

Data 604 – Data Management

Week 2 – Relational Databases and Database Design

Learning Objectives

Understand importance of data modeling

Be able to design a model to 3rd normal form

Model different types of attributes, entities, relationships, and cardinalities

Distinguish unary, binary, and ternary relationships

Convert many-to-many relationships to associative entities

Relational Database

- Codd refers to relational in the mathematical sense where a relation (table) is a set of tuples (rows).
- He proposed the relational model to address weaknesses he observed in the Hierarchical and Network models.
- IBM was slow to adopt for fear of hurting existing business leveraging those models.
- Oracle based their database on this model which became widely popular.

Reference: E.F. Codd 'A Relational Model of Large Shared Data Banks.'

Business Rules

- Are statements that define or constrain some aspect of the business
- Are derived from policies, procedures, events, functions
- Assert business structure
- Control/influence business behavior
- Are expressed in terms familiar to end users
- Are automated through DBMS software

A Good Business Rule Is:

- Declarative – what, not how
- Precise – clear, agreed-upon meaning
- Atomic – one statement
- Consistent – internally and externally
- Expressible – structured, natural language
- Distinct – non-redundant
- Business-oriented – understood by business people

A Good Data Name is:

- Related to business, not technical, characteristics
- Meaningful and self-documenting
- Unique
- Readable
- Composed of words from an approved list
- Repeatable
- Written in standard syntax

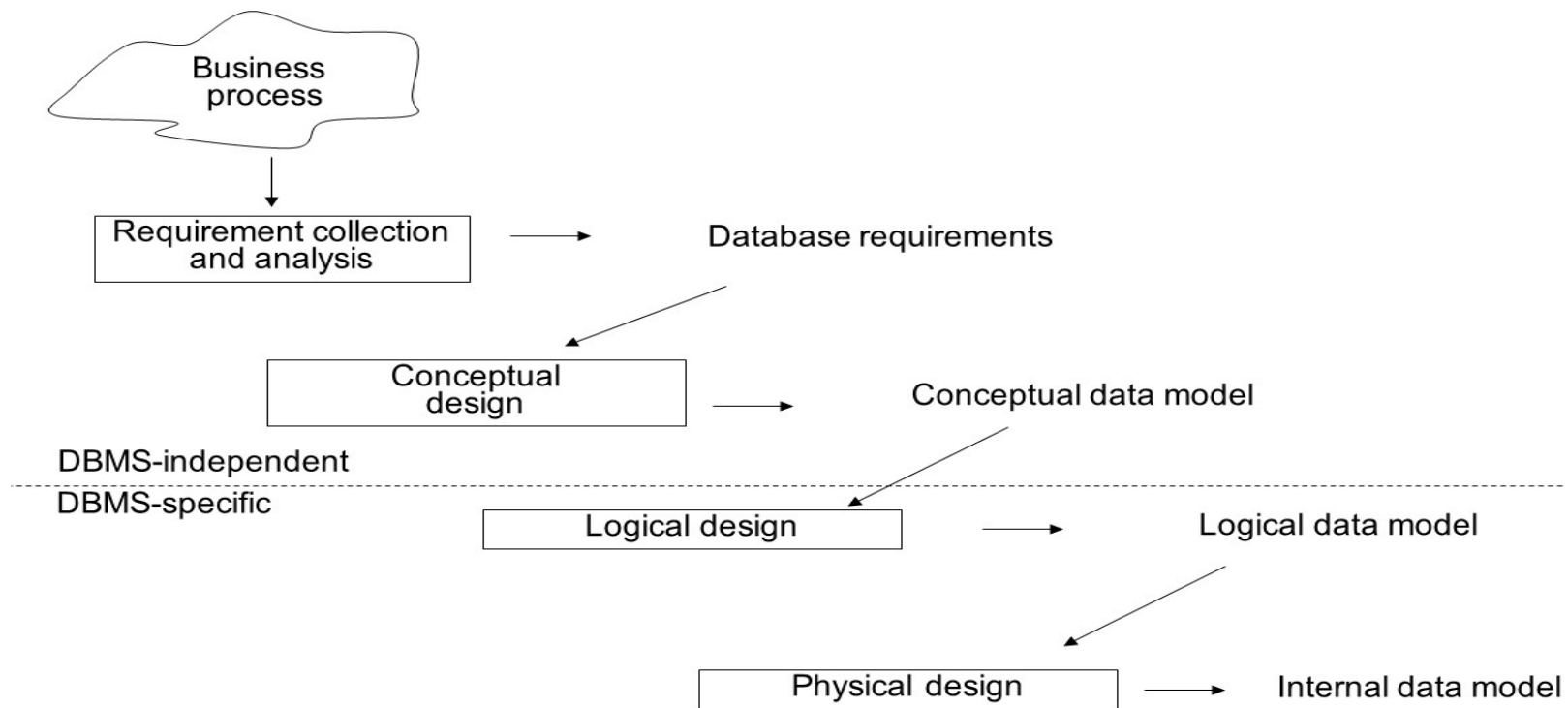
Relation

- A relation is a named, two-dimensional table of data.
- Consists of rows (records) and columns (attribute or field)
- Requirements for a table to qualify as a relation:
 - It must have a unique name.
 - Every attribute value must be atomic (not multivalued, not composite).
 - Every row must be unique (can't have two rows with exactly the same values for all their fields).
 - Attributes (columns) in tables must have unique names.
 - The order of the columns must be irrelevant
 - The order of the rows must be irrelevant.
- All relations are tables, but not all tables are relations. Think of a “table” as simply being a grid with rows and columns. In this sense, a spreadsheet could be a “table”. To qualify as a relation, each row would need to be unique, which implies that each row needs to have a primary key.

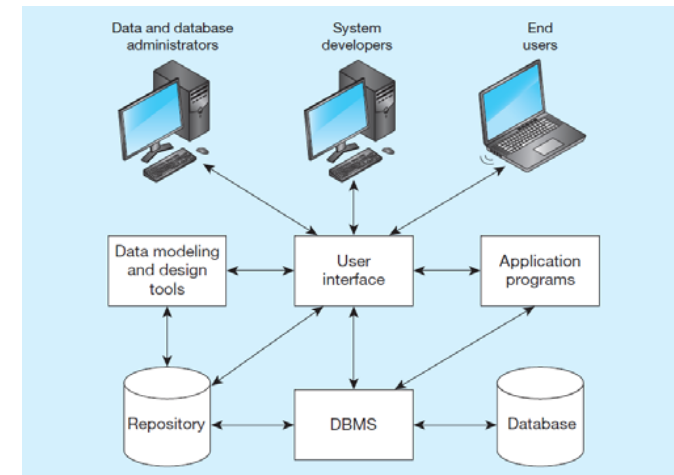
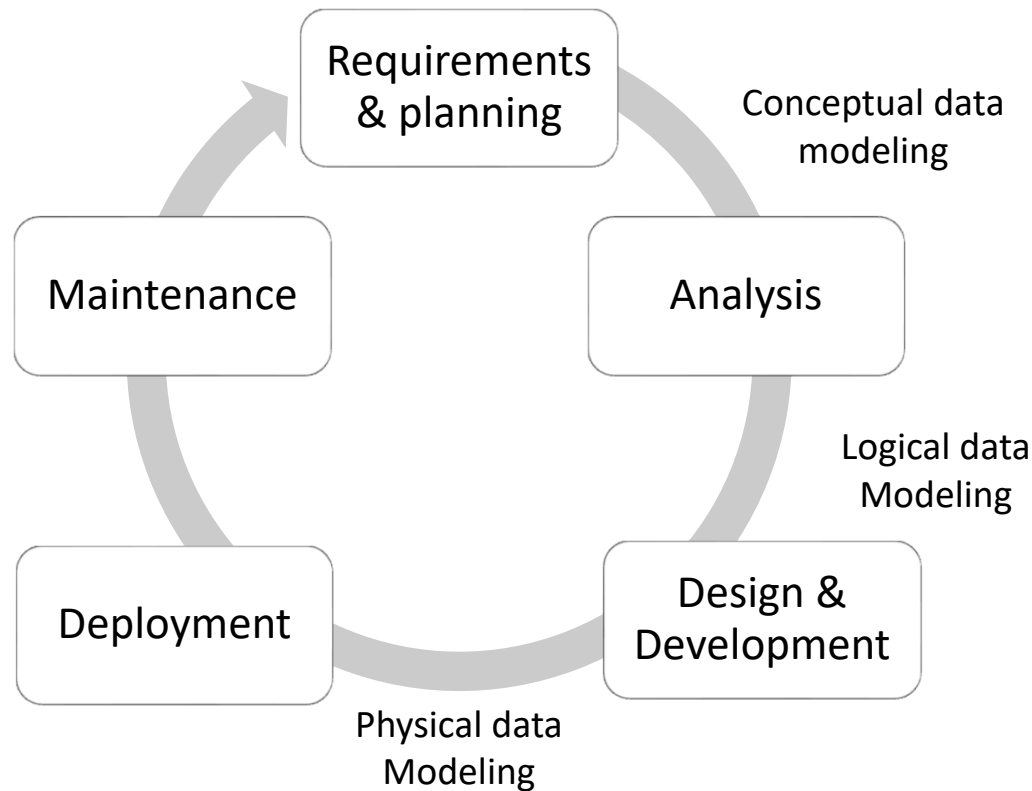
Relational Model

- Data structure
 - Tables (relations), rows, columns
- Data manipulation
 - Powerful SQL operations for retrieving and modifying data
- Data integrity
 - Mechanisms for implementing business rules that maintain integrity of manipulated data

Phases of Database Design



Database development activities during the systems development life cycle (SDLC)

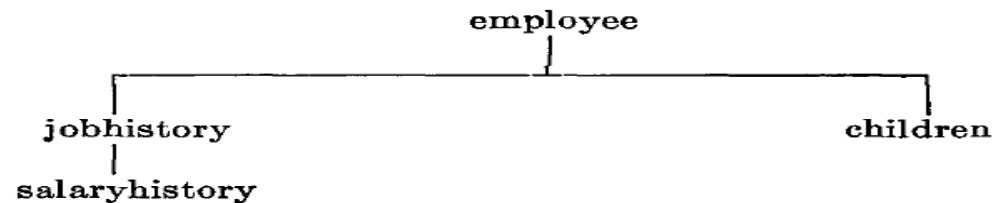


Database Development Life Cycle

- Planning – Requirements, Conceptual Modeling
- Analysis - Conceptual Modeling, Start ER diagram
- Design – Logical Modeling, Normalization, Metadata (Data Dictionary) ,
- Implementation – Run database creation scripts, perform testing, validation testing, production
- Maintenance – Backup plan, index plan, performance monitoring, security

Normalization

- Process to distill sets of data into simple Domains as described by E.F. Codd.



employee (*man#*, name, birthdate, jobhistory, children)
 jobhistory (*jobdate*, title, salaryhistory)
 salaryhistory (*salarydate*, salary)
 children (*childname*, birthyear)

FIG. 3(a). Unnormalized set

employee' (*man#*, name, birthdate)
 jobhistory' (*man#*, *jobdate*, title)
 salaryhistory' (*man#*, *jobdate*, *salarydate*, salary)
 children' (*man#*, *childname*, birthyear)

FIG. 3(b). Normalized set

Image from E.F. CODD Paper: A Relational Model of Data for Large Shared Data Banks

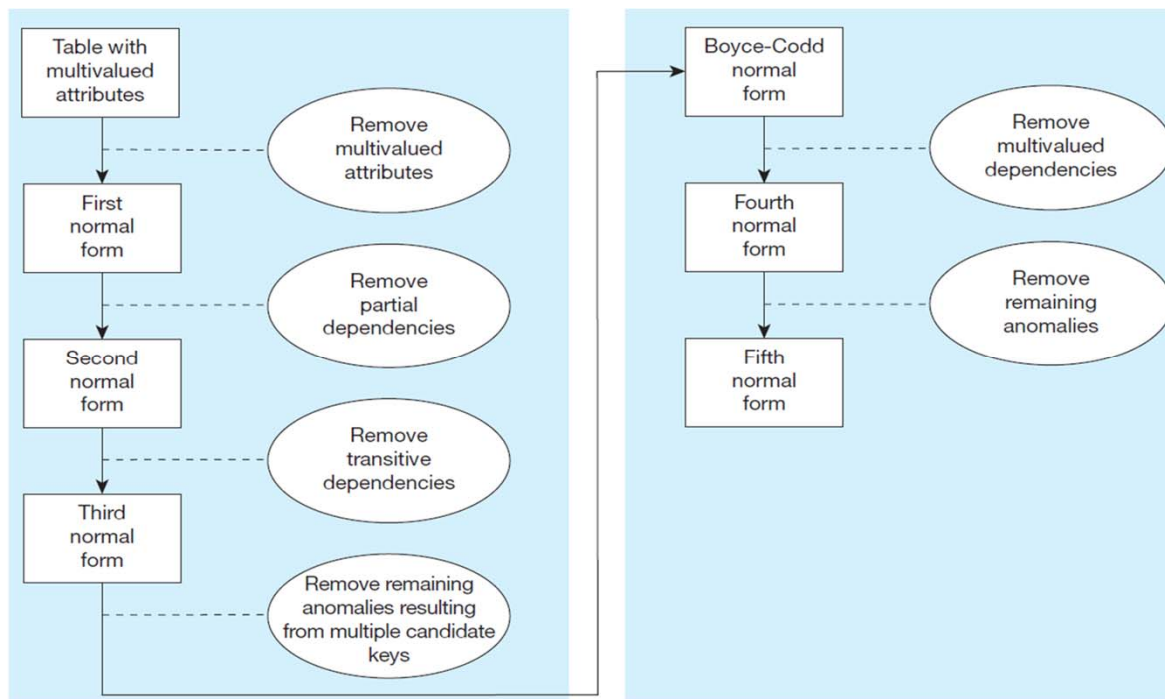
Data Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that **avoid unnecessary duplication of data**
- The process of decomposing relations with anomalies to produce smaller, **well-structured relations**

Well-structured

- Relations that contain minimal data redundancy and allow users to insert, delete, and update rows without causing data inconsistencies
- Goal is to avoid anomalies
 - **Insertion Anomaly** – adding new rows forces user to create duplicate data
 - **Deletion Anomaly** – deleting rows may cause a loss of data that would be needed for other future rows
 - **Modification Anomaly** – changing data in a row forces changes to other rows because of duplication

Steps in Normalization



First Normal Form

- No multivalued attributes
- Every attribute value is atomic
- Slide 17 **is not** in 1st Normal Form (multivalued attributes) → it is not a relation.
- Slide 18 *is* in 1st Normal form.
- **All relations are in 1st Normal Form.**

Not in First Normal Form

OrderID	Order Date	Customer ID	Customer Name	Customer Address	Product ID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2018	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Table with multivalued attributes, not in 1st normal form.

This is **not** a relation.

Now in First Normal Form

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product Standard Price	Ordered Quantity
1006	10/24/2018	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2018	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
1007	10/25/2018	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

This is a relation, but not yet a well-structured one.

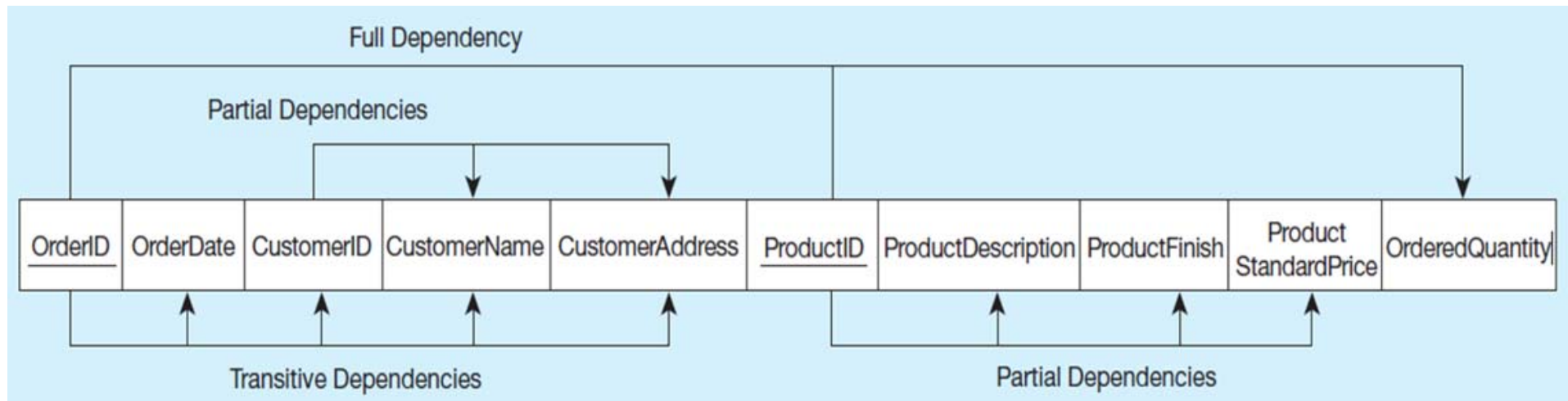
Anomalies that happen in 1NF

- **Insertion** – if new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication
- **Deletion** – if we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price
- **Update** – changing the price of product ID 4 requires update in multiple records

Second Normal Form

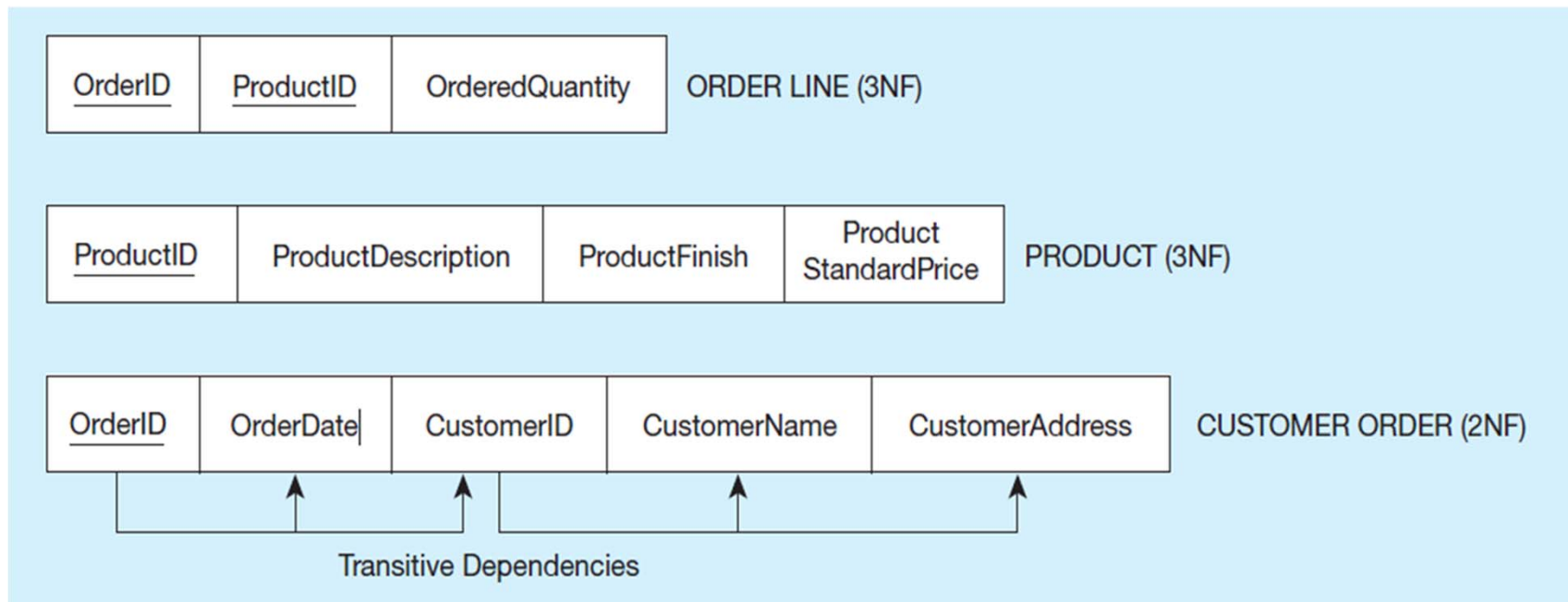
- 1NF plus **every non-key attribute is fully functionally dependent on the ENTIRE primary key**
 - Every non-key attribute must be defined by the entire key, not by only part of the key
 - No partial functional dependencies

Not in Second Normal Form



OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress
 CustomerID → CustomerName, CustomerAddress
 ProductID → ProductDescription, ProductFinish, ProductStandardPrice
 OrderID, ProductID → OrderedQuantity
 Has partial dependencies, therefore, **not** in 2nd Normal Form

Now in Second Normal Form

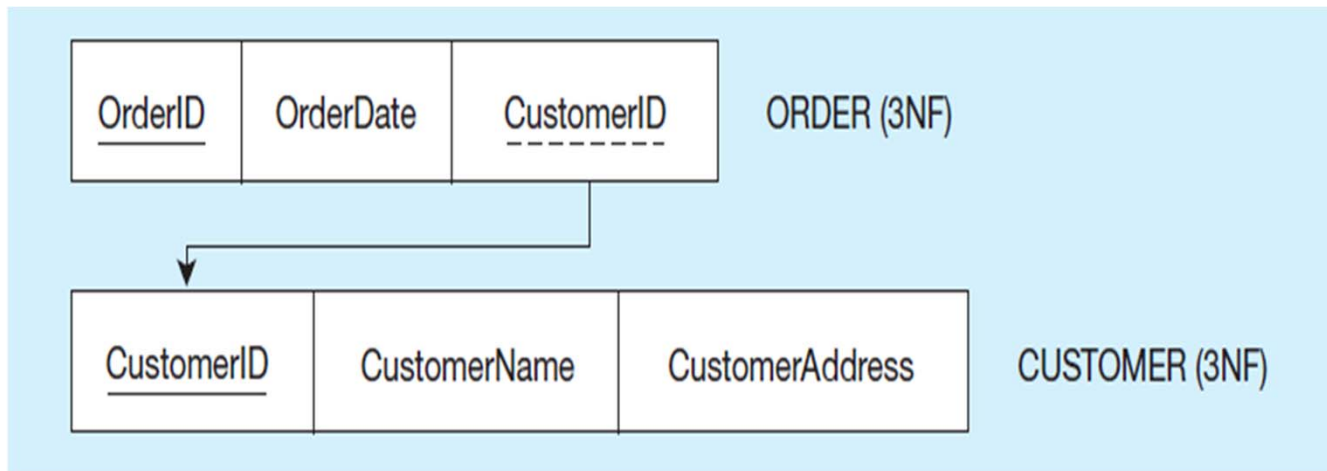


Partial dependencies are removed, but there are still transitive dependencies

Third Normal Form

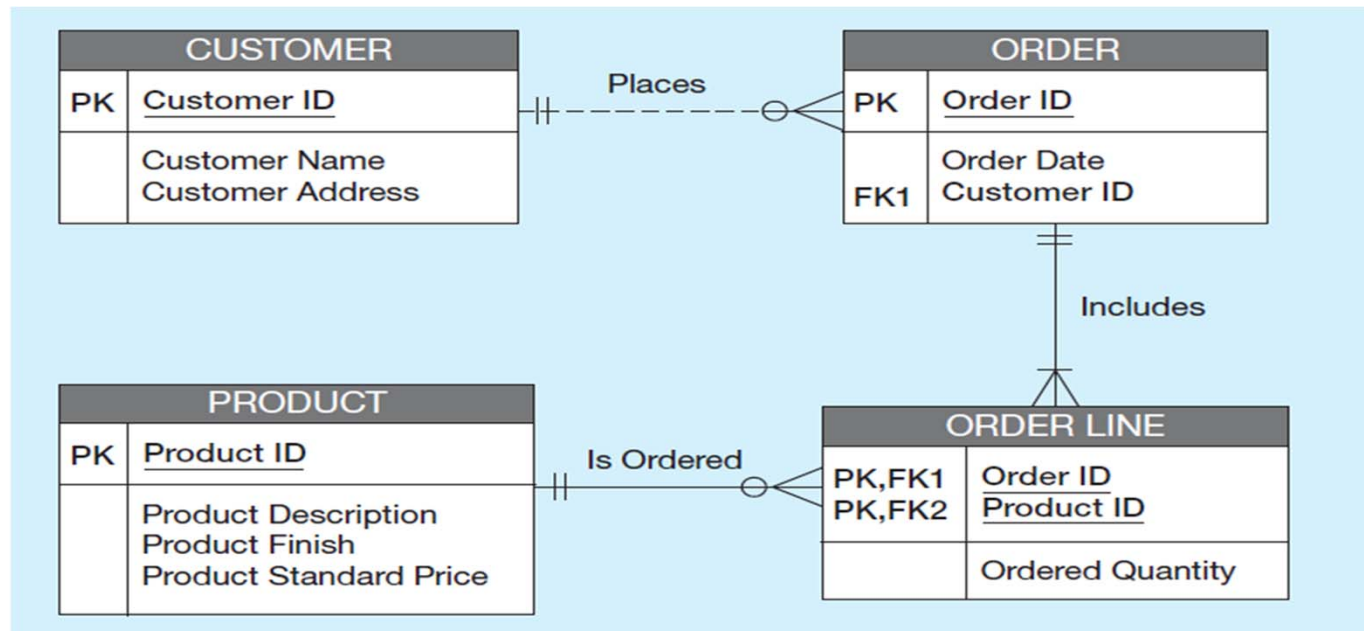
- 2NF PLUS **no transitive dependencies** (functional dependencies on non-primary-key attributes)
- Functional Dependency: the value of one attribute (the **determinant**) determines the value of another attribute
- Note: This is called transitive, because the primary key is a determinant for another non key attribute, which in turn is a determinant for a third
- Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as foreign key in the old table

Transitive Dependencies



Getting it into Third Normal Form
Transitive dependencies are removed.

Relational Schema



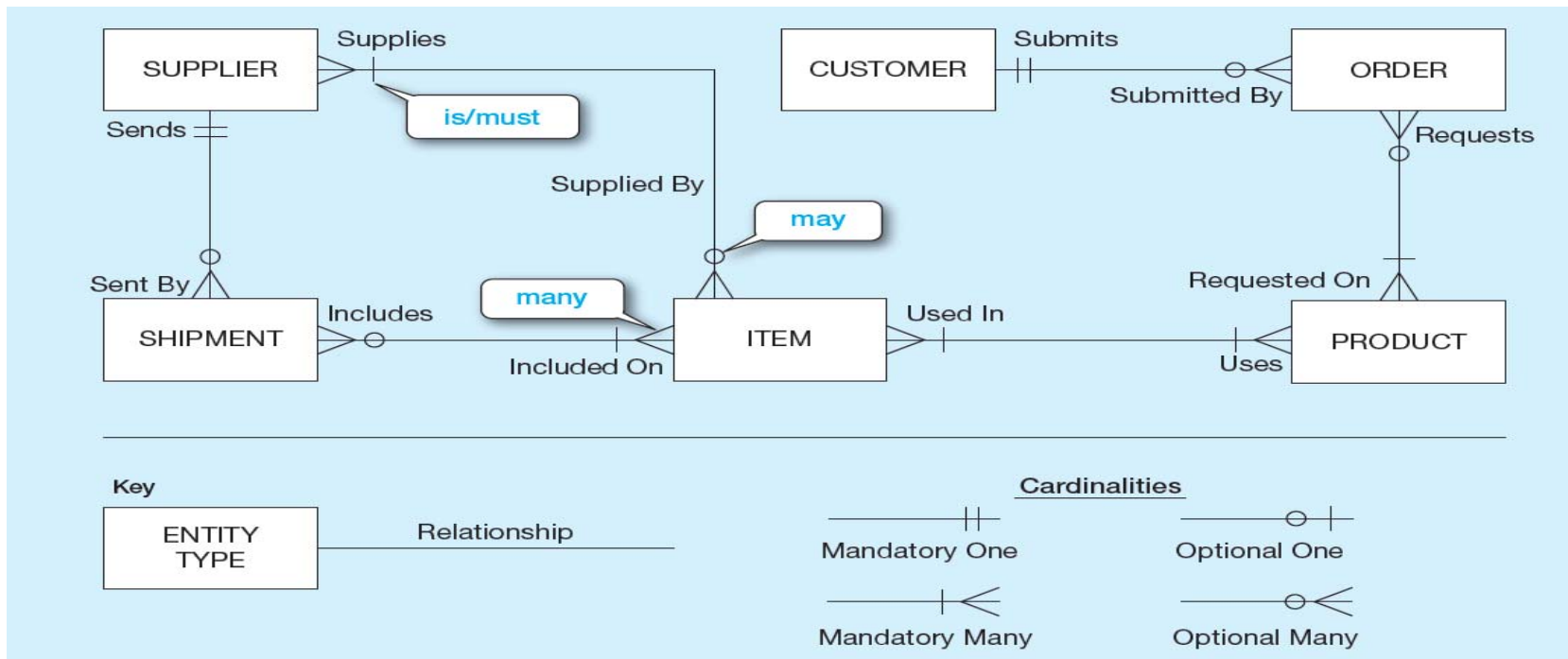
- Video demo

<https://www.cockroachlabs.com/docs/stable/foreign-key.html>

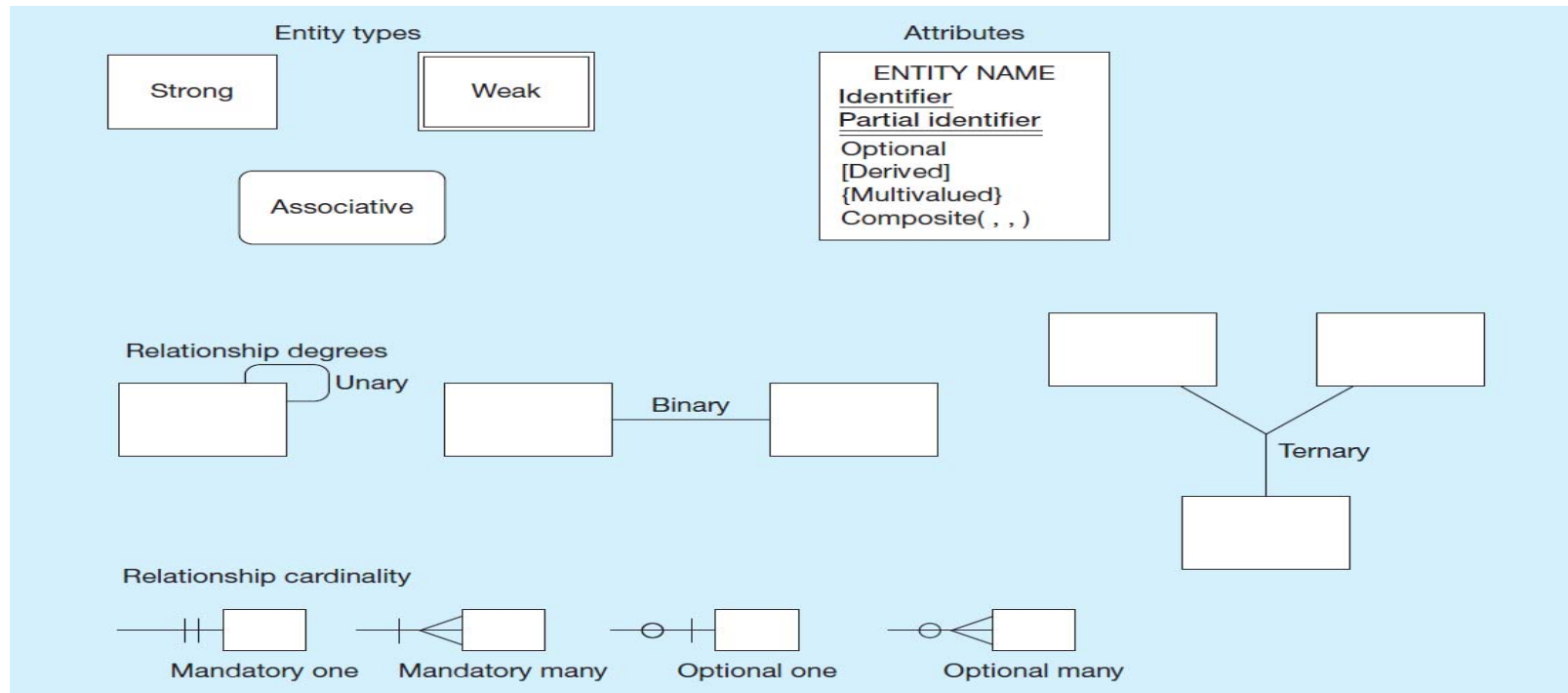
E-R Model Constructs

- Entities:
 - Entity Type – collection of entities (often corresponds to a table)
 - Entity instance – person, place, object, event, concept (often corresponds to a row in a table)
- Attributes:
 - Properties or characteristics of an entity or relationship type (often corresponds to a field in a table)
- Relationships:
 - Relationship type – category of relationship; link between entity types
 - Relationship instance – link between entities (corresponds to primary key–foreign key equivalencies in related tables)

Sample ER Diagram



ER Notation



Entity

- **Should Be:**

- An object that will have many instances in the database
- An object that will be composed of multiple attributes
- An object that we are trying to model

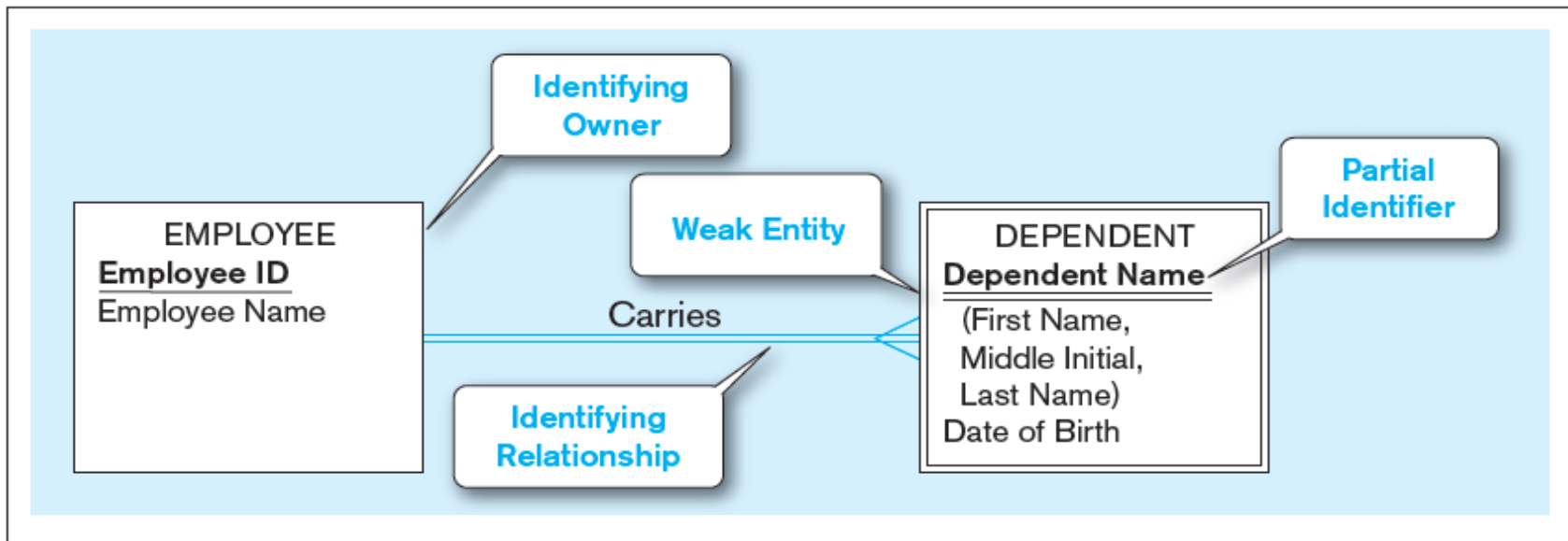
- **Should Not Be:**

- A user of the database system
- An output of the database system (e.g., a report)

Strong vs Weak Entity

- Strong entity
 - exists independently of other types of entities
 - has its own unique identifier
 - identifier underlined with single line
- Weak entity
 - dependent on a strong entity (identifying owner); cannot exist on its own
 - does not have a unique identifier (only a partial identifier)
 - entity box and partial identifier have double lines
- Identifying relationship
 - links strong entities to weak entities

Weak Entity



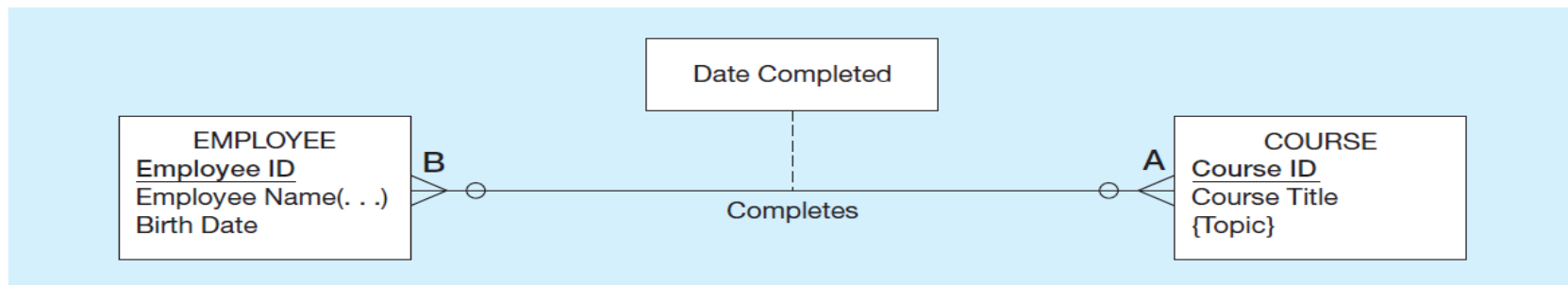
Associative Entities

- When should a **relationship with attributes** instead be an **associative entity**?
 - All relationships for the associative entity should be many
 - The associative entity could have meaning independent of the other entities
 - The associative entity preferably has a unique identifier, and should also have other attributes
 - The associative entity may participate in other relationships other than the entities of the associated relationship
 - Convert ternary relationships to associative entities

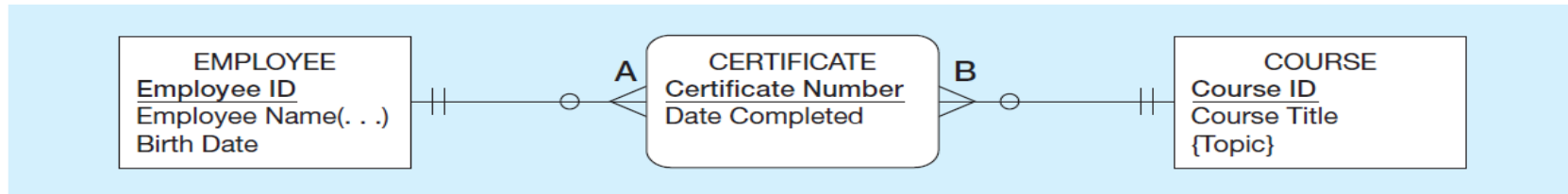
Associative Entities

An associative entity is an entity. It has attributes. It is also a relationship. It serves to link other entities together in a many-to-many relationship.

(a) Attribute on a relationship



(b) An associative entity (CERTIFICATE)



Attributes

- Attribute – property or characteristic of an entity or relationship type
- Classifications of attributes:
 - Required versus Optional
 - Simple versus Composite
 - Single-Valued versus Multivalued
 - Stored versus Derived
 - Identifier

Required vs Optional Attribute

Required – must have a value for every entity (or relationship) instance with which it is associated

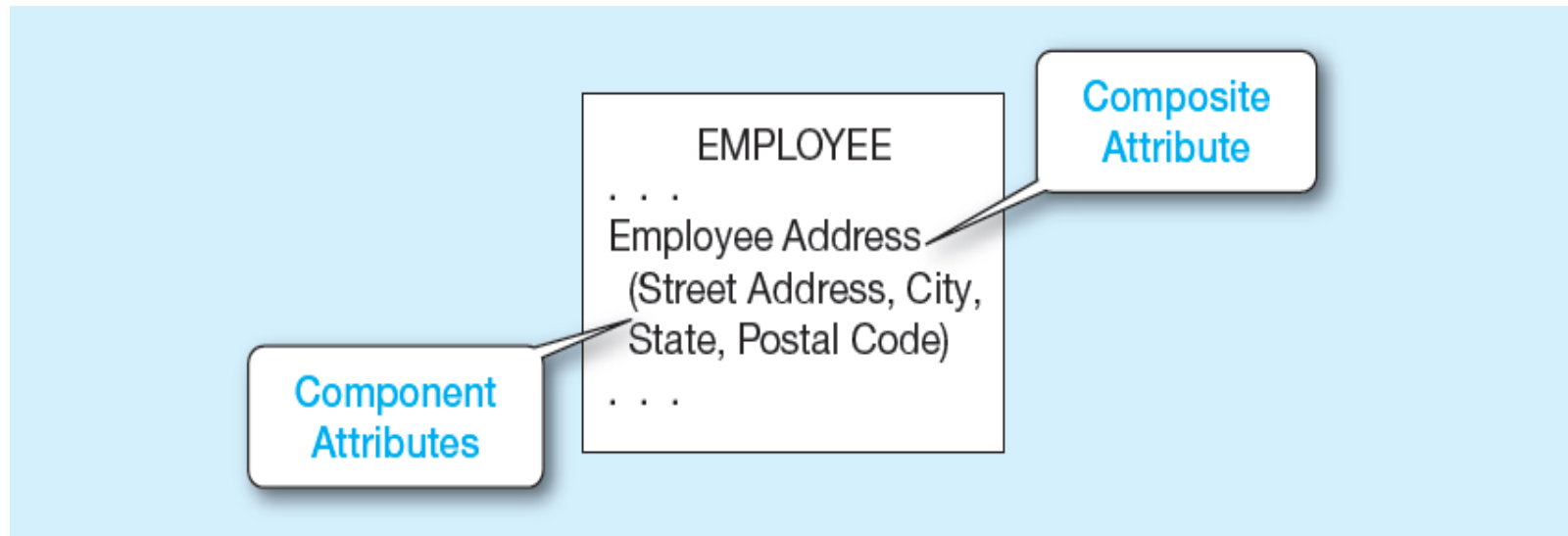
Optional – may not have a value for every entity (or relationship) instance with which it is associated (is null)

Entity type: STUDENT

Attributes	Attribute Data Type	Required or Optional	Example Instance	Example Instance
Student ID	CHAR (10)	Required	28-618411	26-844576
Student Name	CHAR (40)	Required	Michael Grant	Melissa Kraft
Home Address	CHAR (30)	Required	314 Baker St.	1422 Heft Ave
Home City	CHAR (20)	Required	Centerville	Miami
Home State	CHAR (2)	Required	OH	FL
Home Zip Code	CHAR (9)	Required	45459	33321
Major	CHAR (3)	Optional	MIS	

Composite Attribute

- **Composite attribute** – An attribute that has meaningful component parts (sub-attributes)



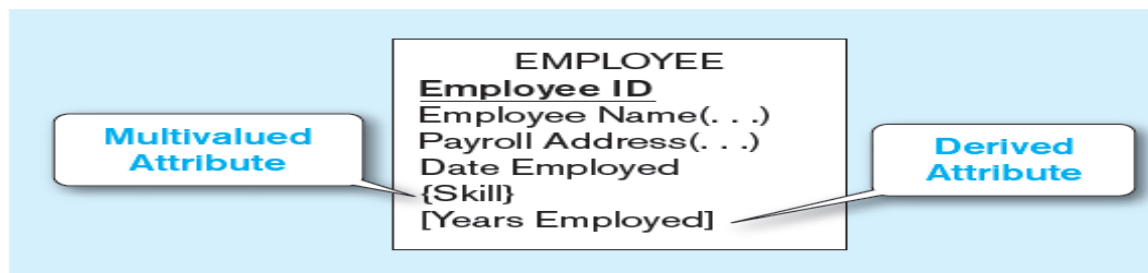
Multivalued and Derived Attributes

- **Multivalued**

- May take on more than one value for a given entity (or relationship) instance
- An employee can have more than one skill

- **Derived**

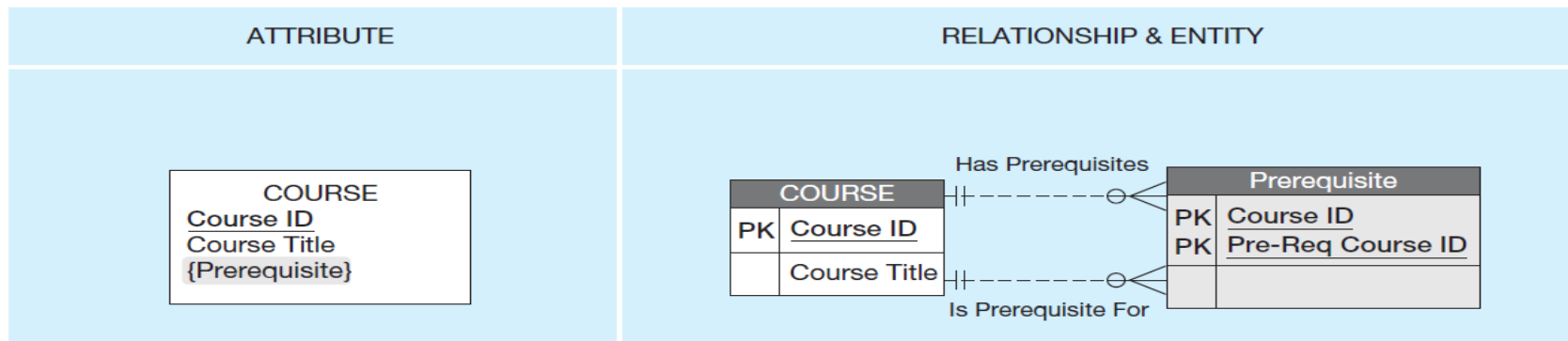
- Values can be calculated from related attribute values (not physically stored in the database)
- Years employed calculated from date employed and current date



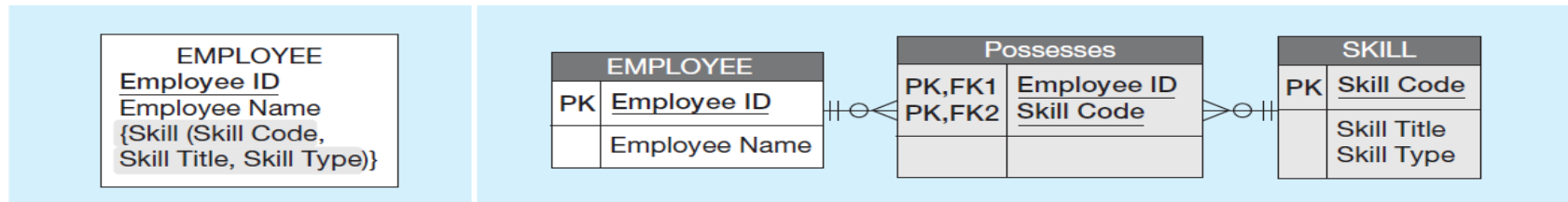
Multivalued Attributes

Multivalued attributes can be represented as relationships

(a) Multivalued attribute versus relationships via bill-of-materials structure



(b) Composite, multivalued attribute versus relationship



Key Fields

- Keys are special fields that serve two main purposes:
 - **Primary keys** are **unique** identifiers of the relation. Examples include employee numbers, social security numbers, etc. **This guarantees that all rows are unique.**
 - **Foreign keys** are identifiers that enable a **dependent** relation (on the many side of a relationship) to refer to its **parent** relation (on the one side of the relationship).
- Keys can be **simple** (a single field) or **composite** (more than one field).
- Keys are usually used as indexes to speed up the response to user queries.

Integrity Constraints

- Domain Constraints
 - Allowable values for an attribute (includes data types and restrictions on values)
- Entity Integrity
 - No primary key attribute may be null. All primary key fields **MUST** contain data values.
- Referential Integrity
 - Rules that maintain consistency between the rows of two related tables.

Referential Integrity

- **Referential Integrity** – rule states that any foreign key value (on the relation of the many side) **MUST** match a primary key value in the relation of the one side. (Or the foreign key can be null.)
 - For example: Delete Rules
 - **Restrict** – don't allow delete of “parent” side if related rows exist in “dependent” side
 - **Cascade** – automatically delete “dependent” side rows that correspond with the “parent” side row to be deleted
 - **Set-to-Null** – set the foreign key in the dependent side to null if deleting from the parent side → not allowed for weak entities

Primary Key

- Identifier (Primary Key) – an attribute (or combination of attributes) that uniquely identifies individual instances of an entity type
- Simple versus Composite Identifier
- Candidate Identifier (candidate or alternate key) – an attribute that could be an identifier; it satisfies the requirements for being an identifier:
 - ☐ A unique identifier. One of the candidate keys will become the primary key
E.g., perhaps there is both student_ID and email address in a table...in this case both are candidate keys.
 - ☐ Each non-key field/attribute is functionally dependent on every candidate key.

Choosing Identifiers/Primary Key

- Choose Identifiers that
 - Will not change in value
 - Will not be null
- Avoid intelligent identifiers (e.g., containing locations or people that might change)
- Substitute new, simple keys for long, composite key
- Book suggests SSN or credit card number but I do not recommend

Identifiers - Autogenerated

- Some database systems can autogenerate keys
- Simple autogenerated keys offer best performance
- Reduces risk of data key errors (what if someone types in the wrong SSN?)
- Not recommended for distributed environment, this could result in duplicate records with two different keys

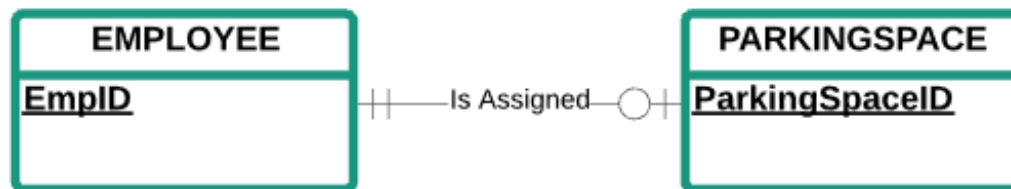
Modeling Relationships

- Relationship Types vs. Relationship Instances
 - The relationship type is modeled as lines between entity types. The relationship instance is between specific entity instances
- Two entities can have more than one type of relationship between them (multiple relationships) e.g. Each staff member belongs to one office
Each office has only one manager (a staff member)
- Associative Entity – combination of relationship and entity

- Relationships can have attributes, these describe features pertaining to the association between the entities in the relationship

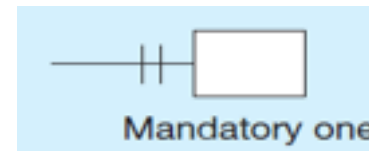
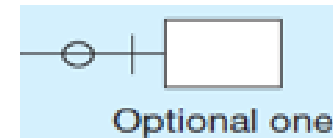
Example: An Employee is assigned a Parking Space

- Identify the entities and its type - Employee and Parking Space
- Identify the relationship – “is assigned”, specify how attributes are connected
- Identify the degree of the relationship – Binary (involves 2 entities)
- Identify the cardinality (bi-directional) – one to one



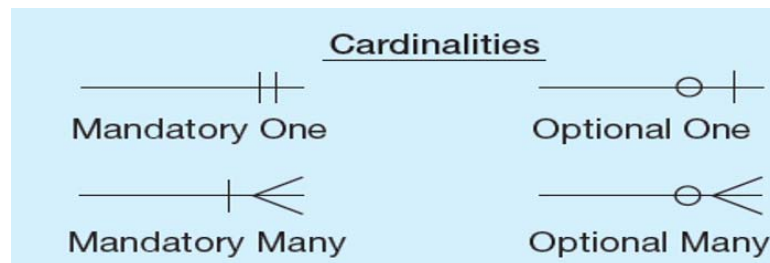
Cardinality

- Cardinality Constraints — the number of instances of one entity that can or must be associated with each instance of another entity
- Minimum Cardinality
 - If zero, then optional
 - If one or more, then mandatory
- Maximum Cardinality
 - The maximum number



Cardinality of Relationships

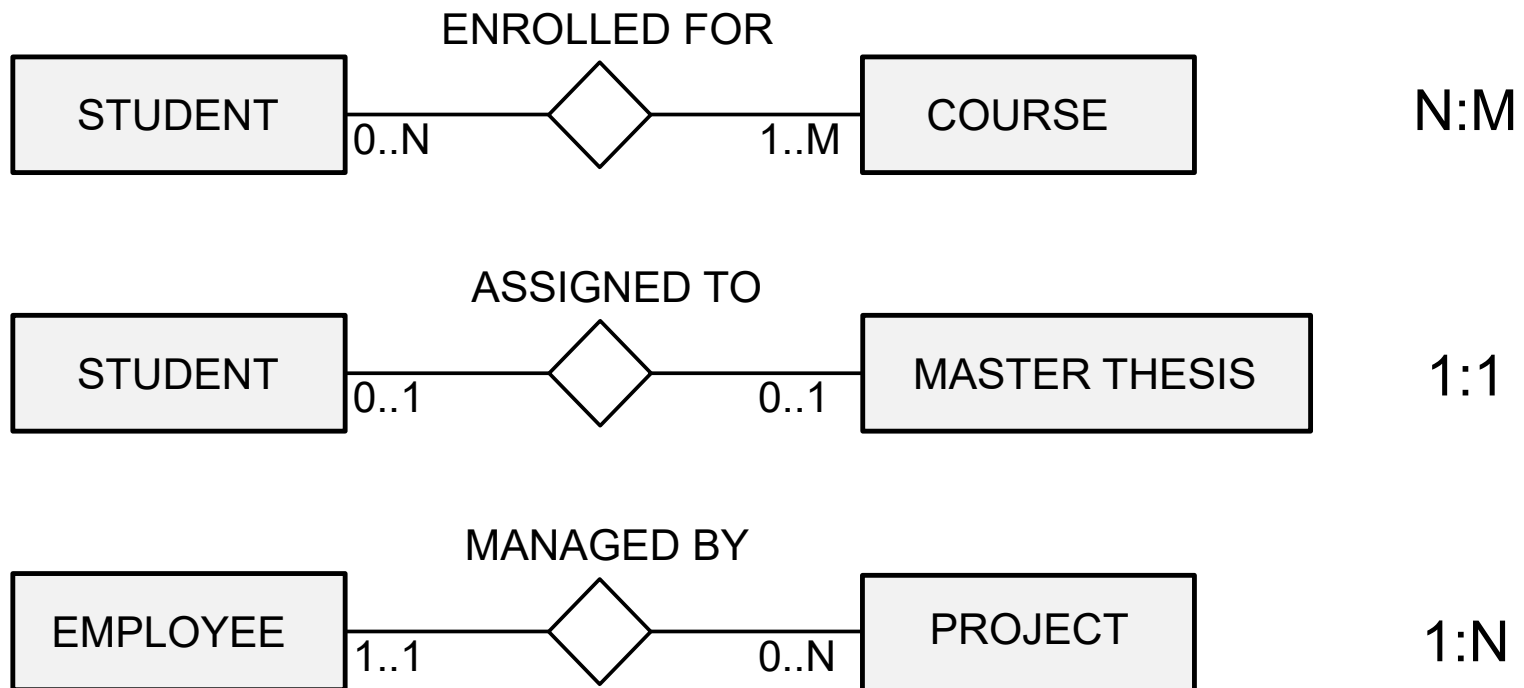
- One-to-One
 - Each entity in the relationship will have exactly one related entity
- One-to-Many
 - An entity on one side of the relationship can have many related entities, but an entity on the other side will have a maximum of one related entity
- Many-to-Many
 - Entities on both sides of the relationship can have many related entities on the other side



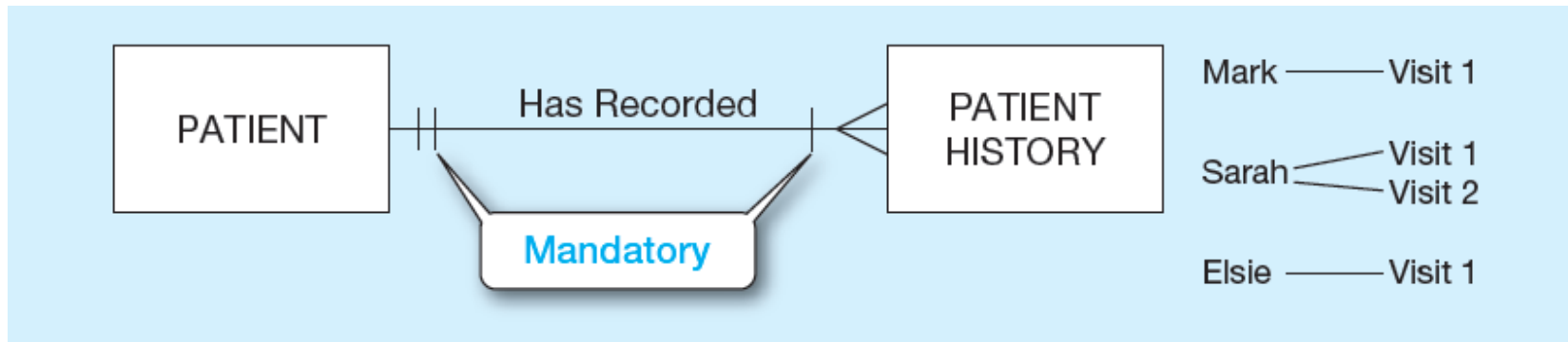
Cardinalities

- Every relationship type can be characterized in terms of its cardinalities, which specify the minimum or maximum number of relationship instances that an individual entity can participate in
- Minimum cardinality can be 0 or 1
 - If 0: partial participation
 - If 1: total participation or existence dependency
- Maximum cardinality can be 1 or N
- Relationship types are often characterized by their maximum cardinalities
 - Four options for binary relationship types: 1:1, 1:N, N:1 and M:N.

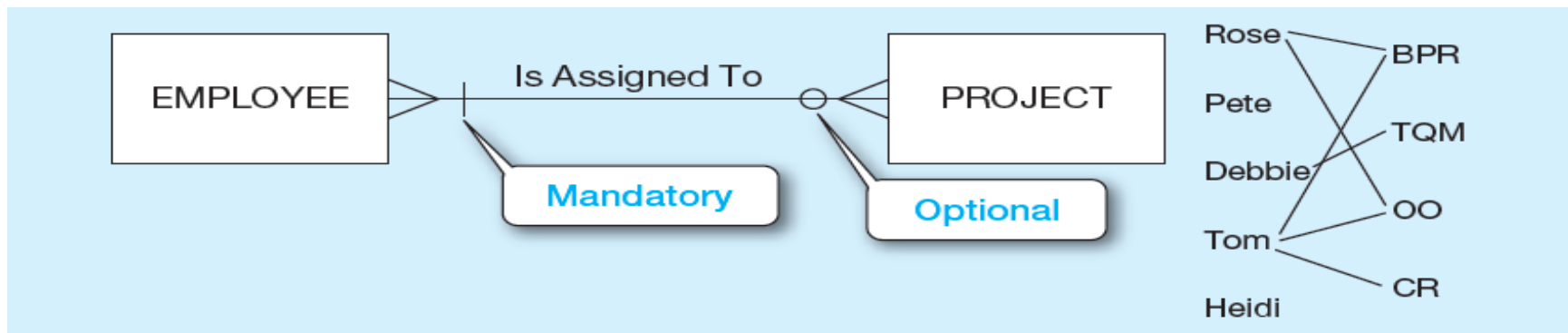
Cardinalities



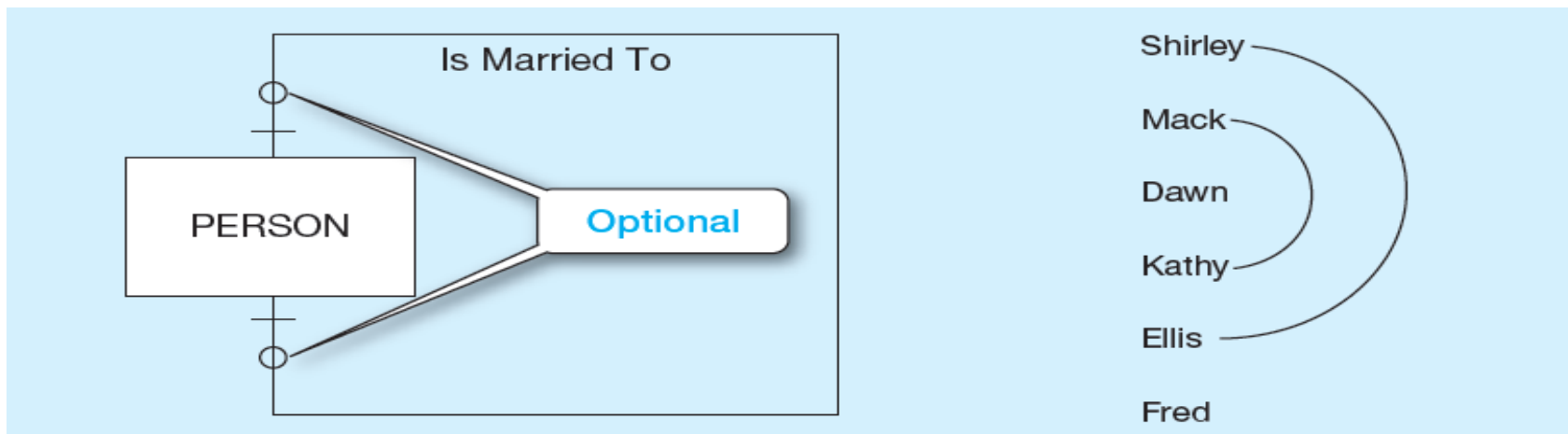
Mandatory Cardinality



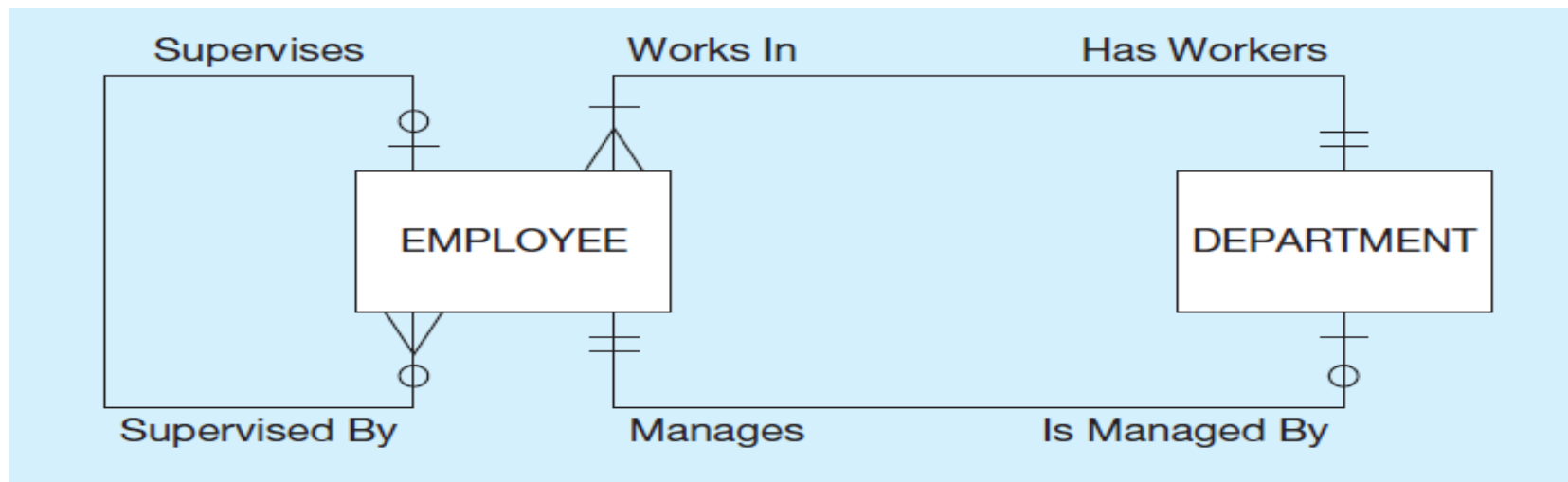
One Mandatory, One Optional



Optional Cardinality



Multiple Relationships

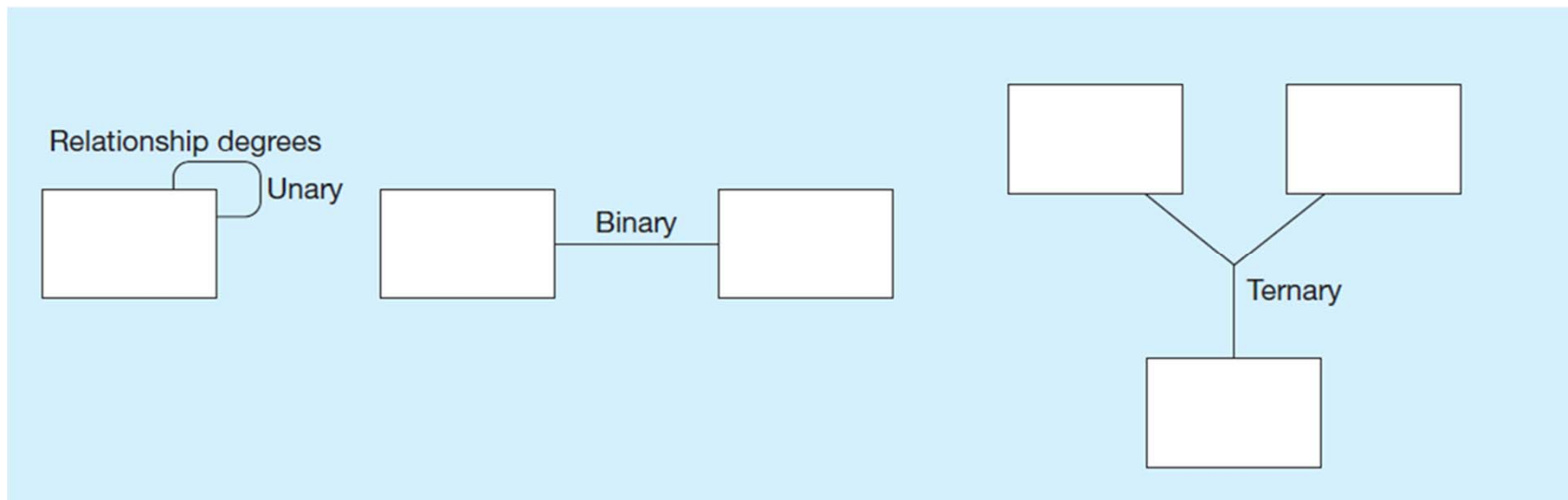


Relationship Degrees

Unary – entities of the same entity type related to each other

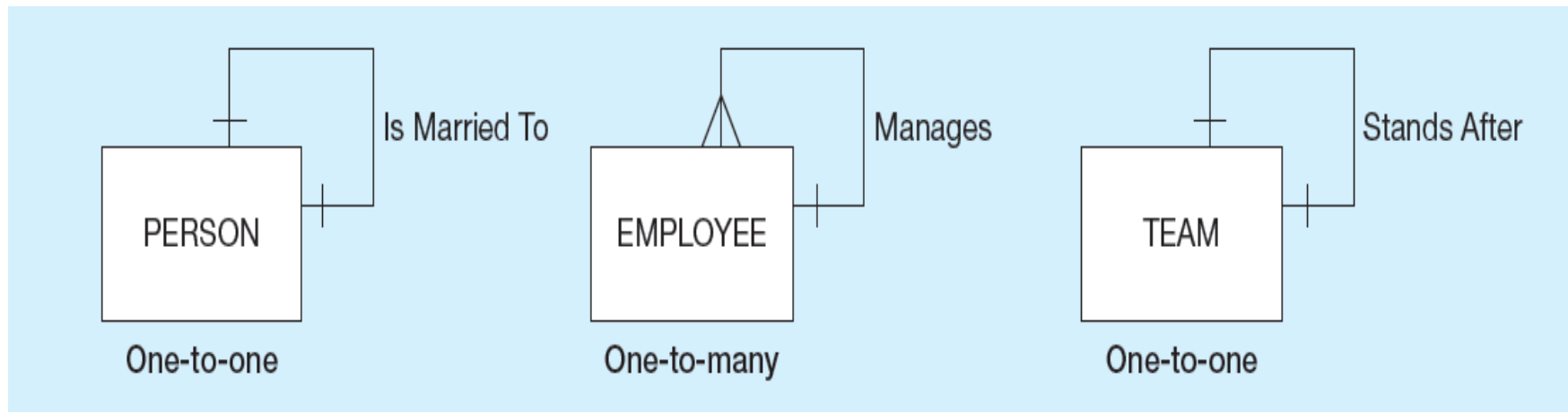
Binary – entities of one type related to entities of another

Ternary – entities of three different types involved in the same relationship



Unary Relationships

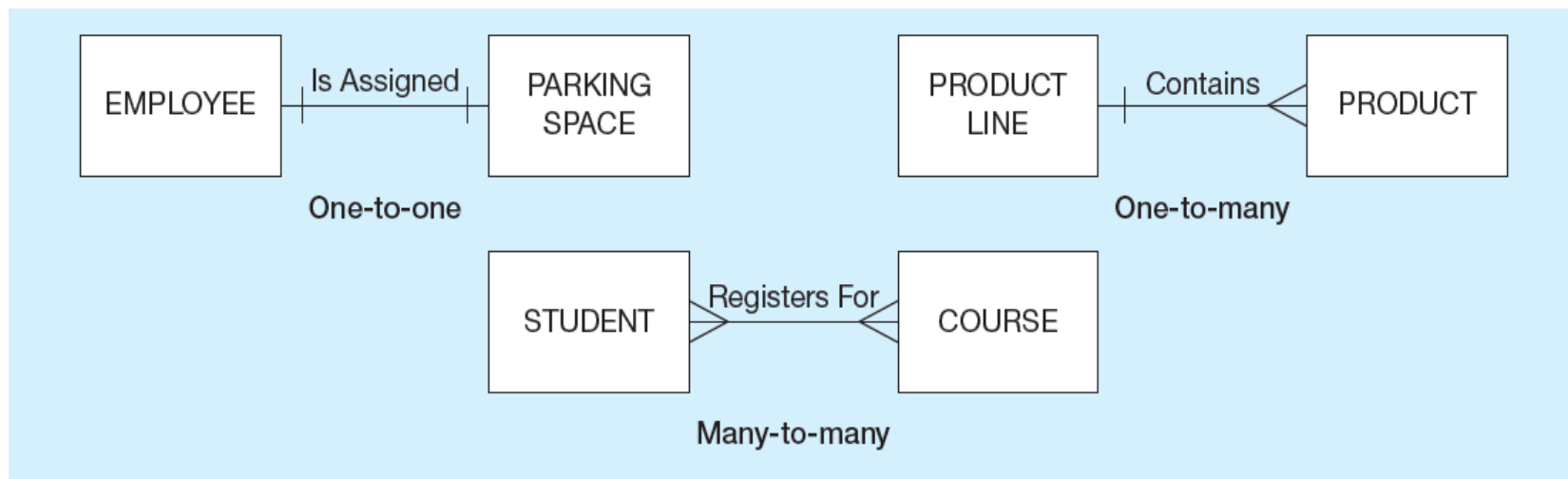
a) Unary relationships



Many-to-many Unary relationship: a Person entity with a “friend” relationship. A particular person can have many friends, and each friend could in turn have other friends.

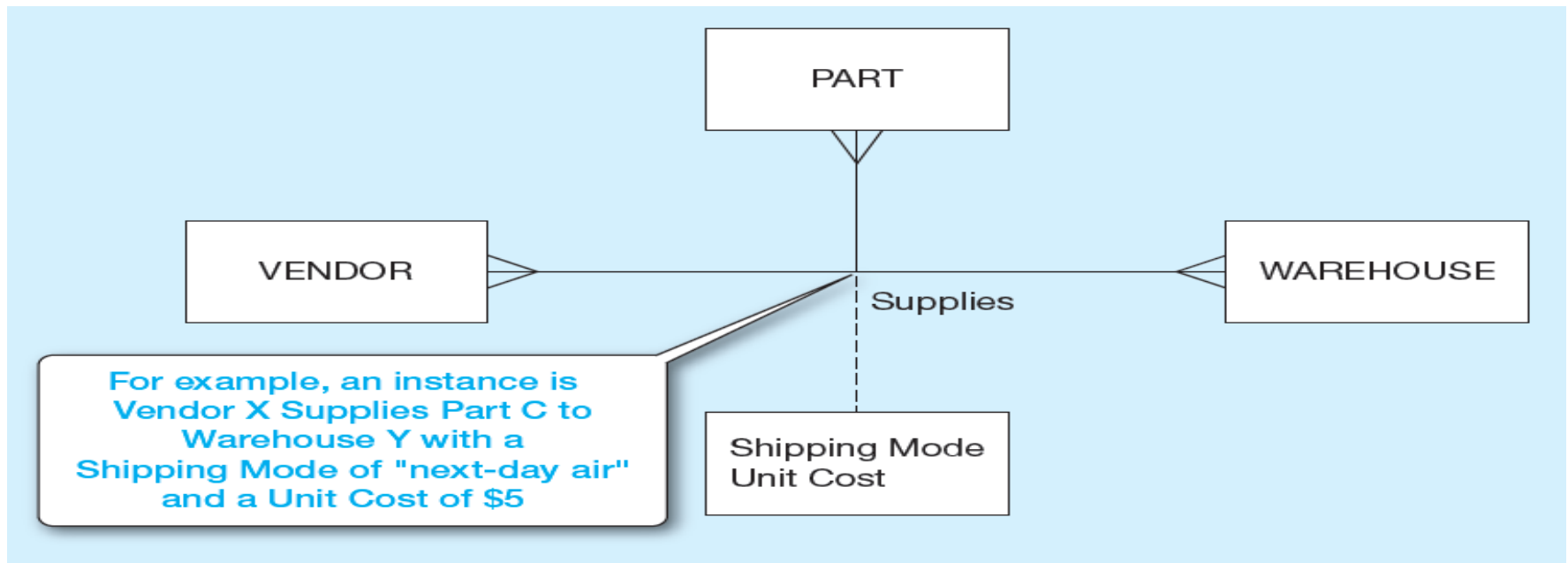
Binary Relationships

b) Binary relationships



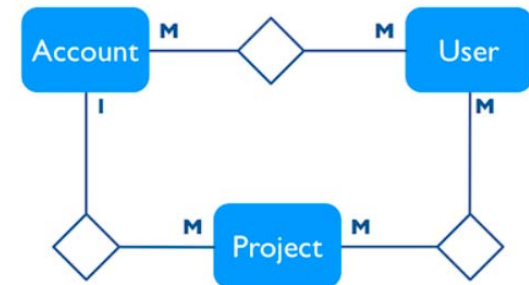
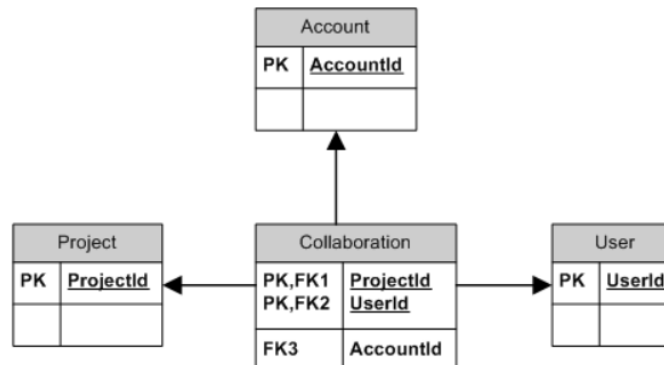
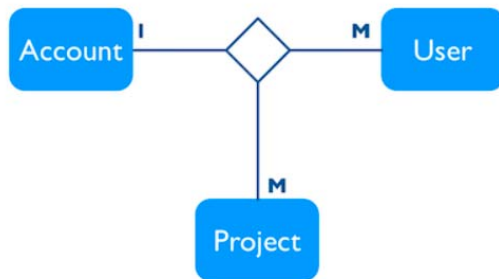
Ternary Relationships

c) Ternary relationships



Ternary Relationships

d) Decomposing a Ternary relationship into binary relationships

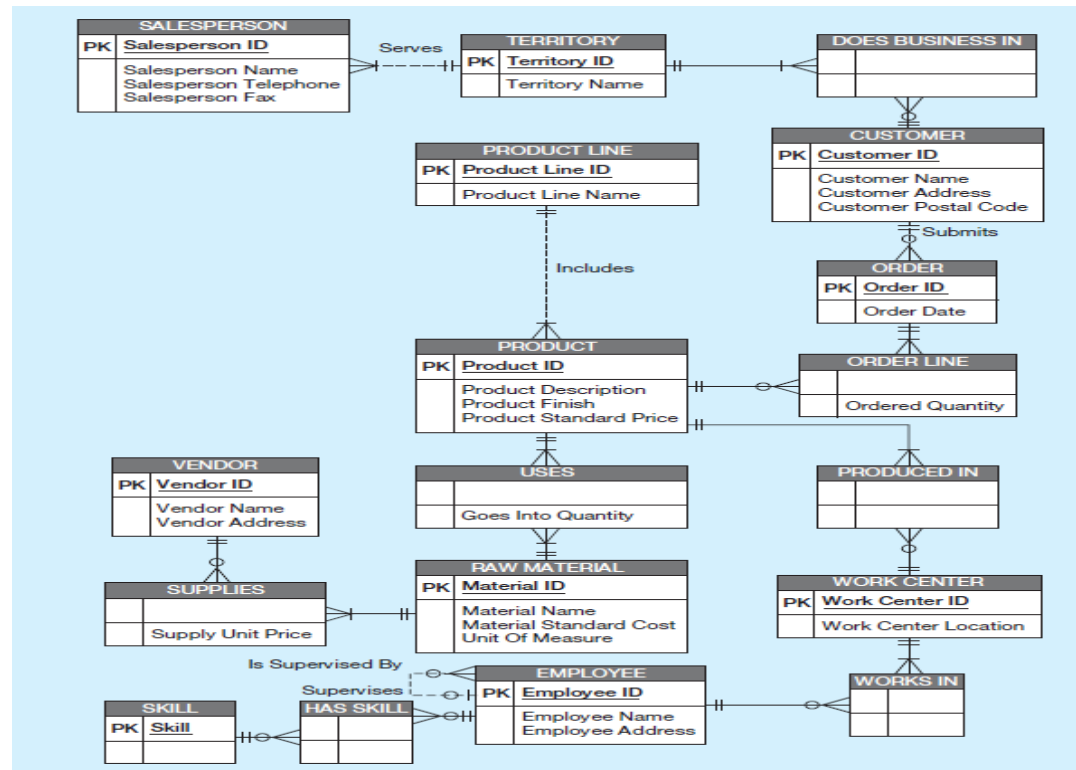


<https://stackoverflow.com/questions/10597698/decomposing-a-ternary-relationship-into-binary-relationships>

Pine View Valley Data Model

data models can be quite comprehensive, including many different entities and relationships.

Different modeling software tools may have different notation for the same constructs.



Limitations of the ER model

- ER model presents a temporary snapshot and cannot model temporal constraints
 - Examples: a project needs to be assigned to a department after one month, an employee cannot return to a department of which he previously was a manager, a purchase order must be assigned to a supplier after two weeks, etc.
- ER model cannot guarantee the consistency across multiple relationship types
 - Examples: an employee should work in the department that he/she manages, employees should work on projects assigned to departments to which the employees belong, suppliers can only be assigned to purchase orders for products they can supply

Limitations of the ER model

- Domains are not included in the ER model
 - Examples: hours should be positive; prodtype must be red, white or sparkling, supstatus is an integer between 0 and 100
- Functions are not included in the ER model
 - Examples: calculate average number of projects an employee works on; determine which supplier charges the maximum price for a product

ER Model Summary

- Entity types
- Attribute types
- Relationship types
- Weak entity types
- Ternary Relationship types
- Examples of the ER model
- Limitations of the ER model

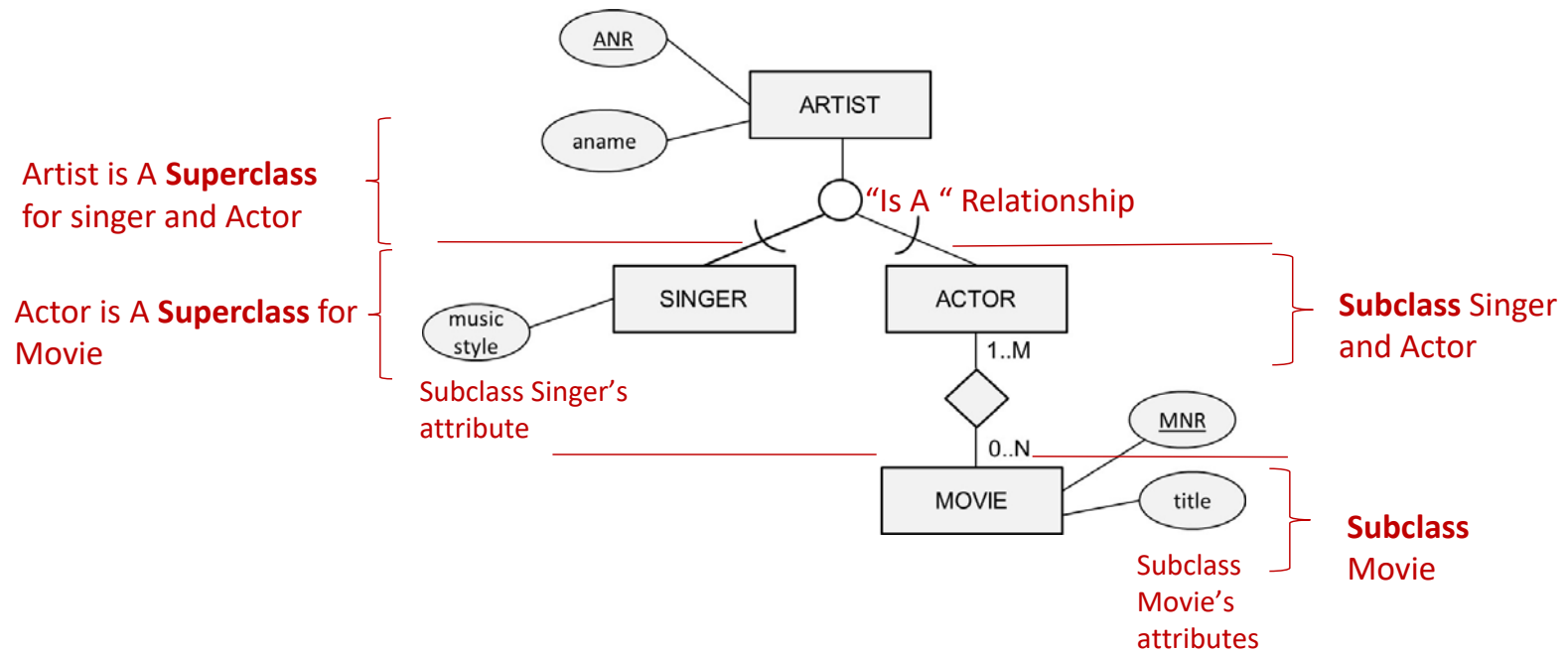
Enhanced Entity Relationship (EER) Model

- Specialization
- Generalization
- Aggregation
- Examples of the EER Model
- Designing the EER Model

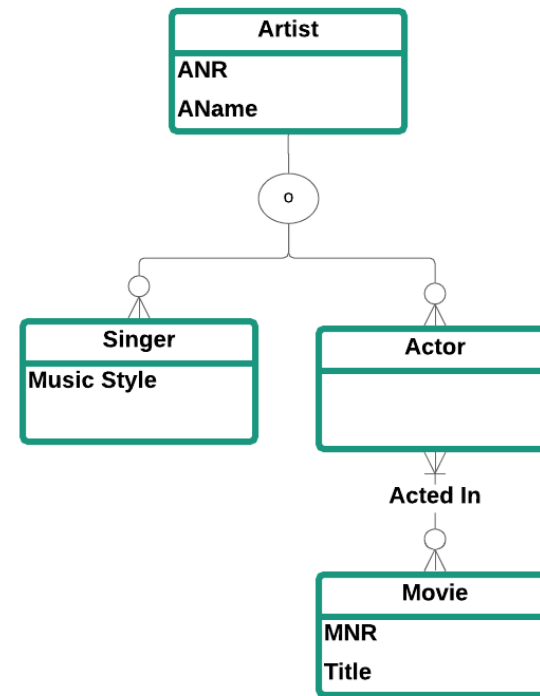
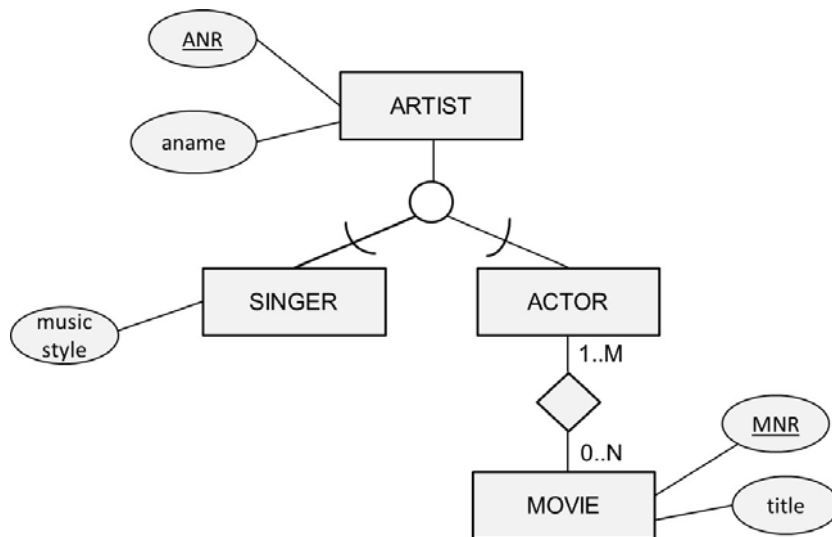
Specialization

- Specialization refers to the process of defining a set of subclasses of an entity type
- The specialization process defines an “IS A” relationship (i.e. A is a subclass of another class B)
- The specialization can then establish additional specific attribute types for each subclass
- The specialization can also establish additional specific relationship types for each subclass
- A subclass inherits all attribute types and relationship types from its superclass

Specialization

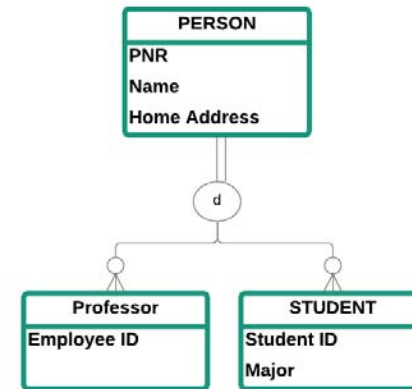
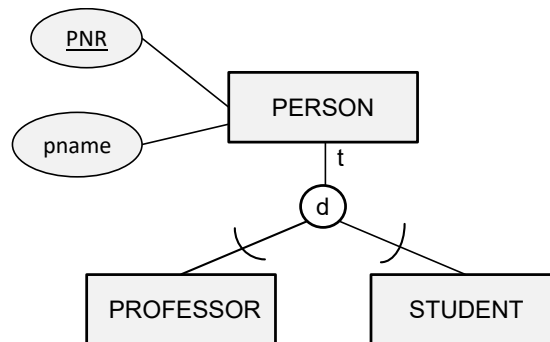


Specialization

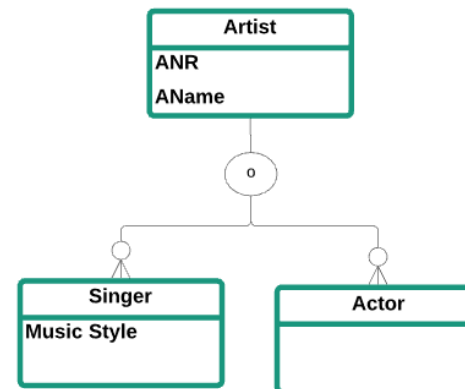
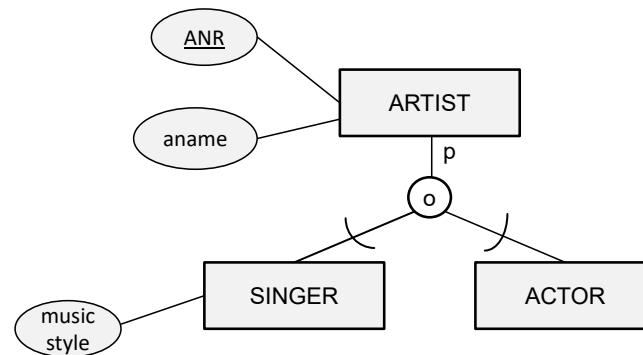


Specialization

Disjoint &
total
specialization



Overlap &
partial
specialization



Generalization

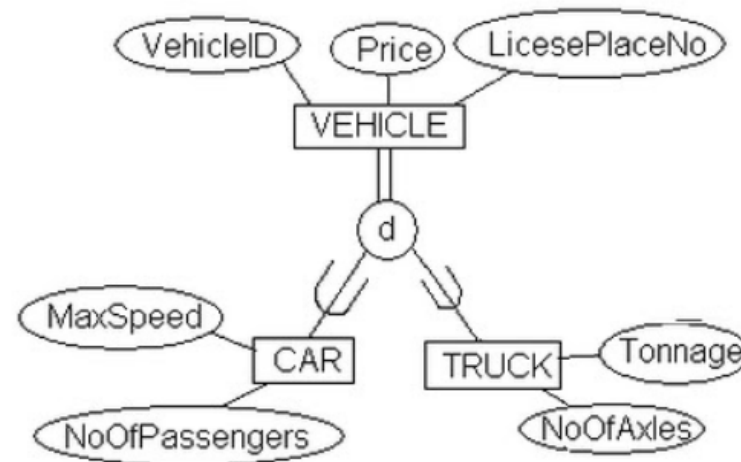
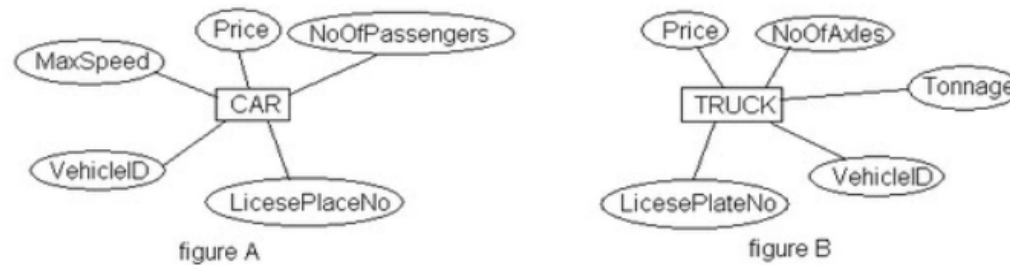
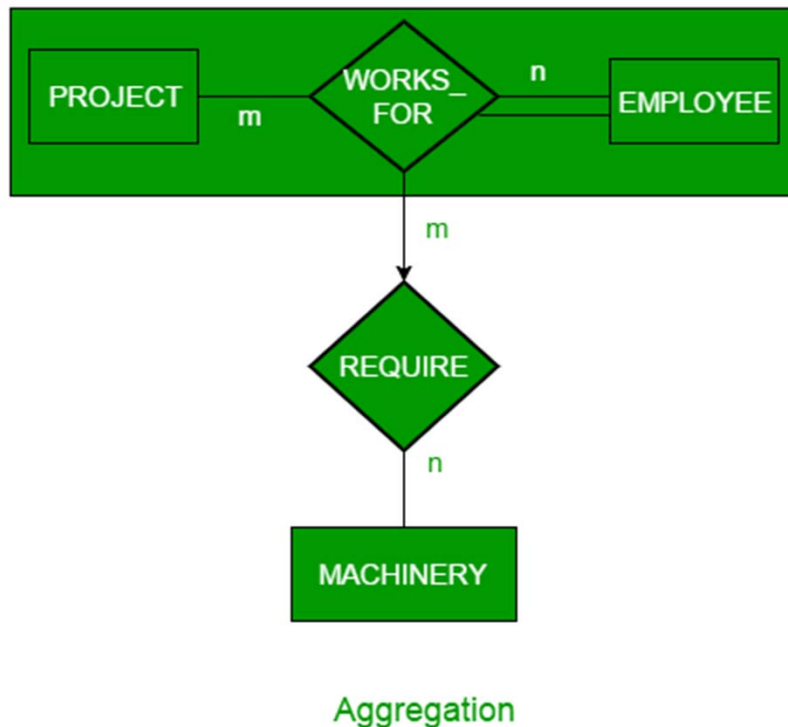


figure C

Aggregation



An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity. Aggregation is an abstraction through which we can represent relationships as higher level entity sets.

E.g. Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.


Designing the EER Model


1. Identify the entity types
2. Identify the relationship types and assert their degree
3. Assert the cardinality ratios and participation constraints (total versus partial participation)
4. Identify the attribute types and assert whether they are simple or composite, single or multiple valued, derived or not
5. Link each attribute type to an entity type or a relationship type
6. Denote the key attribute type(s) of each entity type
7. Identify the weak entity types and their partial keys
8. Apply abstractions such as generalization, specialization, aggregation

<https://www.geeksforgeeks.org/generalization-specialization-and-aggregation-in-er-model/>

9. Assert the characteristics of each abstraction such as disjoint or overlapping, total or partial

<https://www.sites.google.com/site/merasemester/dbm/er-model-3>



 **Pearson** Copyright © 2019, 2016, 2013 Pearson Education, Inc. All Rights Reserved

This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.