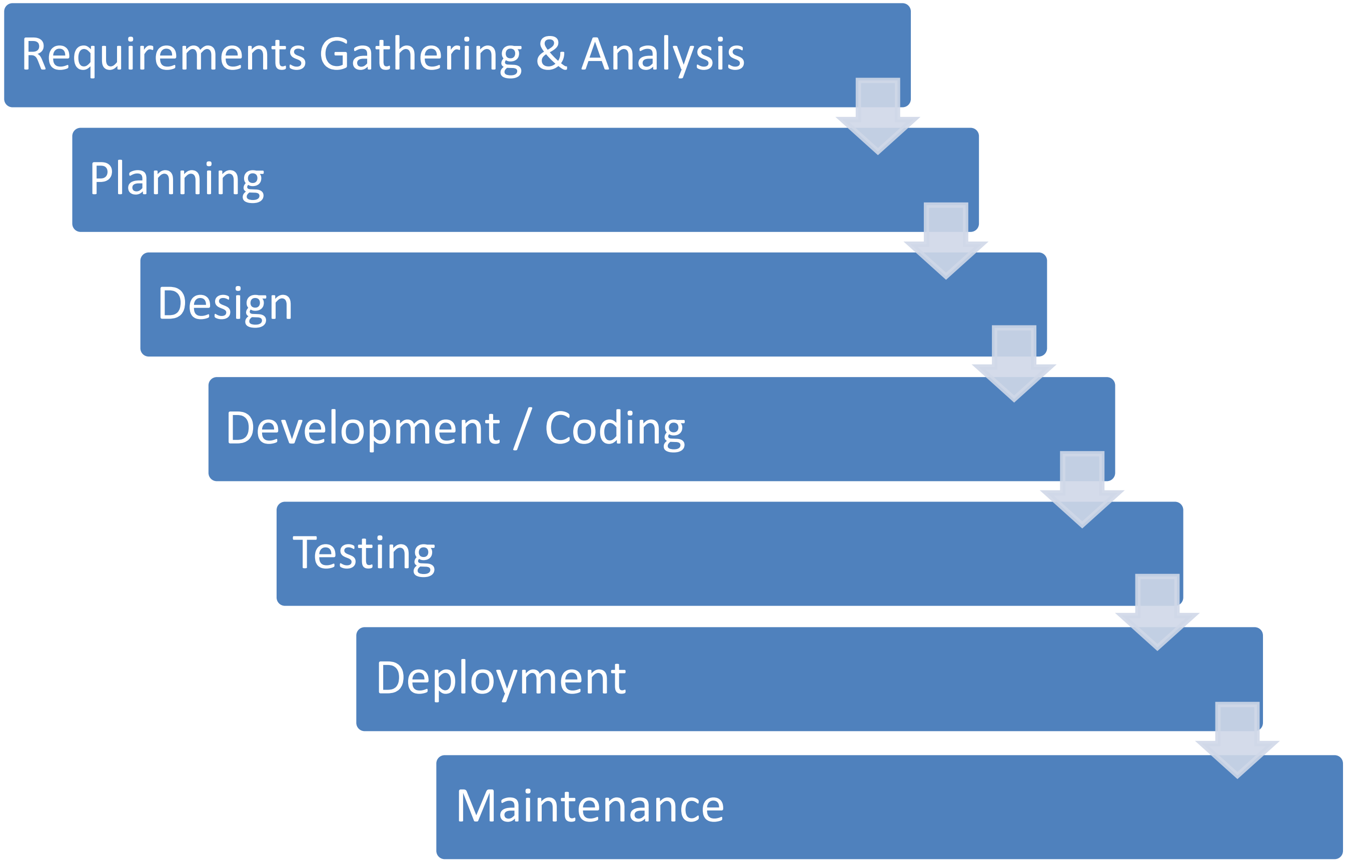
Database Management Systems (CSE 220)

Vikas Bajpai

Acknowledgements

Dr. Daniel Soper, California State University, Fullerton

Software Development Life-Cycle (SDLC)



Requirements Gathering & Analysis

Planning

Design

Development / Coding

Testing

Deployment

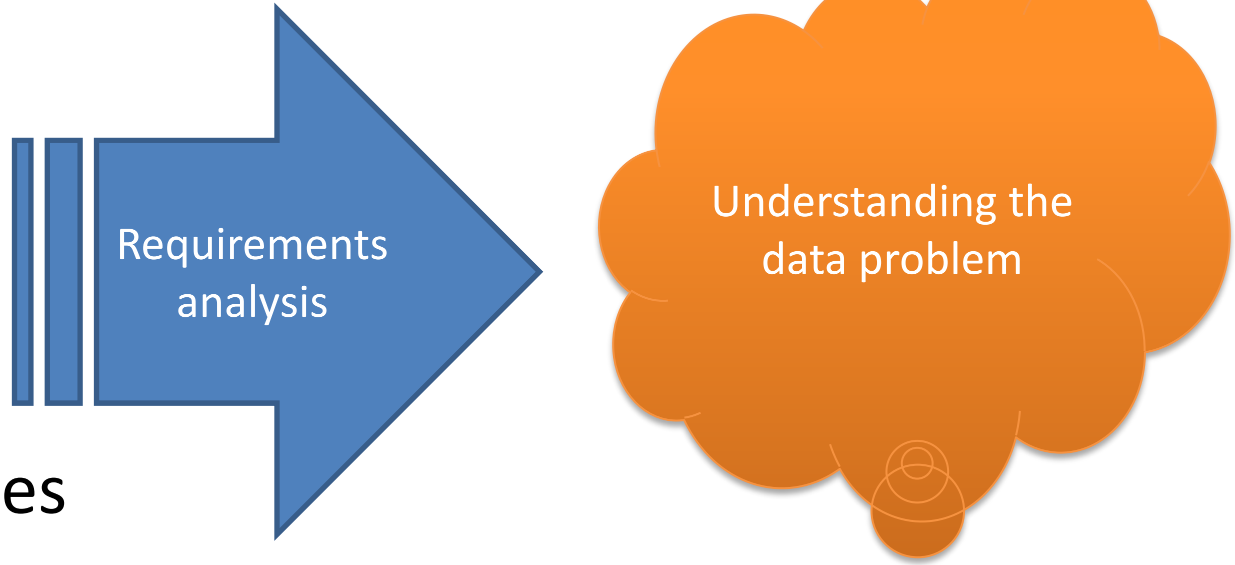
Maintenance

Three stages of Database Development:

1. Requirements Analysis stage
2. Component Design Stage
3. Implementation stage

1. Requirements Analysis stage

- Sources of requirements:
  - User Interviews
  - Forms
  - Reports
  - Queries
  - Use case
  - Business rules
  - Observation
  - JAD sessions



Requirements become the E-R Data Model

- After requirements gathering, these requirements are transformed into an Entity Relationship (E-R) Data Model.
- E-R model is one of the most popular data model.

E-R Model consist of:

1. Entities
2. Attributes
   1. Identifiers (Keys)
   2. Non – key attributes
3. Relationships

3 major graphical components

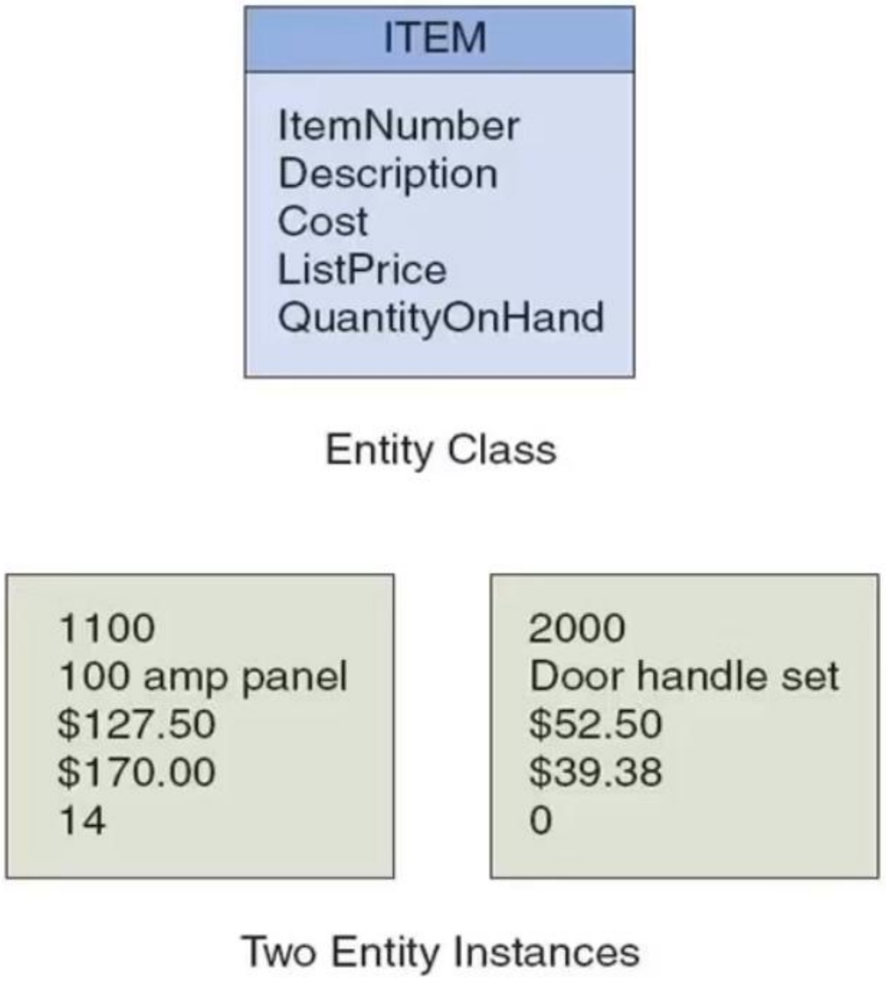
1. Entities

- Entity is a Real-world object distinguishable from other objects.
- An entity is described using a set of attributes.
- Entity Set: A collection of entities of the same kind. E.g., all employees.
  - All entities in an entity set have the same set of attributes.
  - Each entity set has a key

Entity Class vs Entity Instance

- An Entity Class is a description of the structure and format of the occurrences of the entity
  - Similar to a recipe or architectural blueprints
- An entity instance is a specific occurrence of an entity class.

Entity Class vs Entity Instance



ITEM

ItemNumber
Description
Cost
ListPrice
QuantityOnHand

Entity Class

1100
100 amp panel
$127.50
$170.00
14

2000
Door handle set
$52.50
$39.38
0

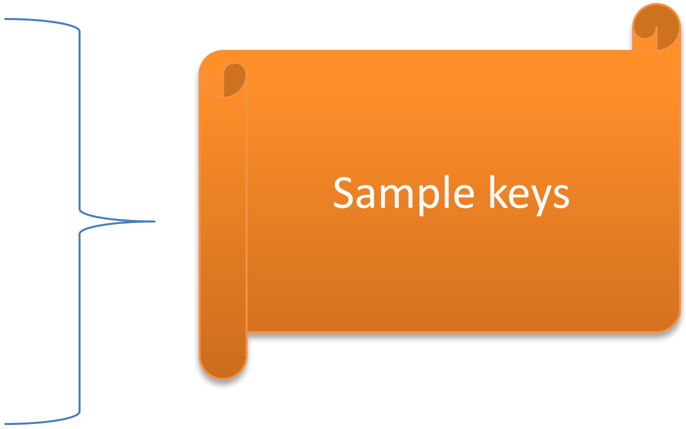
Two Entity Instances

2. Attributes:

- Entities have attributes that together describe the entity
  - Examples: For an Entity Student:
    - Student_Name
    - Student_Reg_Number
    - Student_Branch
    - Student_Join_Date
- Each attribute has a data type and other properties

2.a Identifiers (Keys)

- Entity instances have identifiers (keys)
  - Keys are a type of attribute
- A key will identify a specific instance in the entity class
  - Examples:
    - Student_Reg_Number
    - Student_id
    - Student_emai_id
    - Student_aadhar_card_number



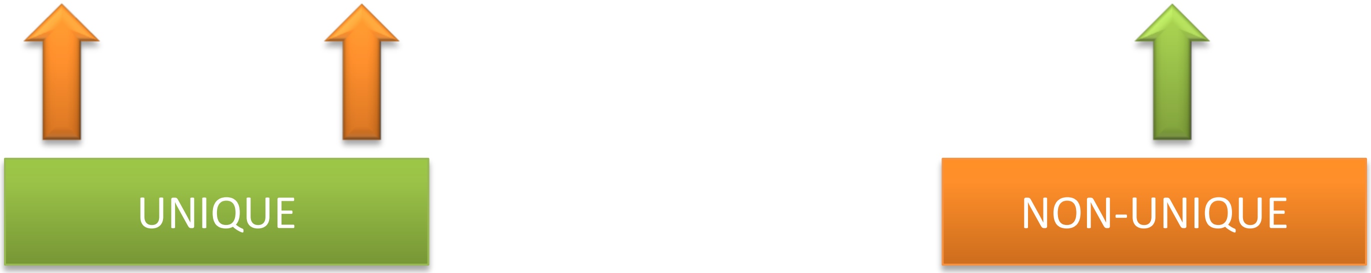
Sample keys

Types of Keys:

- Uniqueness
  - Keys may be UNIQUE or NON-UNIQUE
  - If the key is unique, the data value for the key must be unique among all the instances of the entity
- Composite
  - A composite key consists of two or more attributes
    - Eg: Flight Number and Flight Date
    - Eg: Subject Exam and Exam Date

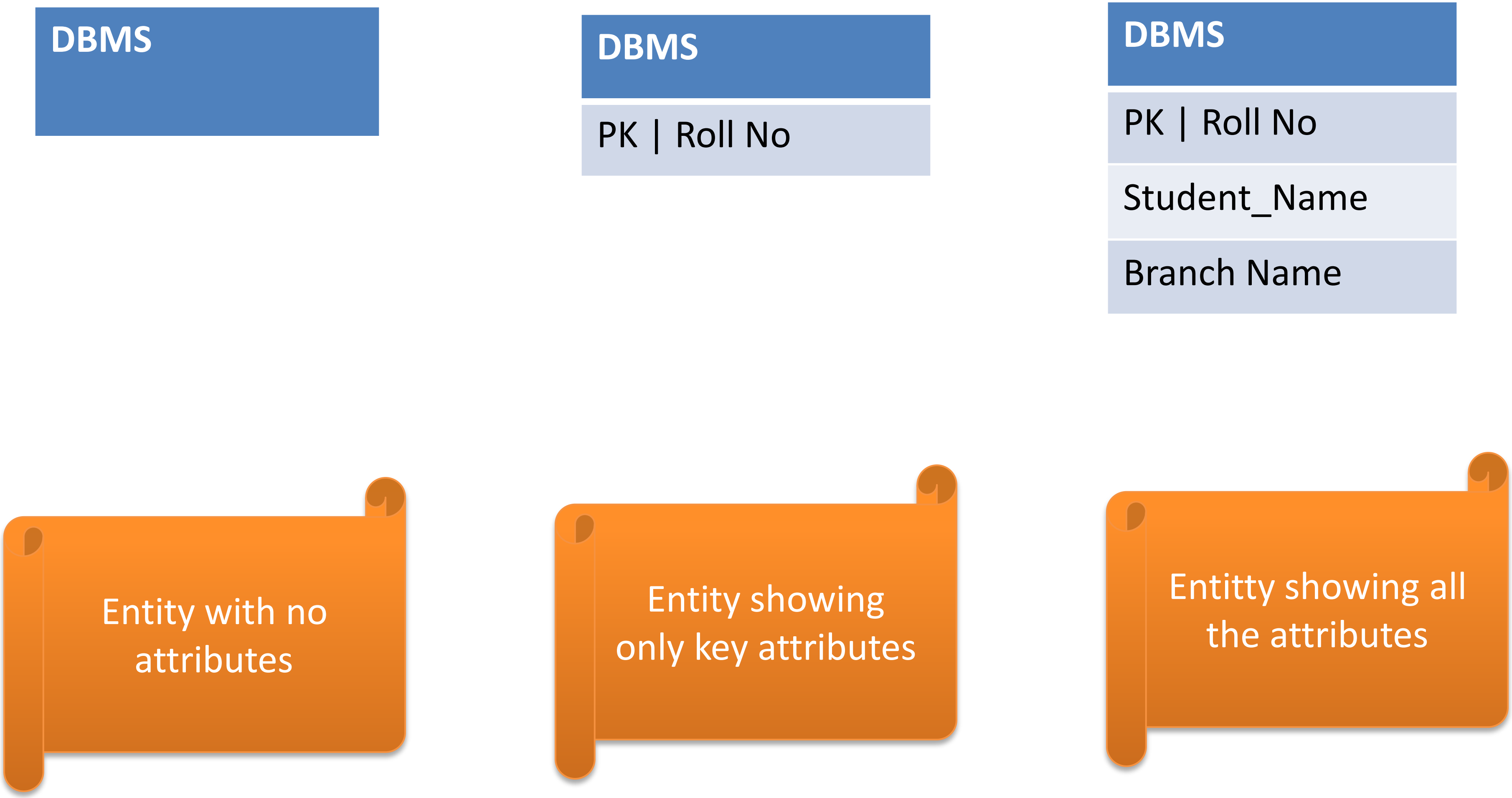
Types of Keys:

S. No.	Roll. No.	Student Name	Branch Name
1	14UCC024	POOJA	CCE
2	14UCS001	AAKARSH SINGH	CSE
3	14UCS002	ABHINANDAN MITTAL	CSE
4	14UCS003	ABHISHREYA DIXIT	CSE
5	14UCS004	ADITI	CCE
6	14UCS005	ADITI ARORA	CCE



*Random, altered data for explanation purpose only

Level of Entity Attribute Display



DBMS

Entity with no attributes

DBMS

PK | Roll No

Entity showing only key attributes

DBMS

PK | Roll No

Student_Name

Branch Name

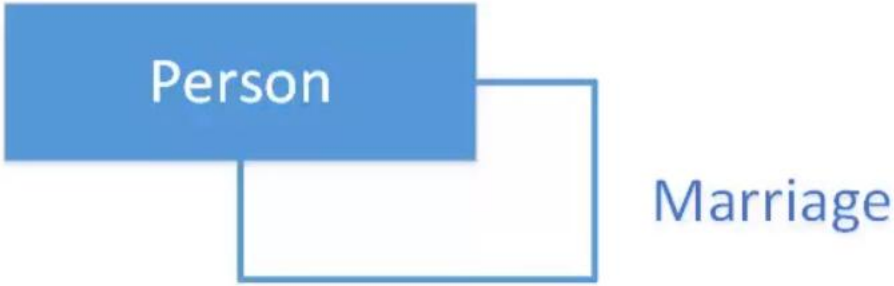
Entitty showing all the attributes

2.b Non – key attributes

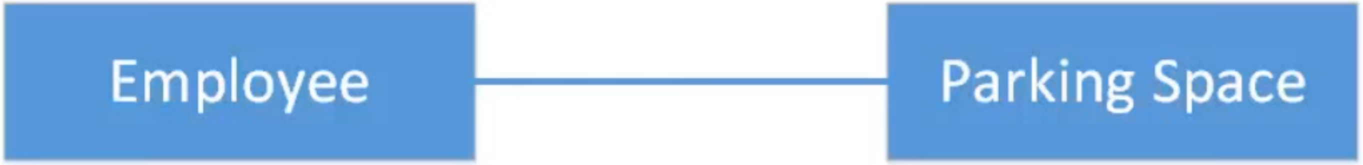
3. Relationships

- Entities can be connected to each other in relationships
- The degree of relationship defines the number of entity classes that participate in the relationship
  - Degree 1 is a unary relationship
  - Degree 2 is a binary relationship
  - Degree 3 is a ternary relationship

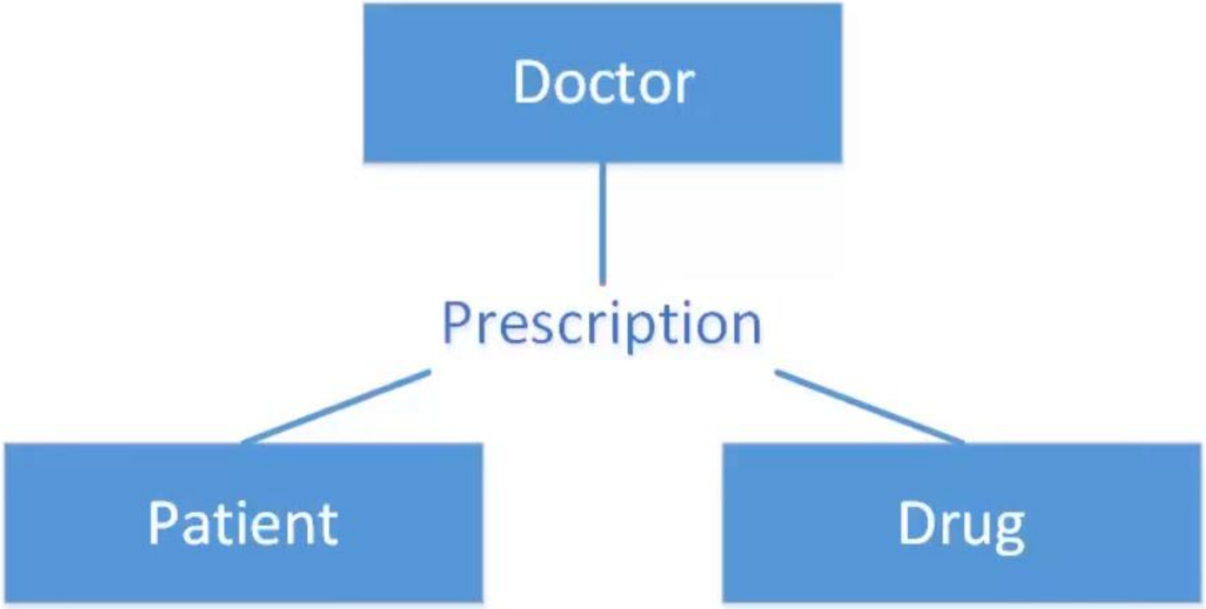
Conceptual Unary Relationship



Conceptual Binary Relationship

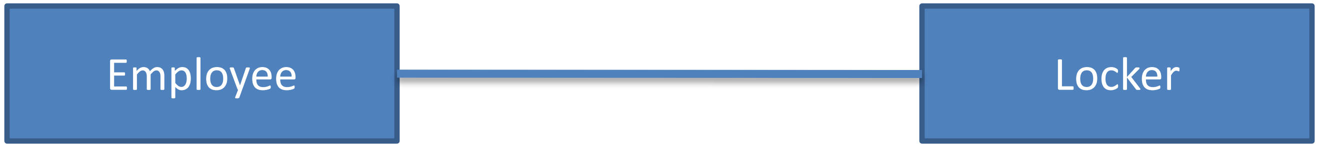


Conceptual Ternary Relationship



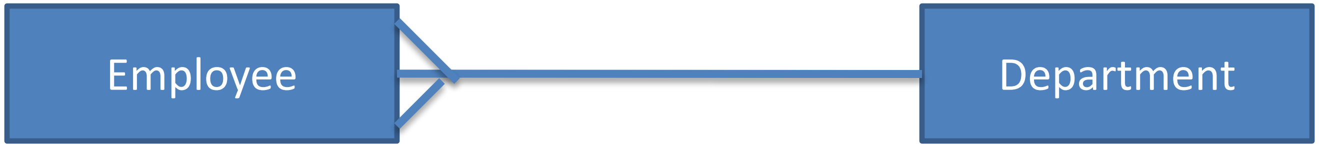
One-to-One Binary Relationship

- 1:! (one-to-one)
  - A single entity instance in one entity class is related to a single entity instance in another entity class
    - An employee may have no more than one locker
    - A locker may only be used by one employee



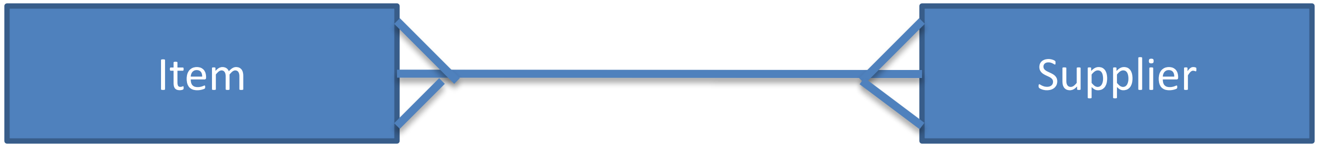
One-to-Many Relationship

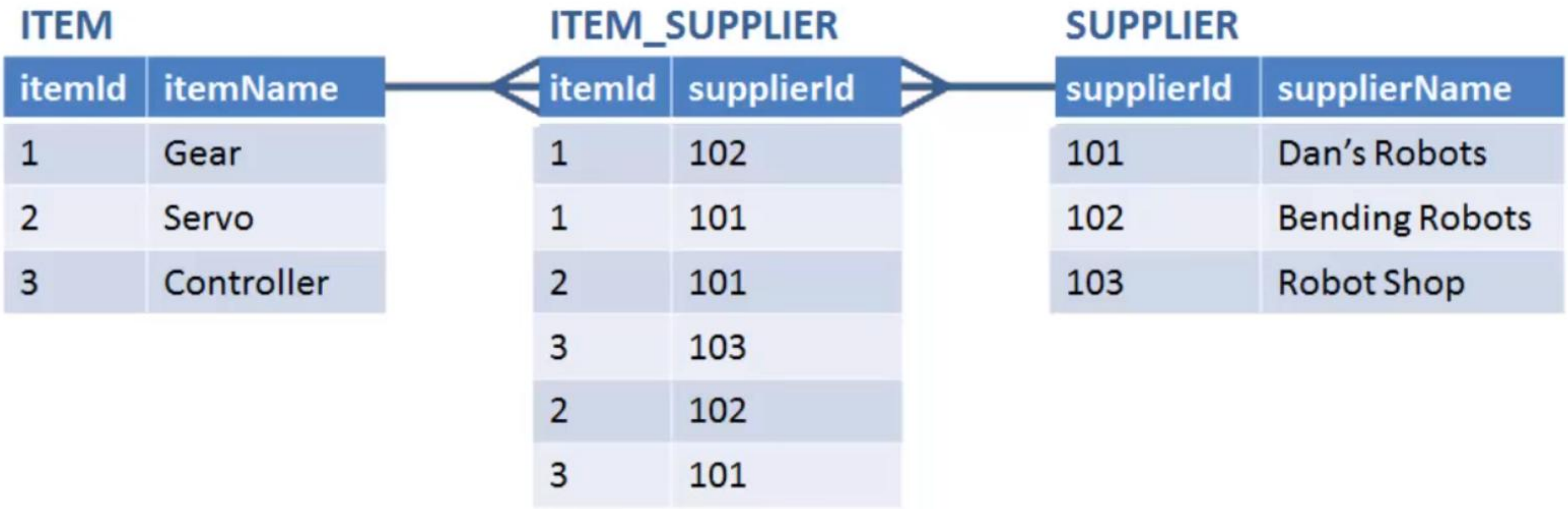
- 1:N (one-to-many)
  - A single entity instance in one entity class is related to many entity instances in another entity class
    - An employee works in one department
    - A department can have many employees



Many-to-Many Relationship

- N:M (many-to-many)
  - Many entity instances in one entity class are related to many entity instances in another entity class
    - A supplier may supply several items
    - A particular item may be supplied by several suppliers





ITEM

itemId	itemName
1	Gear
2	Servo
3	Controller

ITEM_SUPPLIER

itemId	supplierId
1	102
1	101
2	101
3	103
2	102
3	101

SUPPLIER

supplierId	supplierName
101	Dan’s Robots
102	Bending Robots
103	Robot Shop

Cardinality:

- It refers to the uniqueness of data values contained in a column.
- High cardinality means that the column contains a large percentage of totally unique values.
- Low cardinality means that the column contains a lot of “repeats” in its data range.

It is not common, but cardinality also sometimes refers to the relationships between tables. Cardinality between tables can be one-to-one, many-to-one or many-to-many.

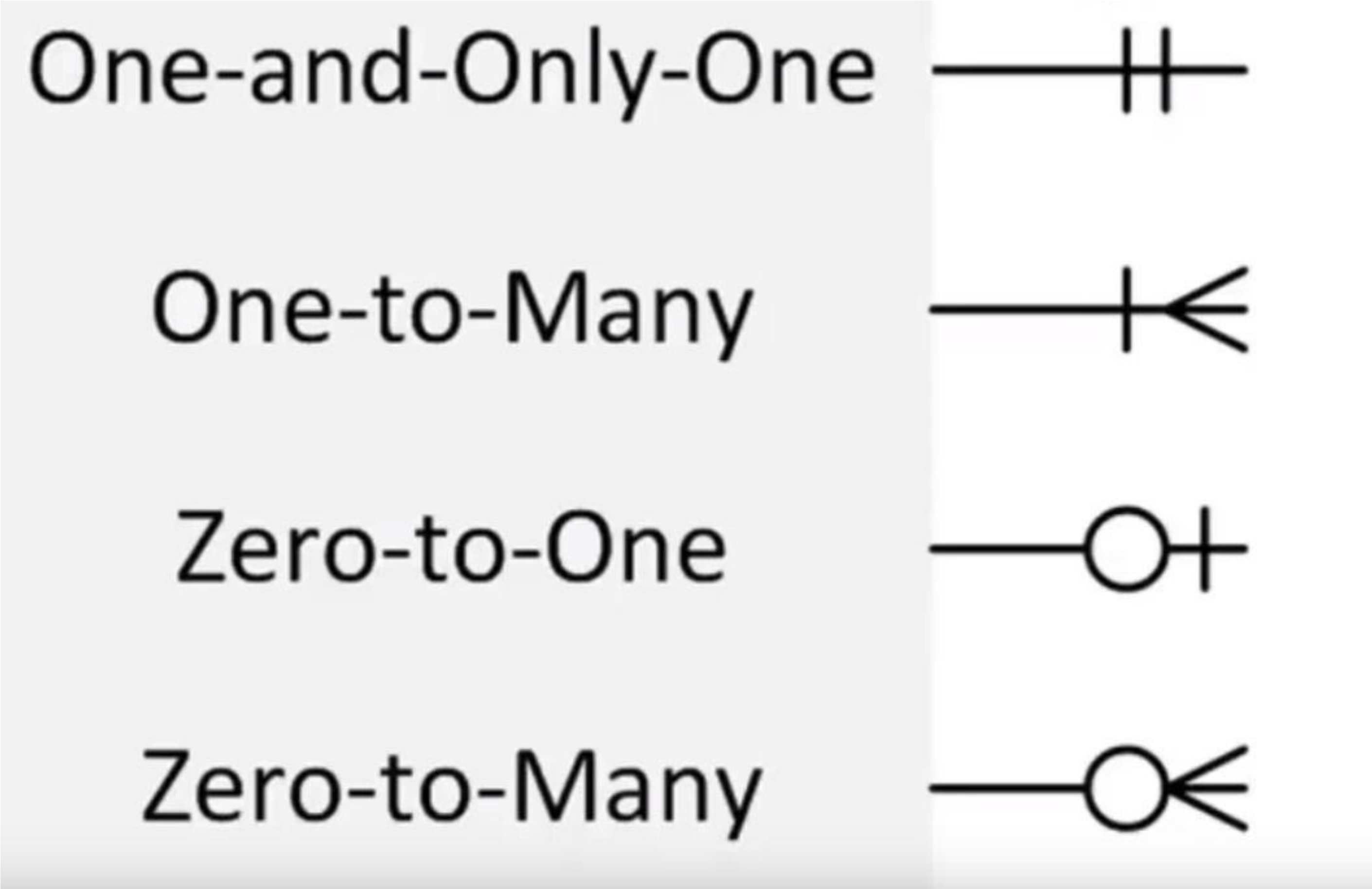
Maximum Cardinality:

- Relationships are named and classified by their cardinalities, which is a word that means count
- Each of three types of relationships shown previously has a different maximum cardinality
- Maximum cardinality is the maximum number of entity instances that can participate in a relationship instance
  - One, many or some other positive fixed number

Minimum Cardinality:

- Minimum cardinality is the minimum number of entity instances that must participate in a relationship instance
- These values typically assume a value of zero (optional) or one (mandatory)

Crow’s Foot Symbols with Cardinalities:



One-and-Only-One

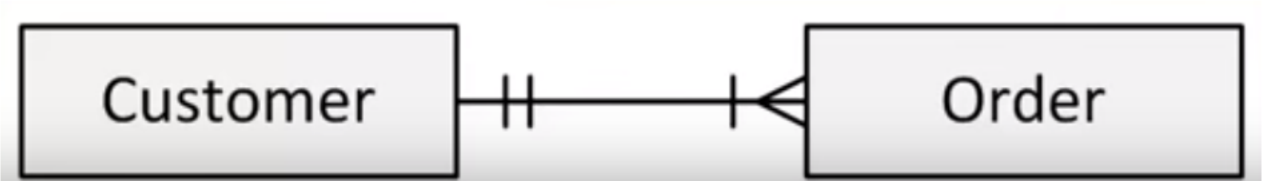
One-to-Many

Zero-to-One

Zero-to-Many

Cardinality Example:

- Maximum cardinality is many for Order and one for Customer
- Minimum cardinality is one for both Customer and Order
  - Each customer can place one or more orders
  - Each order is associated with one and only one customer



Entity-Relationship Diagrams:

- The diagrams in previous slides are called entity-relationship diagrams
  - Entities represented by rectangles
  - Relationships represented by lines
  - Cardinality represented by Crow’s Foot symbols

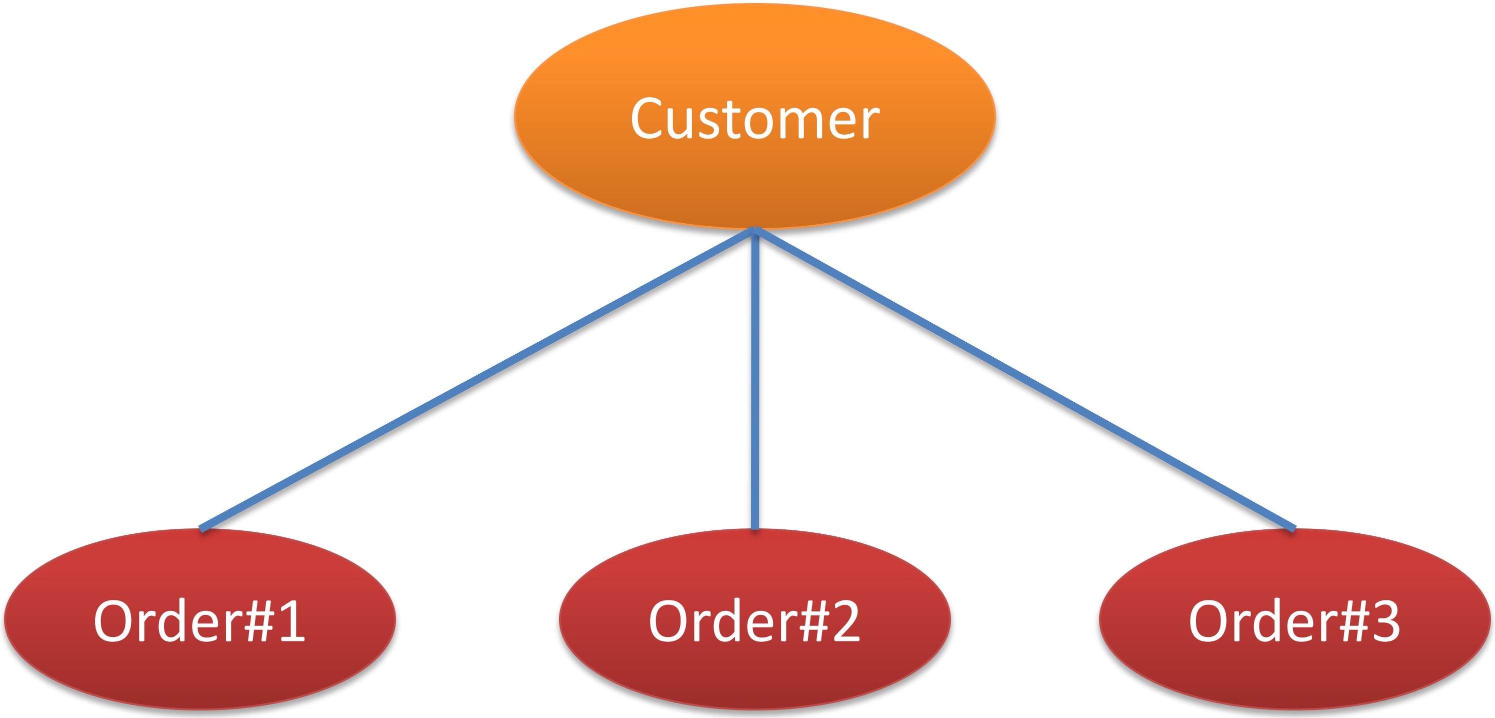
HAS-A relationships:

- The relationships in the previous slides are called HAS-A relationships.
- The term is used because each entity instance has a relationship to a second entity instance
  - An employee has a locker
  - A locker has an employee
- There are also IS-A relationships

Strong and Weak Entities

- A weak entity is an entity whose instances cannot exist in the database without the existence of an instance of another entity
- Any entity that is not a weak entity is called a strong entity
  - Instances of a strong entity can exist in the database independently

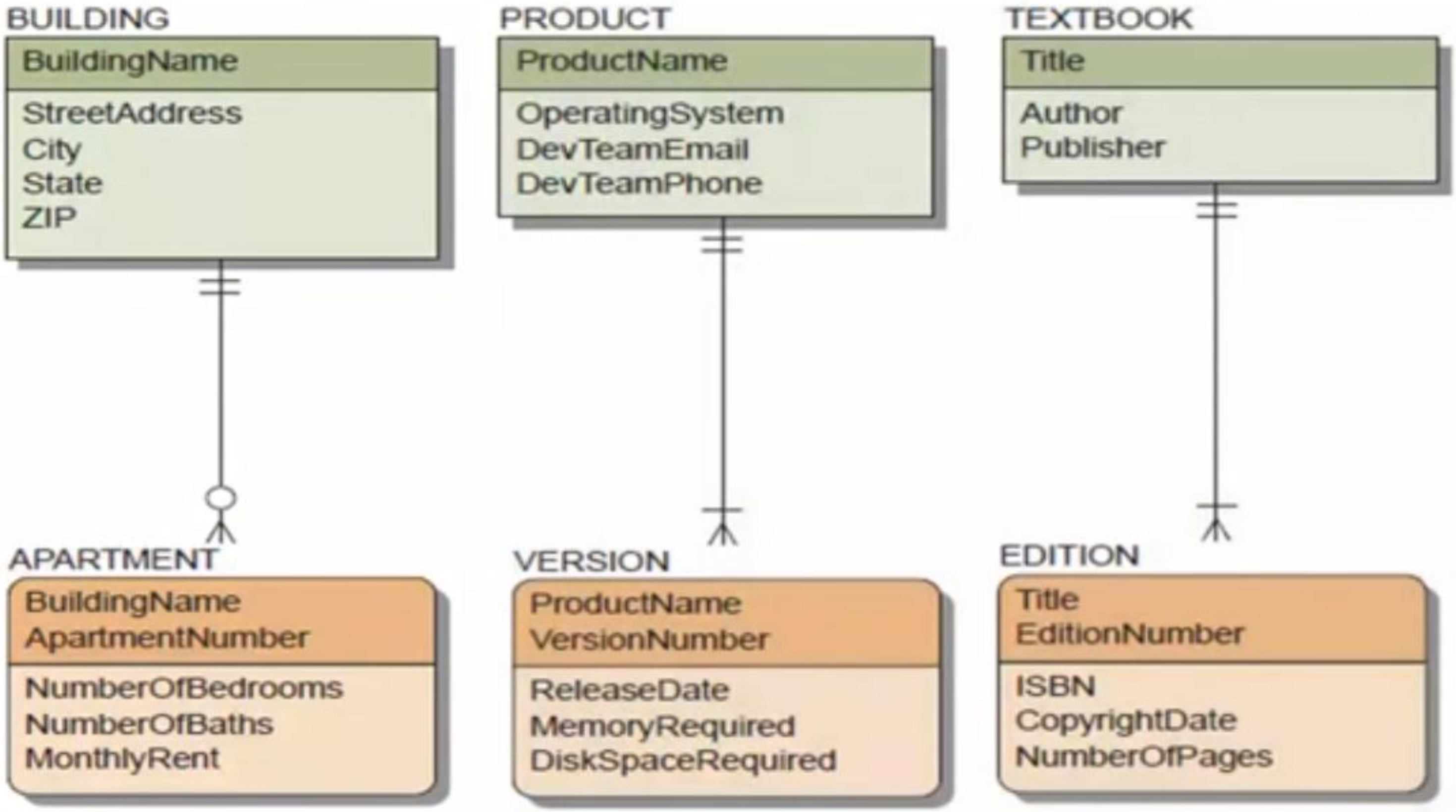
Strong and Weak Entities:



ID-Dependent Weak Entities:

- An ID-Dependent weak entity is a weak entity that cannot exist without its parent entity
- This requirement is enforced by using a composite key for the weak entity
  - The first part of the key is the key for the strong entity
  - The second part of the key is the key for the weak entity itself.

ID-Dependent Weak Entities:



(a) APARTMENT is ID-Dependent on BUILDING

(b) VERSION is ID-Dependent on PRODUCT

(c) EDITION is ID-Dependent on TEXTBOOK

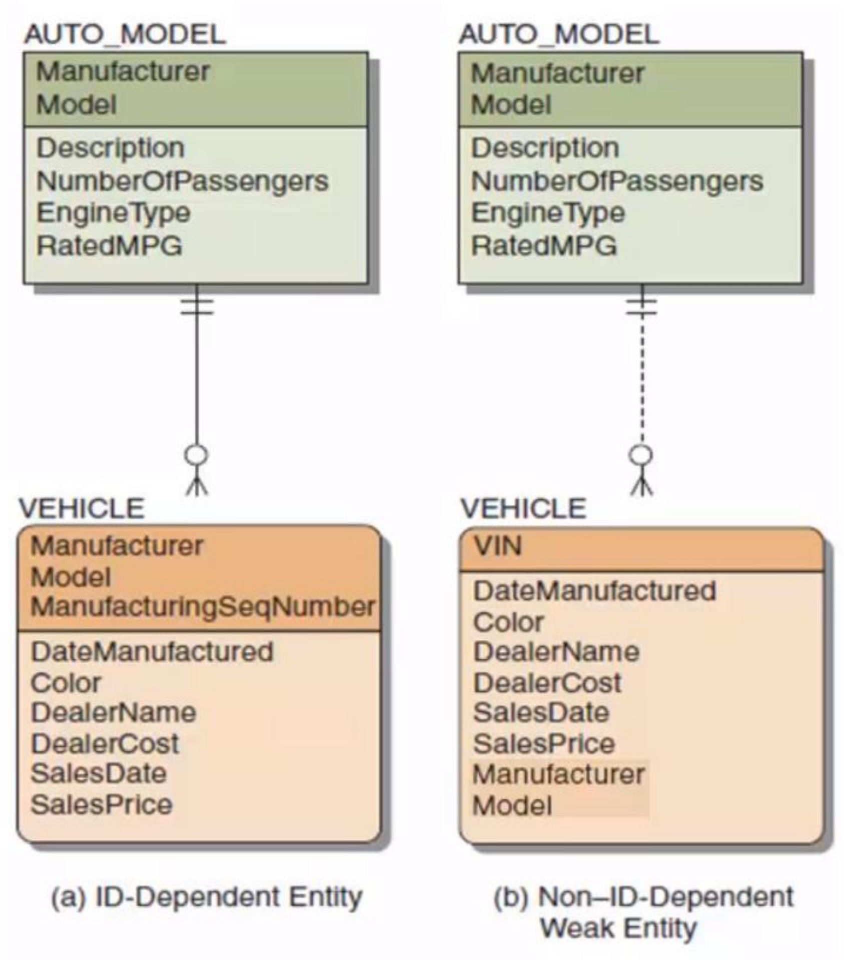
Weak Entity Relationships

- The relationship between a strong and weak entity is termed an identifying relationship if the weak entity is ID-dependent
  - Represented by a solid line
- The relationship between a strong and weak entity is termed a non-identifying relationship if the weak entity is non-ID-dependent
  - Represented by a dashed line
  - Also used between stron entities

Weak entity identifier: Non-ID-dependent

- All ID-dependent entities are weak entities, but there are other entities that are weak but not ID-dependent
- A non-ID-dependent weak entity may have a single or composite key, but the key of the parent entity will be a foreign key within the weak entity.

ID-Dependent vs. Non-ID-Dependent Weak Entities



(a) ID-Dependent Entity

(b) Non–ID-Dependent Weak Entity

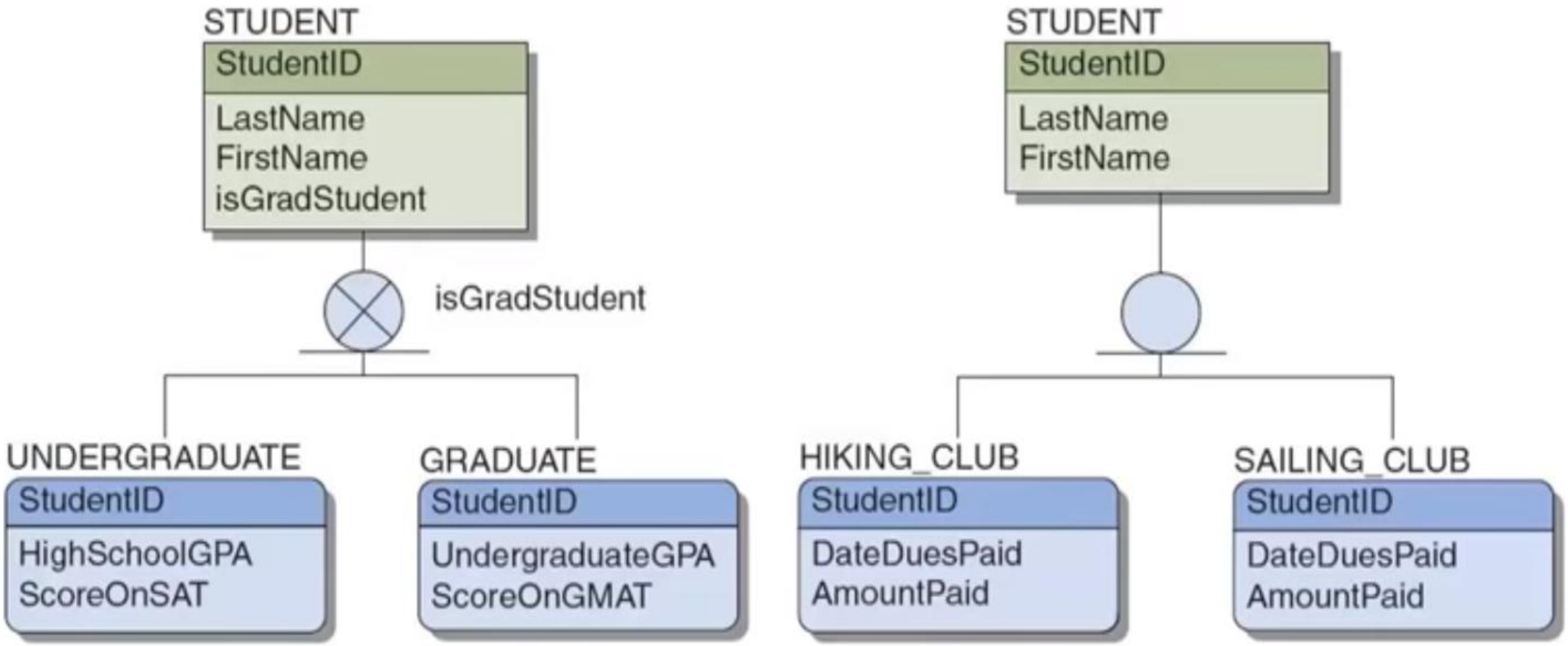
Subtype Entities

- A subtype entity is a special case of another entity (which is called its supertype)
- An attribute of the supertype may be used to indicate which of the subtypes is appropriate for a given instance-This attribute is called a Discriminator
- Subtype can be exclusive or inclusive
  - If excusive, the supertype related to at most one subtype
  - If inclusive, the supertype can relate to one or

Subtype Entity Identifiers

- The identifier of a supertype and all of its subtypes is the same attribute
- The relationships that connect supertypes and subtypes can be IS-A relationsips if a subtype is the same entity as the supertype
  - An instance of the subtype inherits all of the properties of its supertype

Subtype Entity Examples



STUDENT
StudentID
LastName
FirstName
isGradStudent

isGradStudent

UNDERGRADUATE
StudentID
HighSchoolGPA
ScoreOnSAT

GRADUATE
StudentID
UndergraduateGPA
ScoreOnGMAT

(a) Exclusive Subtypes with Discriminator

STUDENT
StudentID
LastName
FirstName

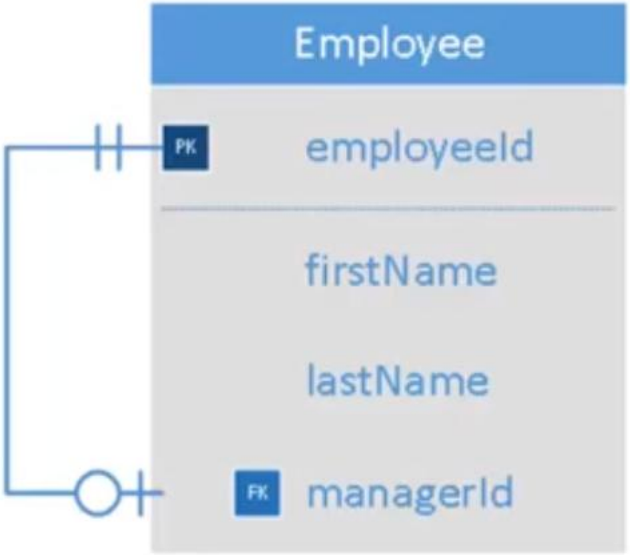
HIKING_CLUB
StudentID
DateDuesPaid
AmountPaid

SAILING_CLUB
StudentID
DateDuesPaid
AmountPaid

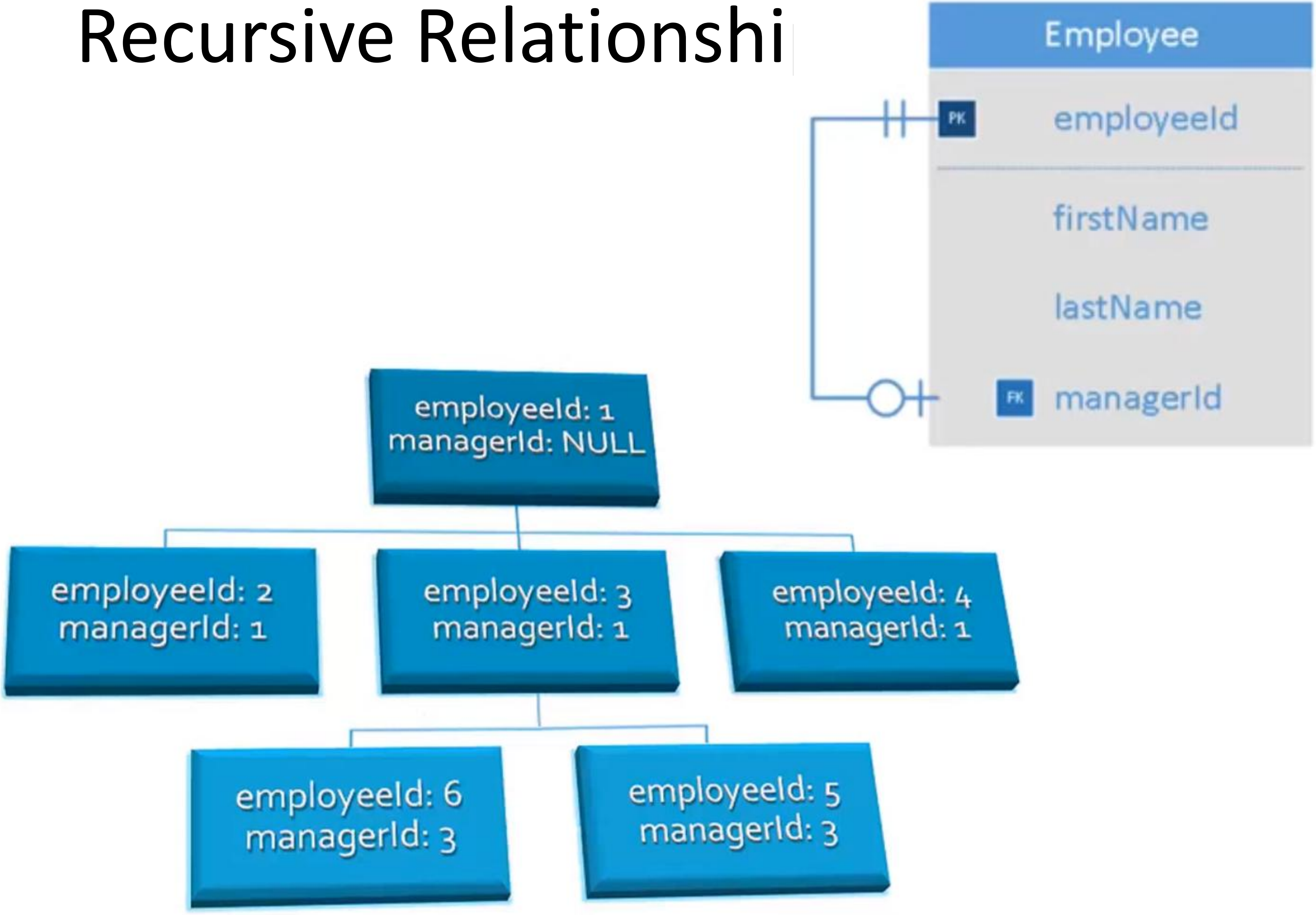
(b) Inclusive Subtypes

Recursive Relationships:

- As noted earlier, it is possible for an entity to have a (unary) relationship to itself-this is called a recursive relationship
- Recursion can be used to implement hierarchical relationships.



Recursive Relationshi


Employee
PK
employeeId
firstName
lastName
FK
managerId
employeeId: 1
managerId: NULL
employeeId: 2
managerId: 1
employeeId: 3
managerId: 1
employeeId: 4
managerId: 1
employeeId: 6
managerId: 3
employeeId: 5
managerId: 3