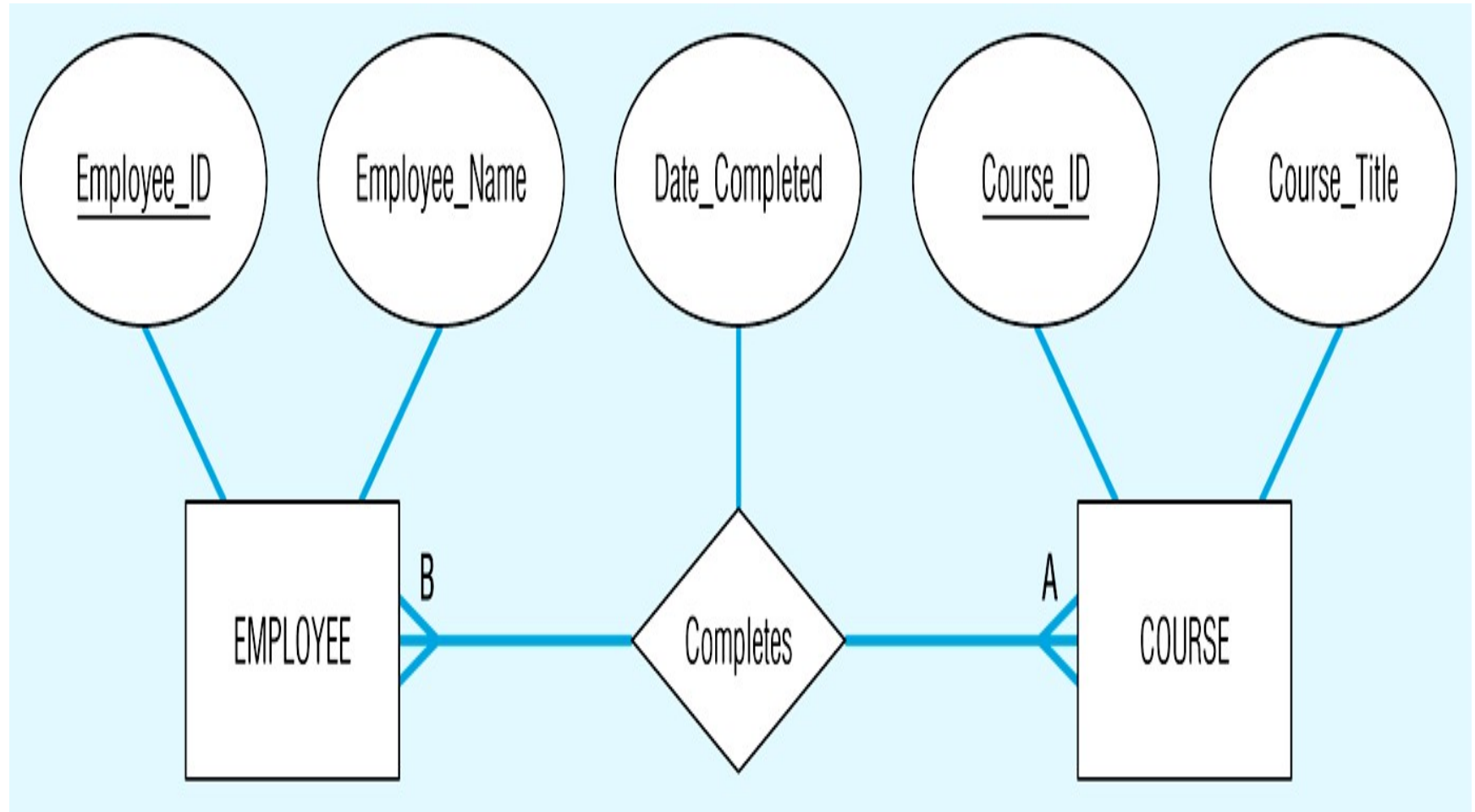


Attributes on relationships

- Attributes may be associated with a many-to-many (or one-to-one) relationship, as well as with an entity
- e.g., an organisation may want to record the date when an employee completes each course
- In the following diagram, the relationship ‘Completes’ joins the EMPLOYEE and COURSE entities, and Date_Completed is joined to this as it is a property of the relationship ‘Completes’

An associative entity

(a) Attribute on a relationship



Associative entities (gerunds)

- The presence of one or more attributes on a relationship suggests that the relationship should perhaps be represented as an entity type
- An associative entity is an entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.
- The associative entity type CERTIFICATE is represented with the diamond relationship symbol enclosed within the entity rectangle

Associative entities (gerunds)

- The purpose of this special symbol is to preserve the information that the entity was initially specified as a relationship on the ER diagram
- There is no relationship diamond on the line between an associative entity and a strong entity, because the associative entity represents the relationship

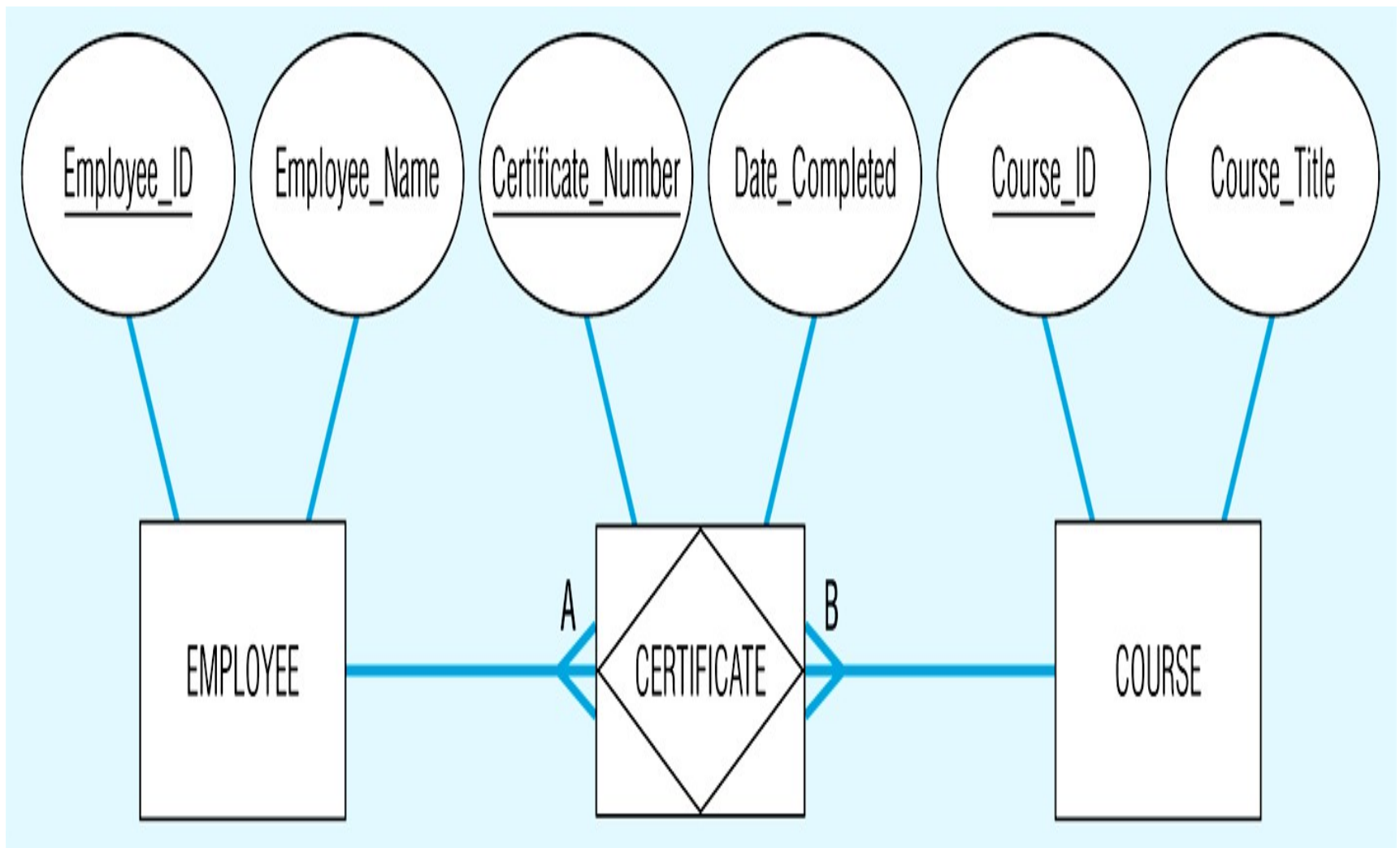
Associative entities

- How do you know when to convert a relationship to an associative entity type? Four conditions should exist:
 1. All of the relationships are ‘many’ relationships
 2. The resulting associative identity type has independent meaning to end-users, and can preferably be identified with a single-attribute identifier
 3. The associative entity has one or more attributes in addition to the identifier
 4. The associative entity participates in one or more relationships independent of the entities related in the associated relationship

Associative entities

- The following figure shows the relationship ‘Completes’ converted to an associative entity type
- A CERTIFICATE is awarded to each EMPLOYEE who completes a COURSE, each certificate has a Certificate_Number that serves as the identifier

(b) An associative entity (CERTIFICATE)



Degree of a relationship

Is the number of entity types that participate in it.

Thus 'Completes' has degree 2, since there are two participating entity types, EMPLOYEE and COURSE

The three most common relationship degrees are unary (degree 1), binary (degree 2) and ternary (degree 3 –see following Fig.)

Higher degree relationships are possible but rarely encountered in practice

Unary relationship

- Is between the instances of a single entity type (also called recursive relationships)
- ‘Is_Married_To’ is a one-to-one relationship between instances of the PERSON entity type
- ‘Manages’ is a one-to-many relationship between instances of the EMPLOYEE entity type

Binary relationships

- Between the instances of two entity types, and is the most common type of relationship encountered in data modelling.
e.g. (one-to-one) an EMPLOYEE is assigned one PARKING_PLACE, and each PARKING_PLACE is assigned to one EMPLOYEE
- e.g. (one to many) a PRODUCT_LINE may contain many PRODUCTS, and each PRODUCT belongs to only one PRODUCT_LINE
- e.g. (many-to-many) a STUDENT may register for more than one COURSE, and each COURSE may have many STUDENTS

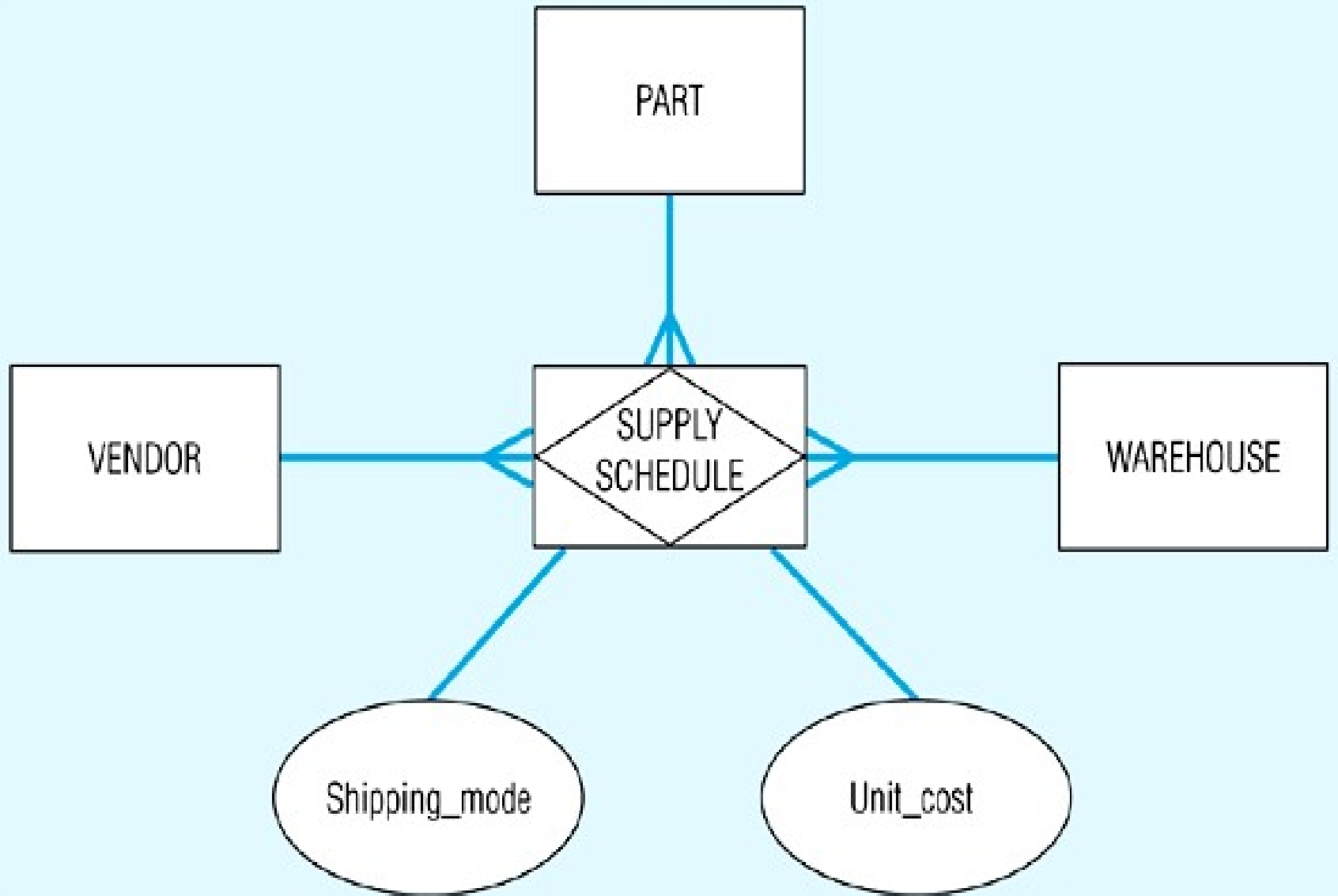
Ternary relationships

- A ternary relationship is a simultaneous relationship among the instances of 3 entity types
- It is the most common relationship encountered in data modelling
- The following Fig. shows a typical ternary relationship
- Here, vendors can supply various parts to warehouses

Ternary relationships

- The relationship ‘Supplies’ is used to record the specific PARTs supplied by a given VENDOR to a particular WAREHOUSE
- There are two attributes on the relationship ‘Supplies’, Shipping_Mode and Unit_Cost
- e.g. one instance of ‘Supplies might record that VENDOR X can ship PART C to WAREHOUSE Y, that the Shipping_Mode is ‘next_day_air’ and the Unit_Cost is £5-00 per unit

Ternary relationships

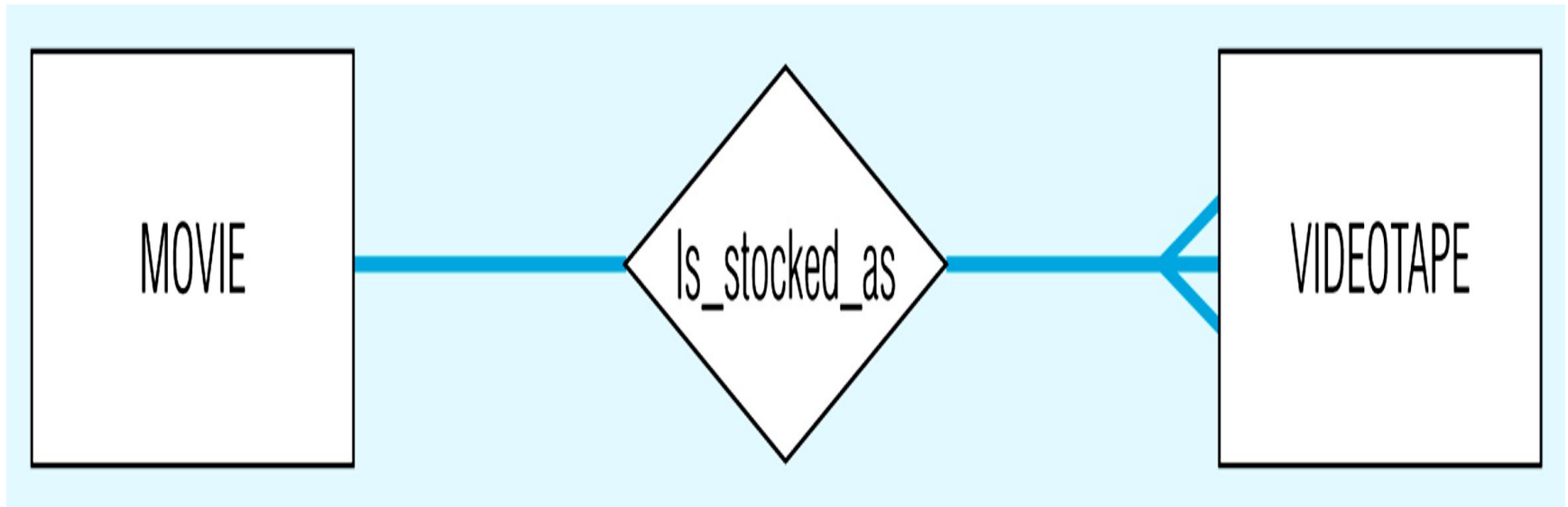


Cardinality constraints

- The number of instances of one entity that can or must be associated with each instance of another entity.
- If we have two entity types A and B, the cardinality constraint specifies the number of instances of entity B that can (or must) be associated with entity A
- e.g. a video store may stock more than one VIDEOTAPE for each MOVIE, this is a 'one-to-many' relationship as in the following Fig.

Introducing cardinality constraints

(a) Basic relationship



Minimum cardinality

- Yet there may be a more precise way of saying this
- The minimum cardinality of a relationship is the minimum number of instances of an entity B that may be associated with each instance of an entity A
- In our example, the minimum number of VIDEOTAPES of a MOVIE is zero (entity B is an optional participant in the 'Is_Stocked_As' relationship)
- This is signified by the symbol zero through the arrow near the VIDEOTAPE entity in the following Fig.

Maximum cardinality

- Is the maximum number of instances of an entity B that may be associated with each instance of entity A
- In the following Fig., the maximum cardinality for the VIDEOTAPE entity type is ‘many’ (an unspecified number greater than 1)
- This is indicated by the ‘crow’s foot’ symbol on the arrow next to the VIDEOTAPE entity symbol

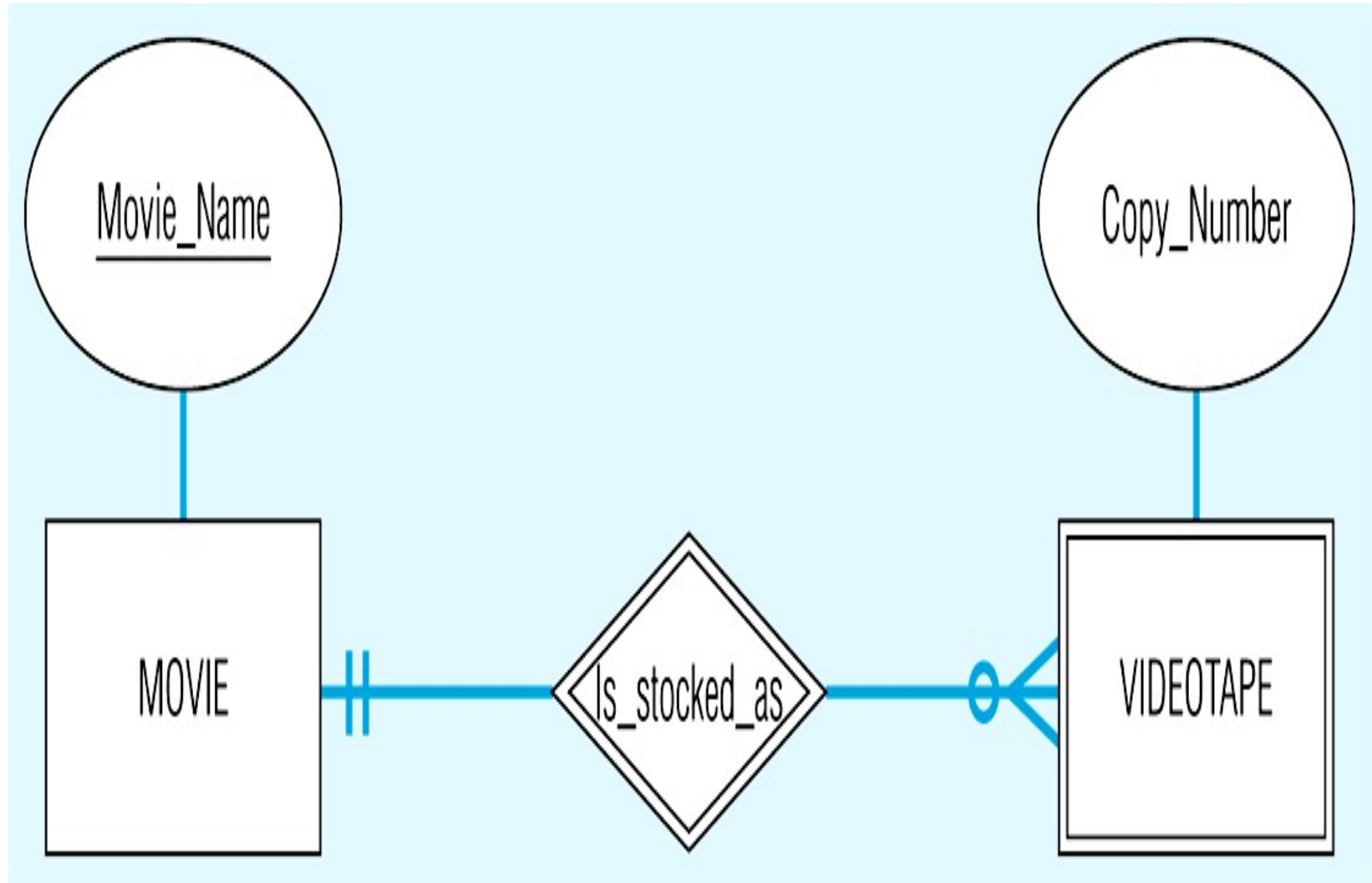
Mandatory one cardinality

- Relationships are bi-directional, so there is also cardinality notation next to the MOVIE entity
- Notice that as the minimum and maximum are both one, this is called mandatory one cardinality (i.e., each VIDEOTAPE of a MOVIE must be a copy of exactly one movie)
- In the following Fig. Some attributes have been added. VIDEOTAPE is represented as a weak entity because it cannot exist unless the original owner movie also exists

Mandatory one cardinality

- The identifier of the MOVIE is 'Movie_Name'
- VIDEOTAPE does not have a unique identifier, however the partial identifier Copy_Number together with Movie_Name would uniquely identify an instance of VIDEOTAPE

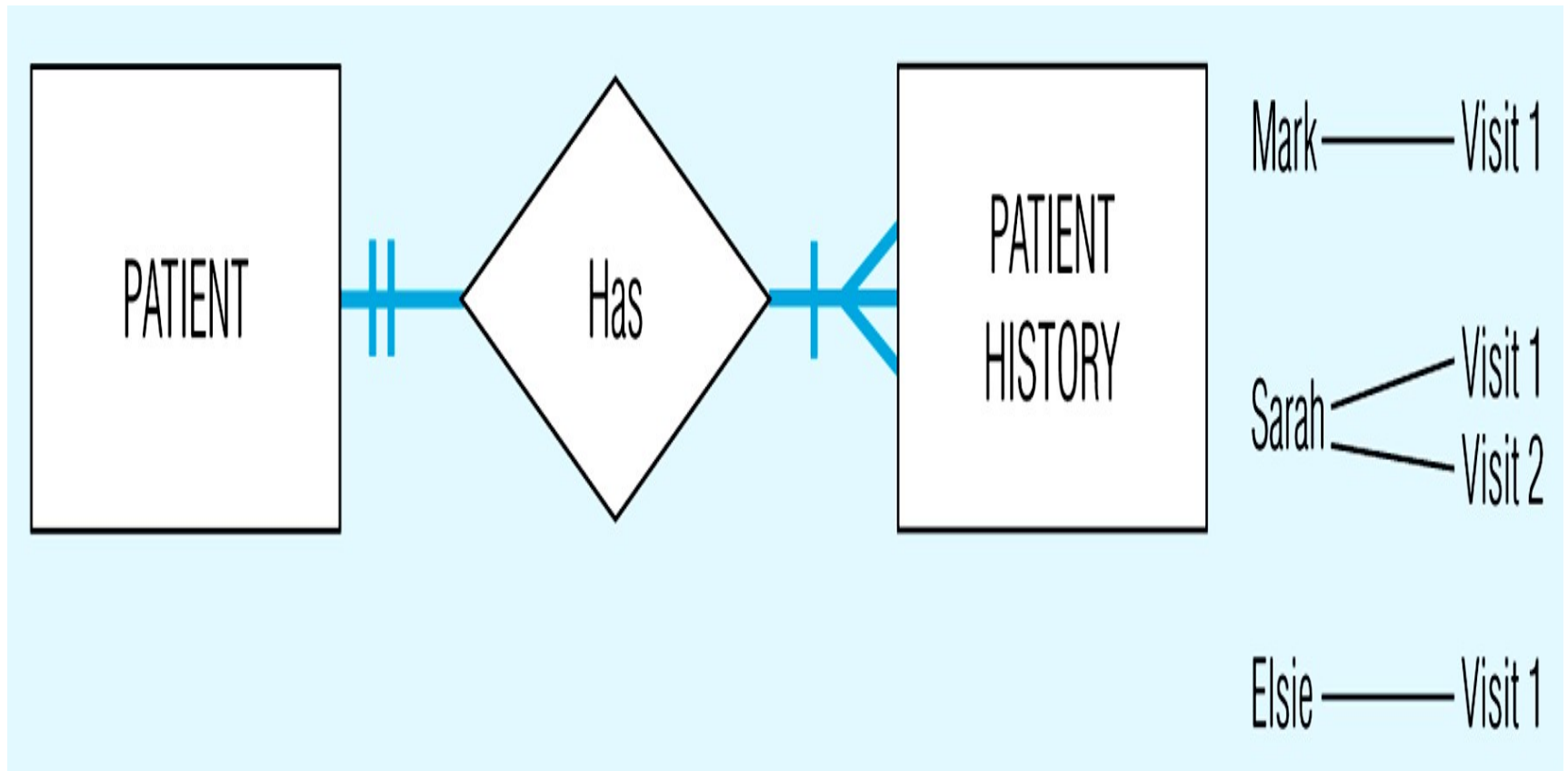
(b) Relationship with cardinality constraints



Example of mandatory cardinality constraints

- Each PATIENT has one or more PATIENT_HISTORIES (the initial patient visit is always recorded as an instance of PATIENT_HISTORY)
- Each instance of PATIENT_HISTORY 'Belongs to' exactly one PATIENT (see following Fig.)

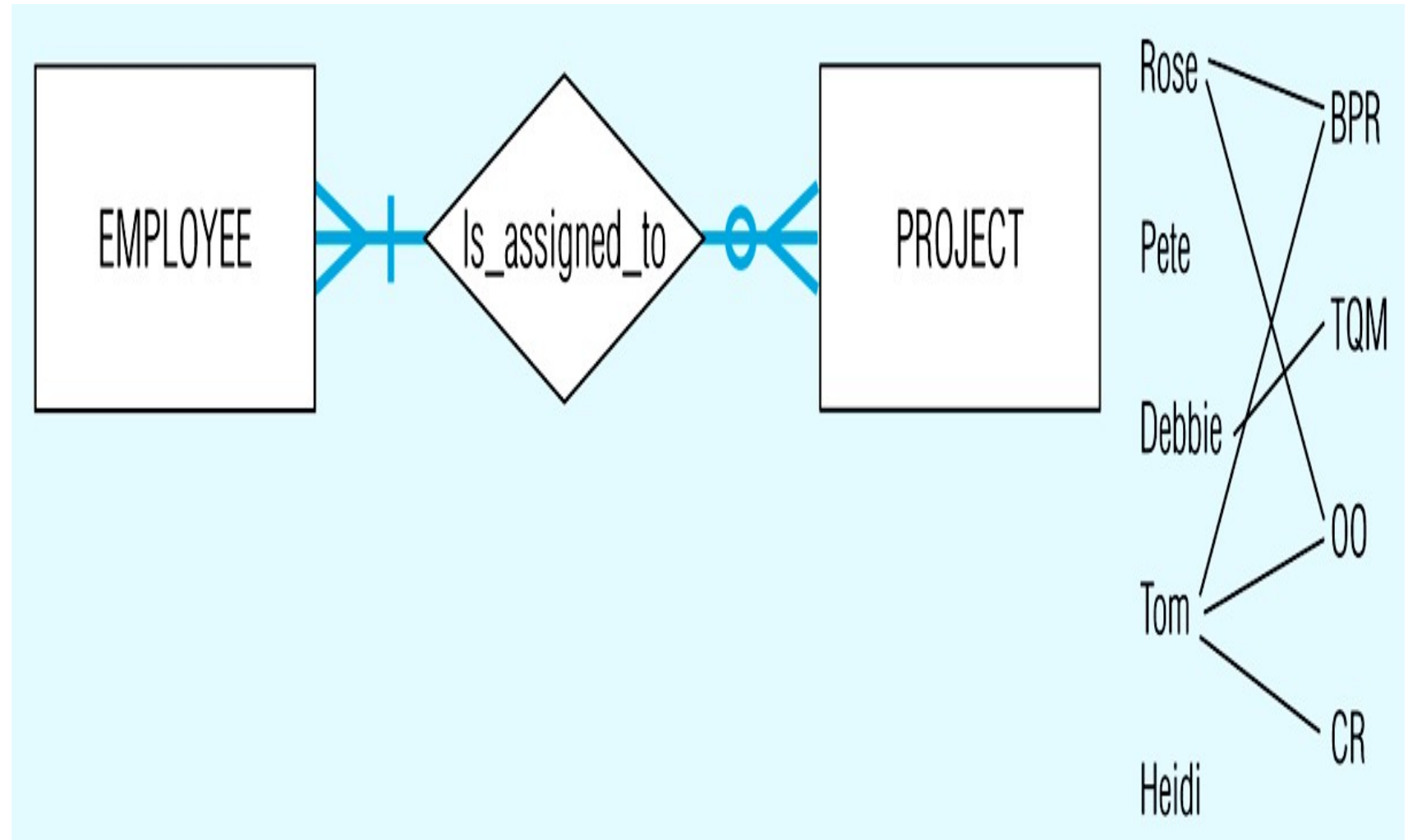
Mandatory cardinalities



Example of one optional, one mandatory cardinality constraint

- EMPLOYEE Is_Assigned_To PROJECT
- Each PROJECT has at least one EMPLOYEE assigned to it (some projects have more than one)
- Each EMPLOYEE may or (optionally) may not be assigned to any existing PROJECT, or may be assigned to one or more PROJECTs (see following Fig.)

One optional, one mandatory cardinality



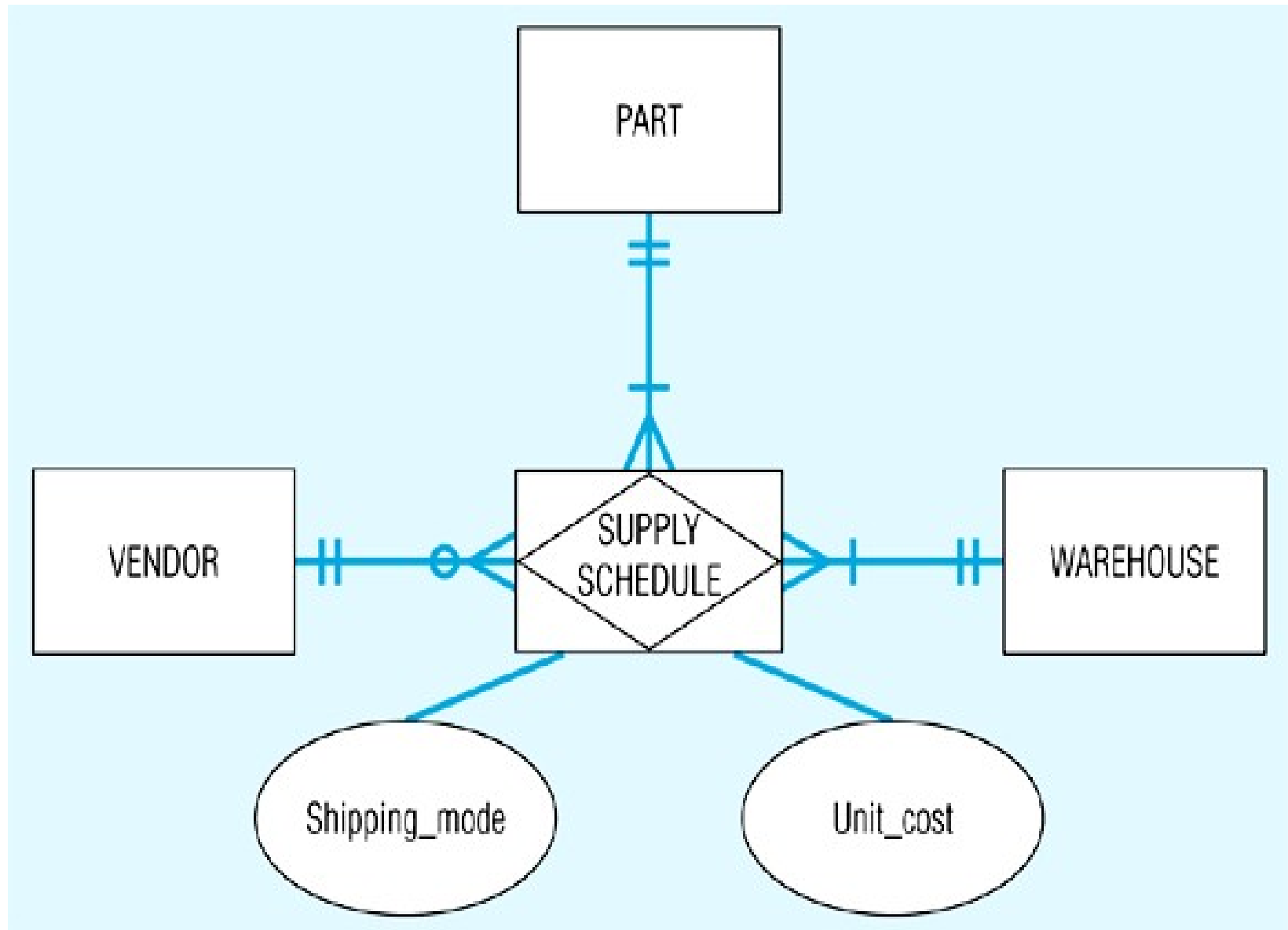
An example using a ternary relationship

- PART and WAREHOUSE are mandatory participants in the relationship, whilst VENDOR is an optional participant
- The cardinality of each of the participating entities is mandatory one, since each SUPPLY_SCHEDULE instance must be related to exactly one instance of each of these participating entity types

An example using a ternary relationship

- Each VENDOR can supply many PARTs to any number of WAREHOUSES, but need not supply any parts
- Each PART can be supplied by any number of VENDORS to more than one WAREHOUSE, but each part must be supplied by at least one vendor to a warehouse
- Each WAREHOUSE can be supplied with any number of PARTS from more than one VENDOR, but each warehouse must be supplied with at least one part

Cardinality constraints in a ternary relationship



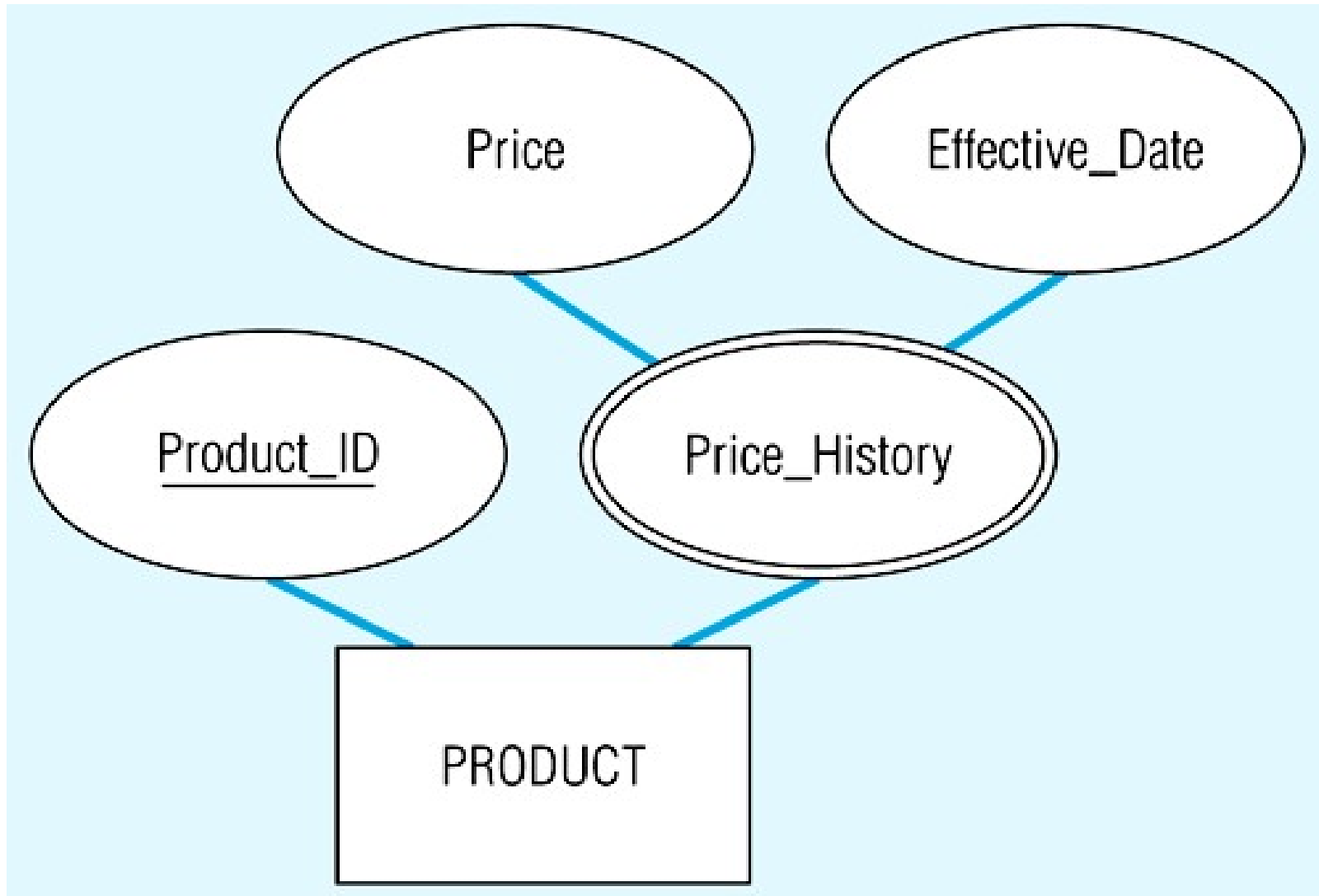
An example using a ternary relationship

- A ternary relationship is not equivalent to three binary relationships
- Unfortunately you cannot draw ternary relationships with many CASE tools
- Instead you must represent ternary relationships as three binaries
- If you are forced to do this, then do not draw the binary relationships with diamonds and make sure the cardinality next to the three strong entities are mandatory one

Modelling time-dependent data

- Some database values change over time (e.g. price)
- We may need to preserve a history of the prices and the time period over which each was in effect
- We can think of a series of prices and the effective date (see Fig.), giving a (composite) multivalued attribute Price_History (that has components 'Price' and 'Effective_Date')

Simple example of time stamping

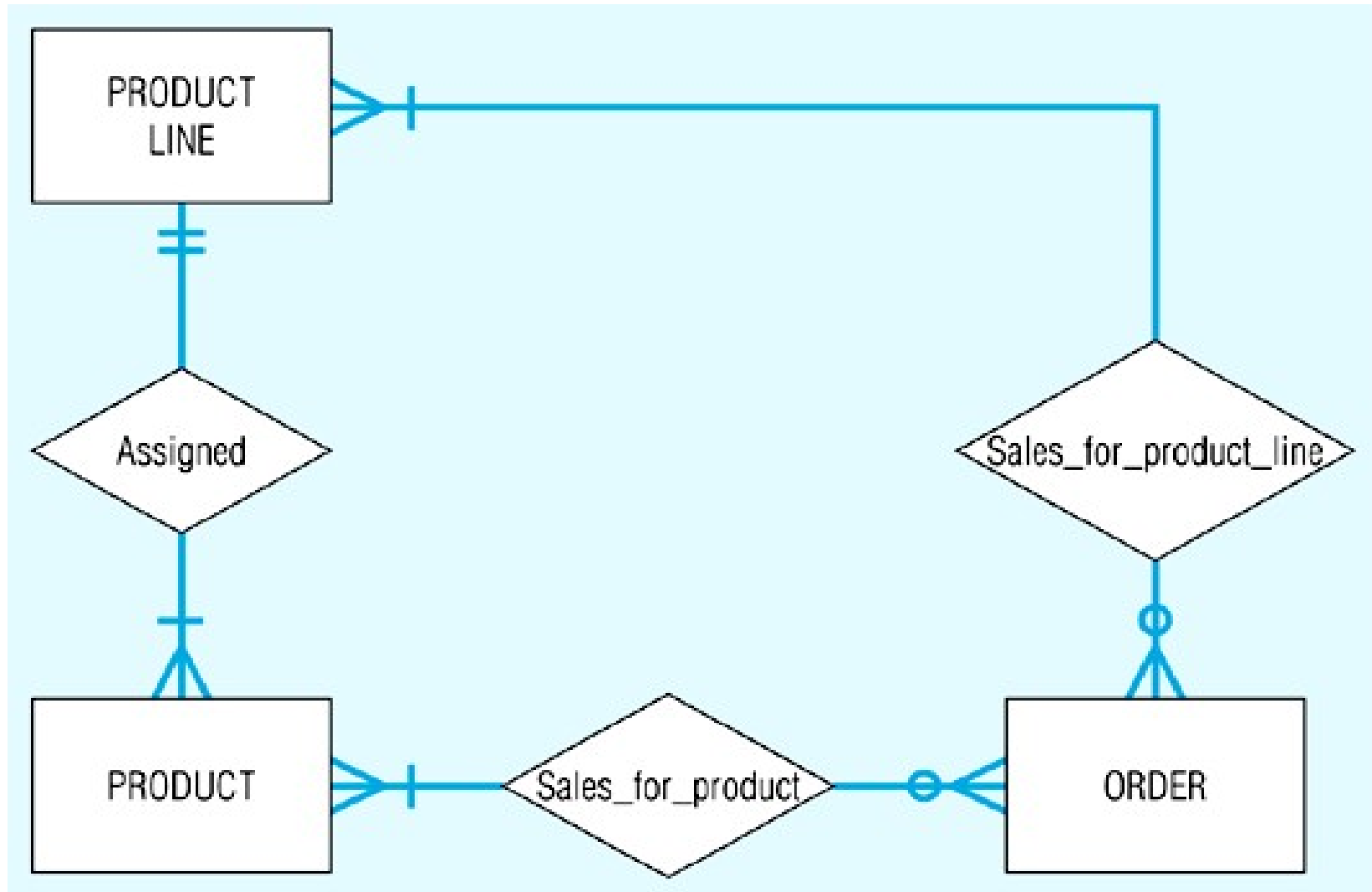


Time stamps

- Are simply time values associated with a data value
- May be recorded to indicate the time the value was entered (transaction time), time the value becomes valid or stops being valid, or the time when critical actions were performed (such as updates, corrections or audits)

More complex time-dependent data

- Suppose that in the middle of the year some PRODUCTS are reassigned to different PRODUCT_LINES, so all sales reports will show cumulative sales for a product based on its current product line, rather than the one at the time of the sale
- To model this, a new relationship Sales_for_Product_Line has been added between ORDER and PRODUCT_LINE, so that as customer orders are processed, they are credited to both the correct product and the correct product line as the time of the sale
- Many current data models are inadequate in handling time-dependent data, but some data-warehousing systems provide explicit designs for time dependent data



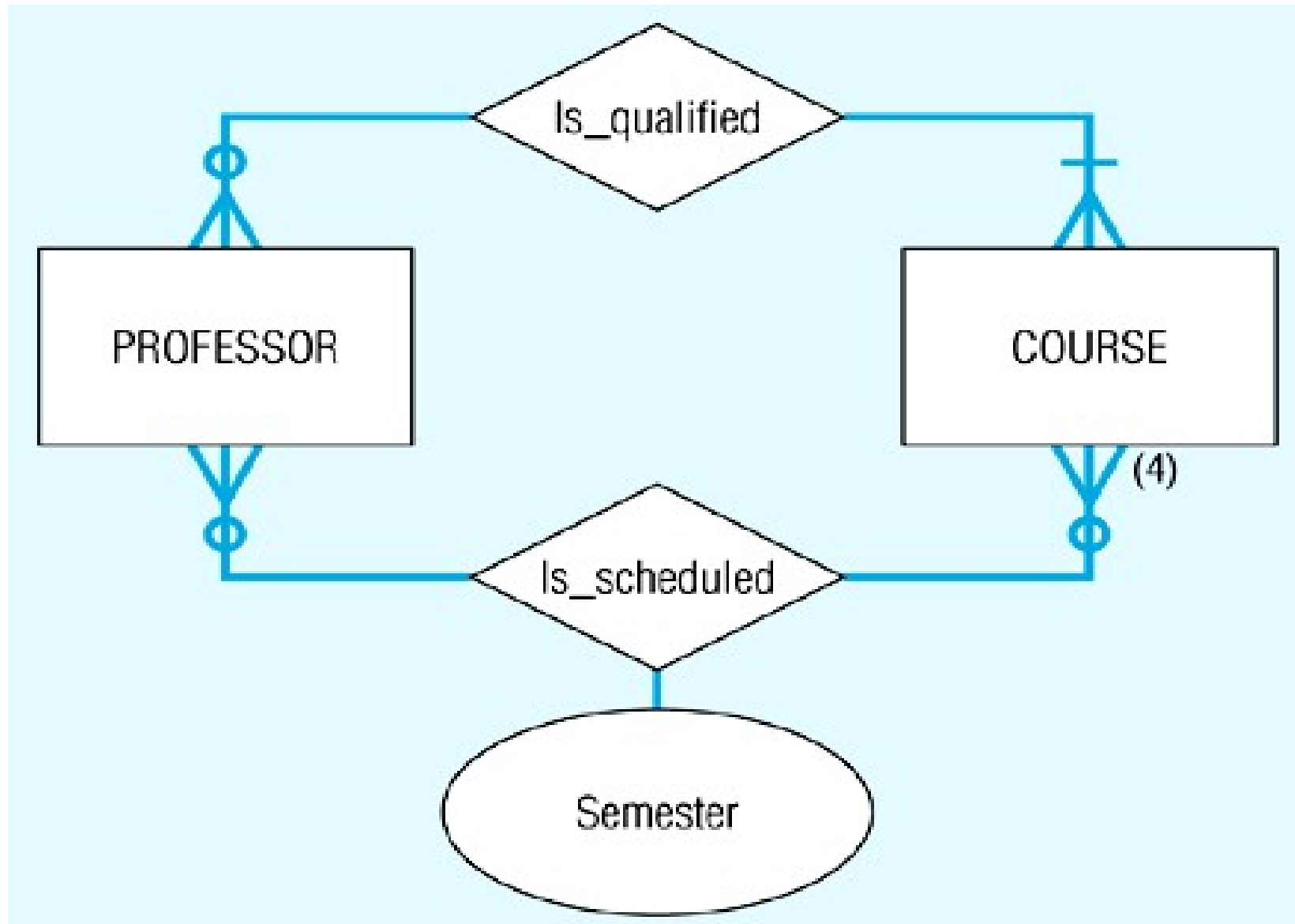
Multiple relationships

- In some situations an organisation may want to model more than one relationship between the same entity types
- The following figure shows two relationships between PROFESSOR and COURSE
- The relationship Is_Qualified associates professors with the courses they are qualified to teach
- A given course may have more than one person qualified to teach it, or (optionally) may not have any qualified instructors (such as a new course)
- Each professor should be qualified to teach at least one course (we hope!)

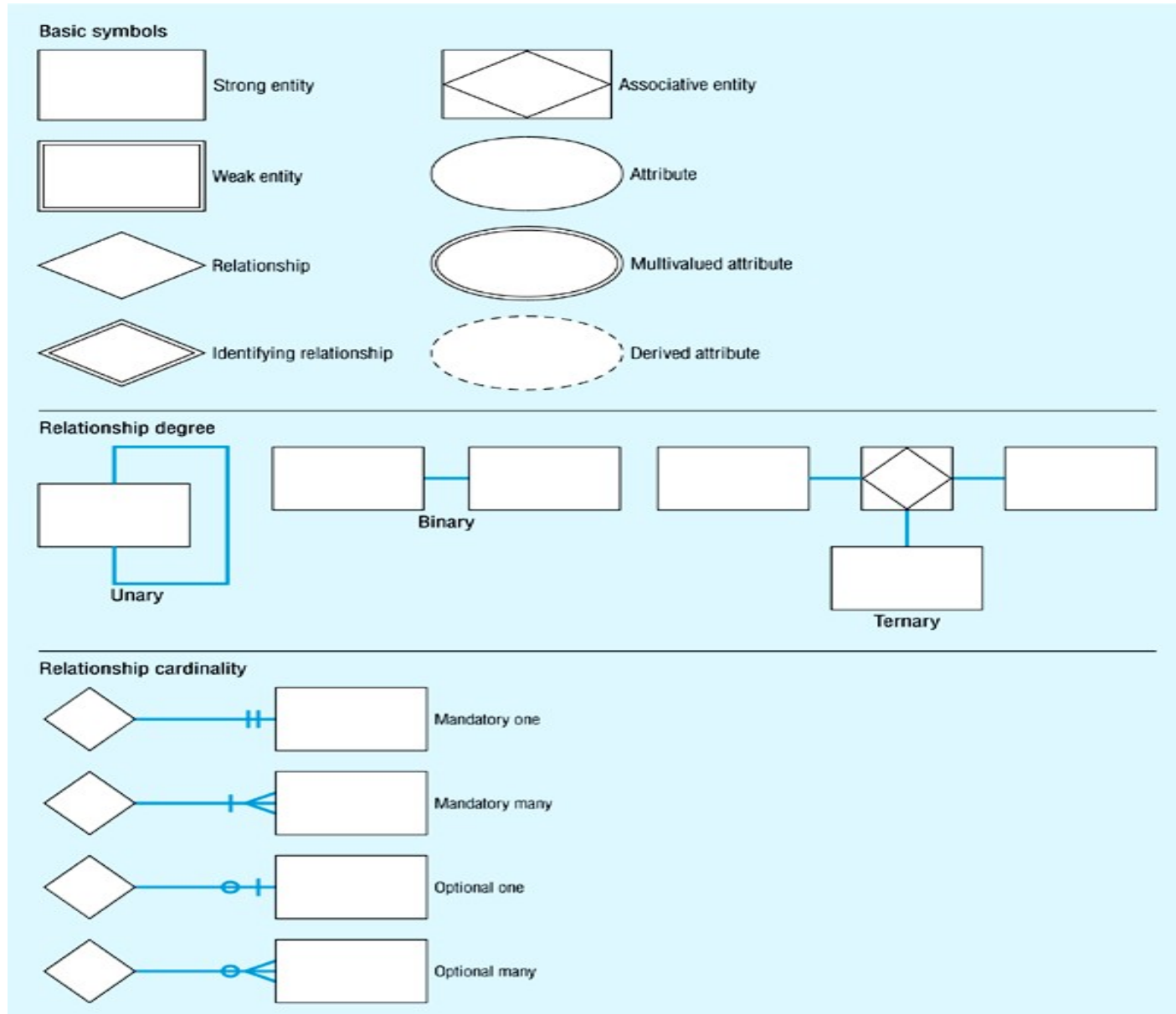
Multiple relationships

- The second relationship in this figure associates professors with the courses they actually teach during a given semester (where the maximum cardinality for a given semester is 4)
- This shows how a fixed constraint (upper or lower) can be recorded
- The attribute 'Semester' (which could be a composite attribute with components 'Semester_Name' and 'Year') is on the relationship Is_Scheduled)

(b) Professors and courses (fixed upon constraint)



Review of Basic E-R Notation



Questions?