Database Management System 1305212

Week 04:
Conceptional Design & ER Model

Semester 2/2022

W. Intayoad

School of Information Technology

Mae Fah Luang University

Contents

- Database design:
  - *Conceptional, Logical, Physical database design*
- Designing E-R Diagrams
- Refinement strategy :
  - *Bottom-Up, top-down, Inside out, Mixed*

Database Design

Data Modeling

- A process of documenting the business requirements in an accurate and consistent format
- Entity Relationship Diagram (E-R Diagram) or Unified Modeling Language (UML) can be used for representing data models.
- Data models serve as a blueprint that allows the data architect to translate all the business process to database
- In data modelling there are traditional 3 levels or stages of database model development
    1) *Conceptional Design*
    2) *Logical Design*
    3) *Physical Design*

Database Design

Conceptional Design

- Hight level data design
- Define entities (e.g., customer, employee, order)
- Relationship between entities

Logical Design

- More complicate and more detailed
- Define all the attributes
- Data constrains (PK, FK)
- DB Domain
- Normalization

Physical Design

- Translate data model into physical database (specific DBMS)
- Entities to tables
- Attributes to Columns
- Denormalization

Designing ER Diagrams

Steps for Designing Database

Requirement Analysis: gathering, determining requirements

ER Diagram Modeling to represent conceptual schema / conceptional model

Database Refinement : Top-Down, Bottom-up Strategy

ER Diagram

Entity Relationship Diagram (ERD) (Peter Chen 1971)

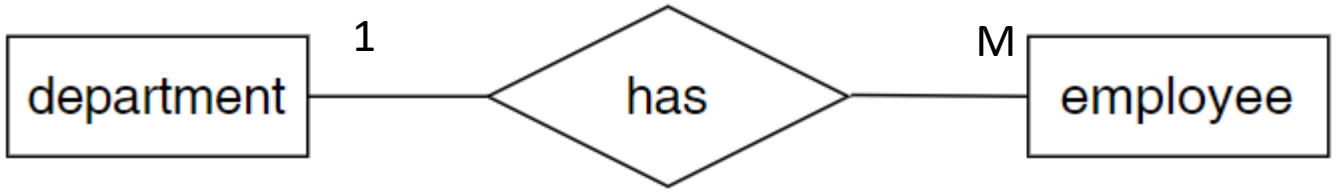
Represents logical structure of database

entities, attributes ,relationships among entities

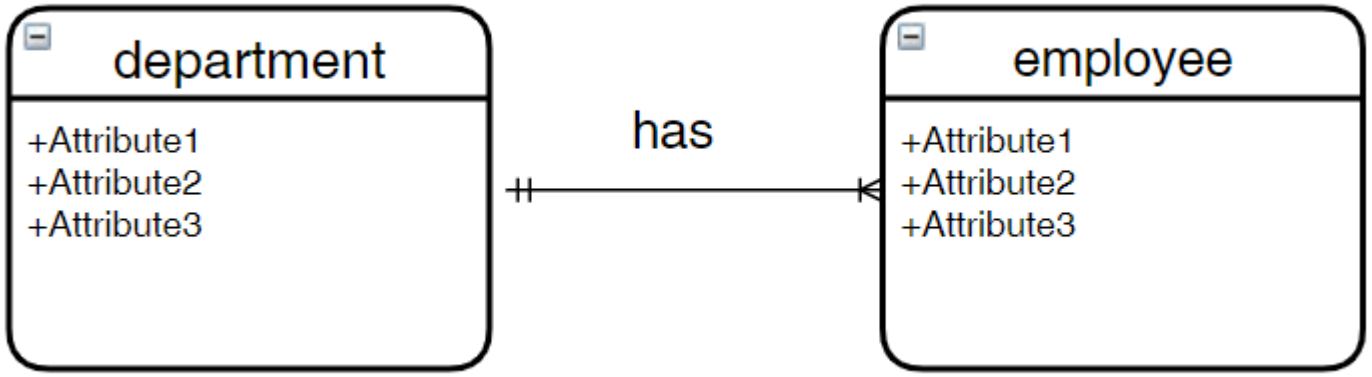
Helps DB designer or architecture analyze data requirements to produce a well-designed database.

Types of ER Diagram

- Chen Notation



- Crow's Foot Notation



- A department has many employees.
- An employee must be in one department.

Designing ER Diagram

***Step** for designing ER diagram*

1. Define Entities
2. Define Entity Relationships
3. Define Attributes
4. Define Database Domain
5. Define Primary Key, Foreign Key

Entity

- Data contain in the DB
- Recommend to use *noun phrase*
- Cannot exist without a *relationship* with another entity
- Example:
  - *Registration system*: student, teacher, course, subject, register
  - *Sale system*: customer, products, order_header, order_detail, employee

Entity Relationship

- Describe general business rules and processes
- Recommend to use *verb phrase*
- Example
    - Students *are supervised* by teachers
    - Teachers *teach* courses
    - Customer can *buy* many products in one order
    - An order *belongs* to one customer
    - Each order *has* responsible sale person

The ***degree*** of a relationship

The number of entity types that participate in the relationship

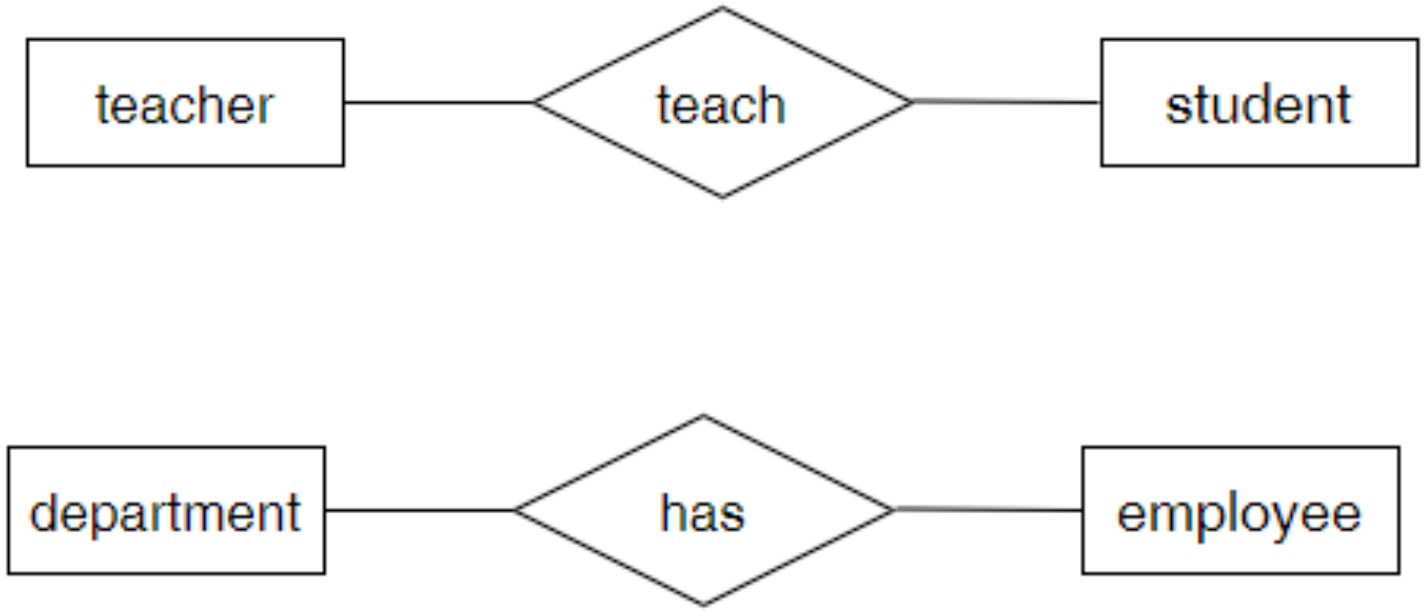
1. Binary: Relationship between 2 entities

2. Unary/Recursive: Relationship with its own entity

3. Ternary: Relationship with 3 entities

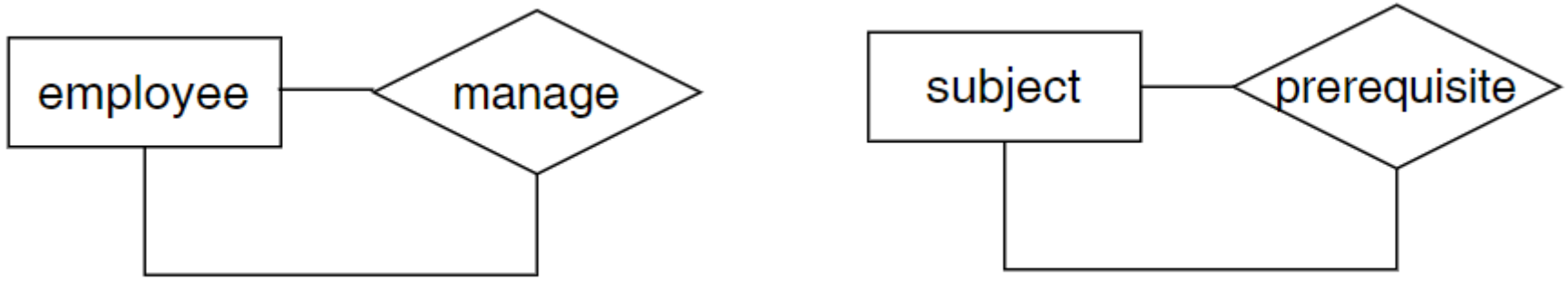
The ***degree*** of a relationship (cont.)

- Binary Relationship



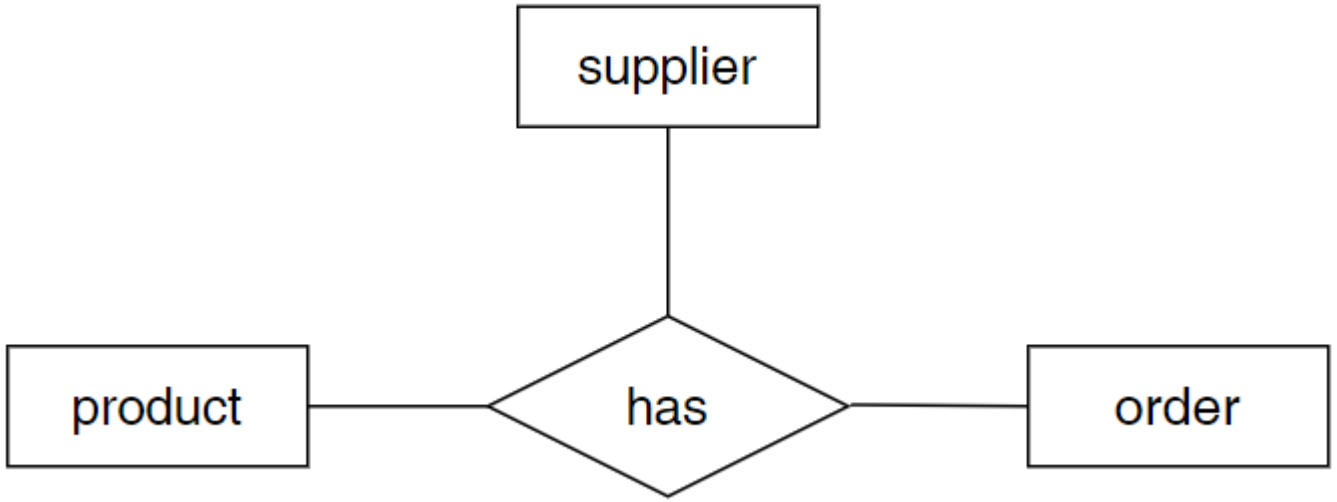
The ***degree*** of a relationship (cont.)

- Unary/Recursive Relationship example:



The ***degree*** of a relationship (cont.)

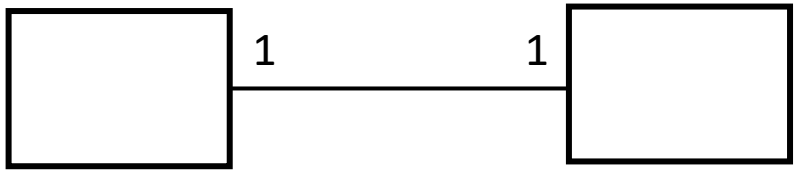
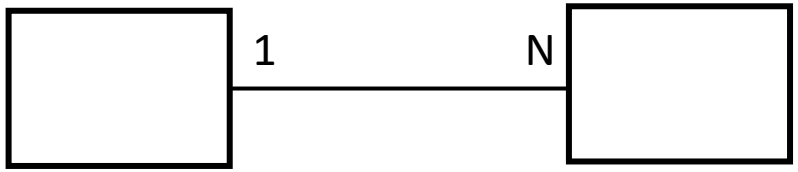
- Ternary Relationship example



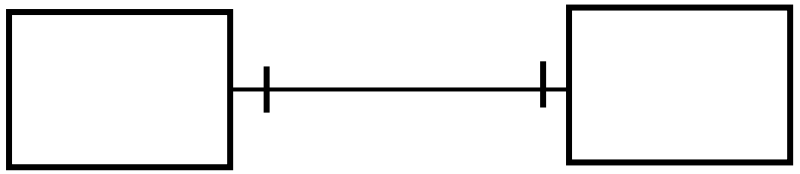
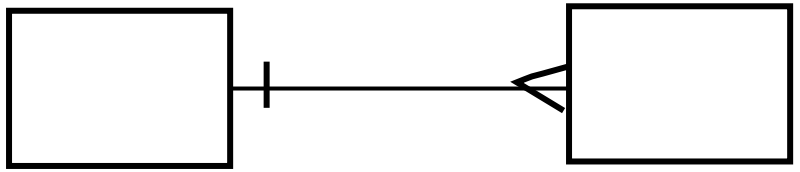
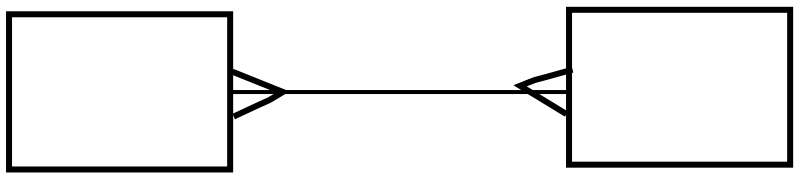
Cardinal Relationships

- The cardinality of a relation between two tables is the number of rows of one table associate another table.
- Cardinality limits are usually derived from the business rules or external constraints.
- The Cardinality shows the types of relationships: 1:1, 1:M or M:1, M:N

Carnality Relationship ***Chen Model***

1:1	One-to-One	
1:N N:1	One-to-Many Many-to-One	
M:N	Many-to-Many	

Carnality Relationship *Crow's Foot Model*

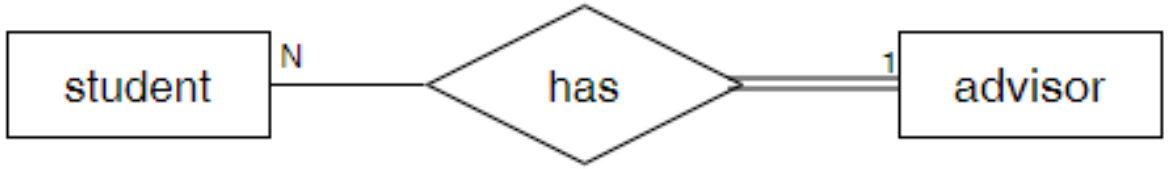
1:1	One-to-One	
1:N N:1	One-to-Many Many-to-One	
M:N	Many-to-Many	

Participation Constraints

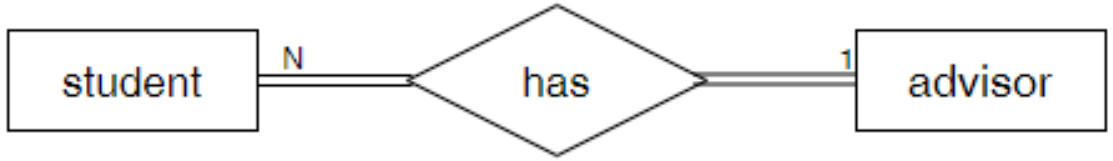
- We also specify the **mandatory** or **optional** relationship in Cardinal Relationships.
- A Mandatory relationship means that there must be at least one matching in each entity.
  - Every order is placed by a customer
  - Every student must have an advisor
- An Optional relationship means that there may or may not be a matching row in each entity.
  - A customer may place many orders
  - A teacher may have many advisee

Example 1

- A Every student must have an advisor, an advisor may have many advisees.



- Every student must have an advisor, an advisor must have at least 1 advisee.



Chen

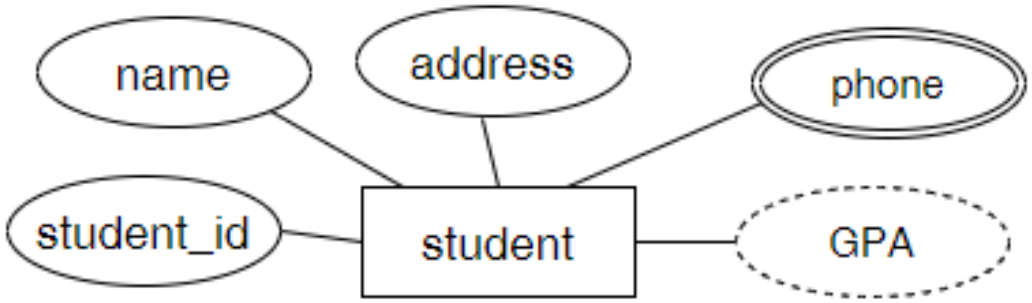
Crow's foot

Attribute

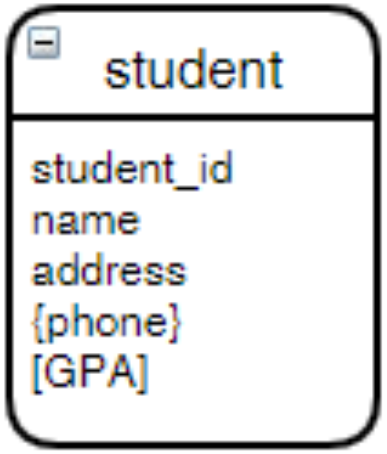
- Add attributes for each entity from user requirements
- Each entity has a set of attributes
- Attribute is a property/ characteristic of an entity
- The numbers of attributes in a table is called “***degree***”.

Attributes

- Student: student_id, name, address, phone, GPA



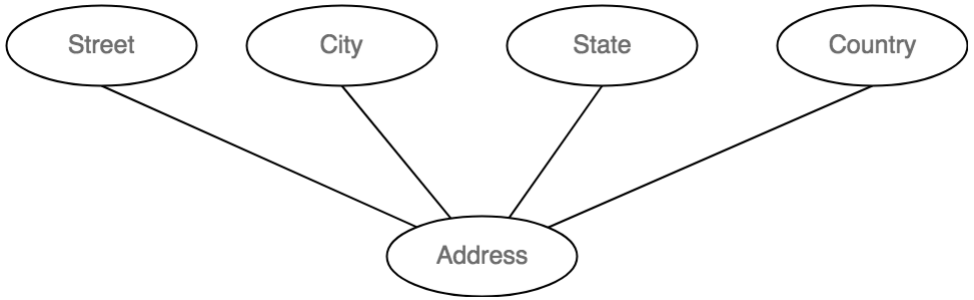
Chen model



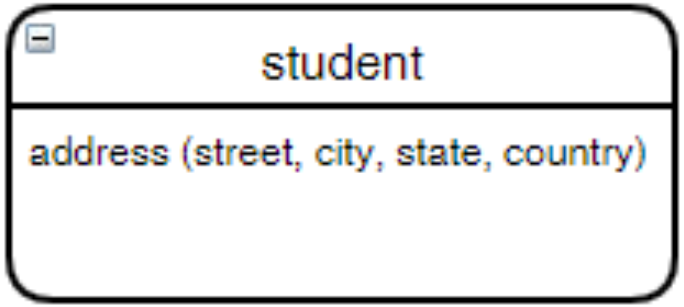
Crow's foot model

Types of Attributes

- Simple attributes
  - Cannot be subdivided into component
  - e.g., student_id, first_name, age
- Composite attributes
  - Can be splitted into component,
  - e.g., address (house_no, street, city, province, zipcode)



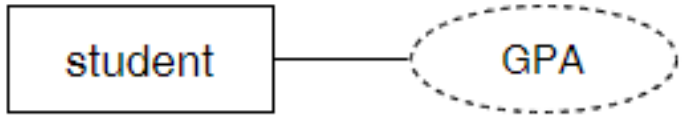
Chen model



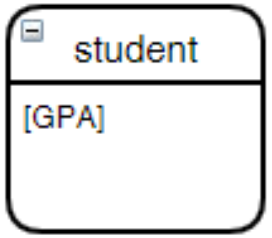
Crow's foot model

Types of Attributes (cont.)

- Derived attributes
  - Can be computed/derived from other attributes or related entities
  - e.g., GPA, total_amount, price_after_tax

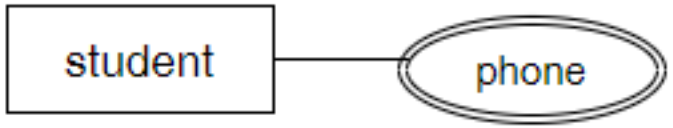


Chen model

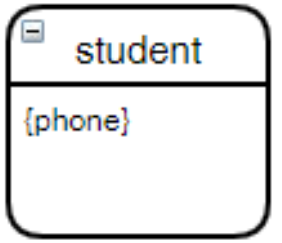


Crow's foot model

- Multivalued attributes
- Store more than one value
- E.g., phone_number a person can have more than phone



Chen model



Crow's foot model

Attribute: *Derived Attributes*

- Storing derived attribute in the database:
- Advantages:
  - Saves CPU processing cycle
  - Saves data access time
  - Data value is readily available
- Disadvantages:
  - Requires constant maintenance to ensure derived value is current

Ref: N. Iam-On, Copyright 2012

Attribute: *Derived Attributes*

- ***Not*** Storing derived attribute in the database:
- Advantages:
  - Saves storage space
  - Computation always yields current value
- Disadvantages:
  - Uses CPU processing cycles
  - Increases data access time
  - Adds coding complexity to queries

Ref: N. Iam-On, Copyright 2012

Attribute Domain

- The set of values allowed in an attribute
- Validation rules for value entered in attribute
- Ensuring that the value entered in each attribute of an entity is consistent
- For example
  - Data type: varchar, number, currency
  - Unique value: student id,
  - Null value: allowed nulled, cannot be null
  - Format: date format, email, telephone
  - Allowed value: F or M for gender

Primary Key (PK)

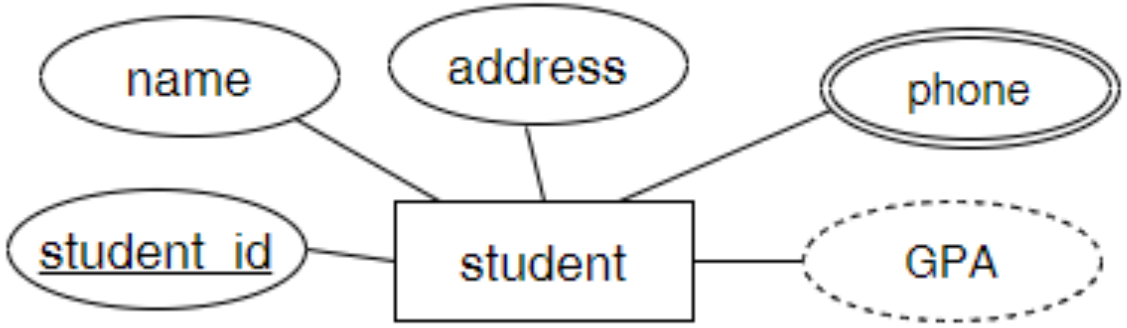
- Only one PK in any relation. (an attribute, or as et of attribute)
- Select from candidate key (unique, not null)
- Use in a table to identify rows.
- The candidate which are not selected to be a primary key is called Alternative key or Secondary key.
- Ex. candidate key: {cus_id}, {SNN}

  Primary key : {cus_id}

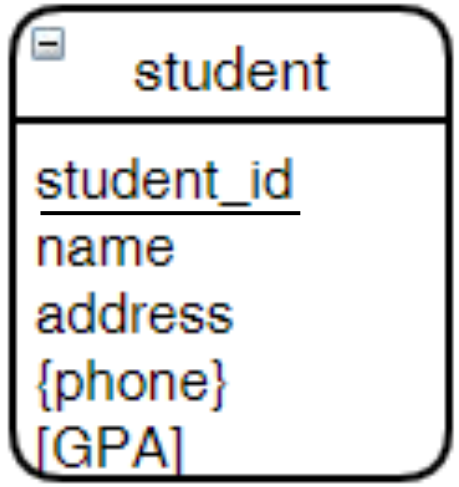
  Alternative key: {SNN}

Primary Key (Cont.)

- Primary keys are *underlined* in ER diagram



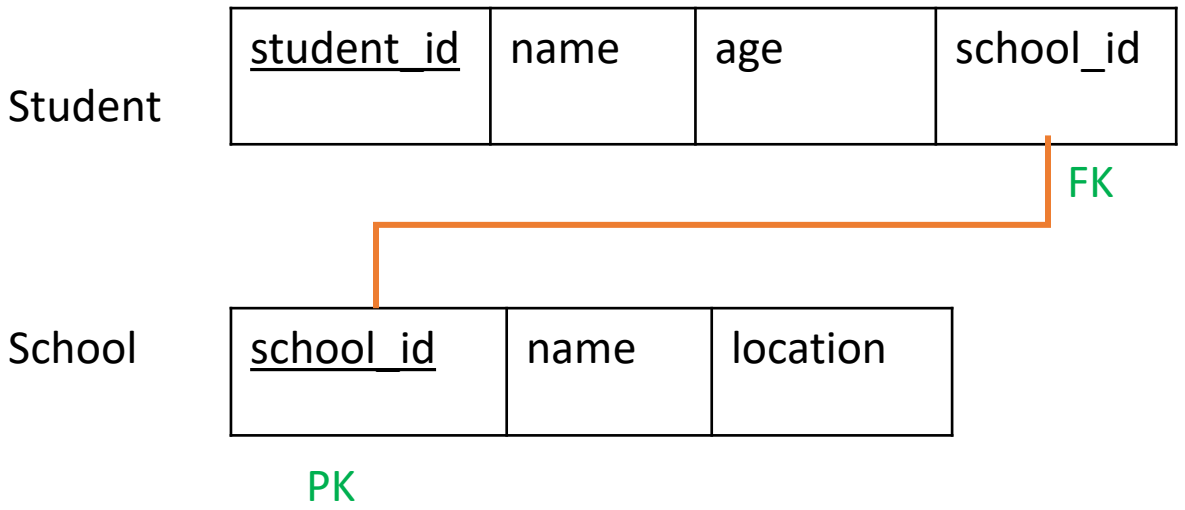
Chen model



Crow's foot model

Foreign Key (FK)

- An attribute or a set of attribute that use to ***link*** to another relation or in the same relation.
- FK is *linked* to the PK in another relation.
- FK must either match to an existing PK or be *null*.



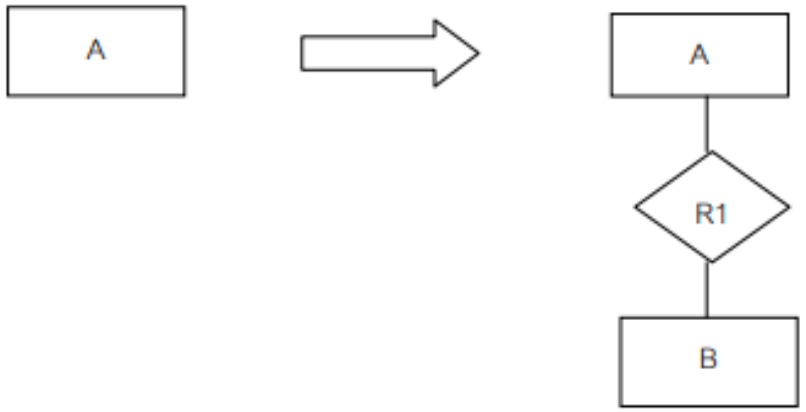
Refinement Strategy

- Designing database is the iterative process
- Starting schema get refinement
- There are 2 approaches for refinement
    - Top-Down strategy: Breaking down of a system to gain insight into its compositional sub-systems in a reverse engineering fashion.
    - Bottom-Up strategy: The individual base elements of the system are first specified in detail and then link these elements together.

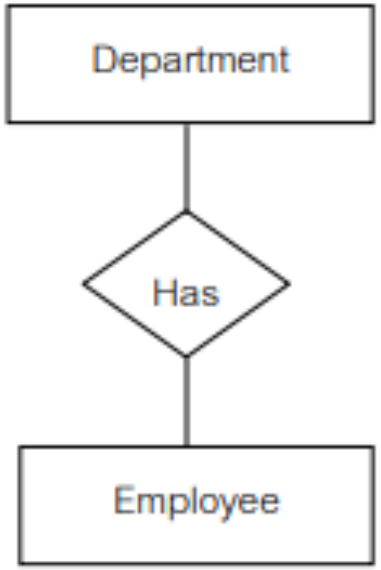
Top-Down Strategy

A. Consider the *relationship* with other entity

- Entity → Related entities



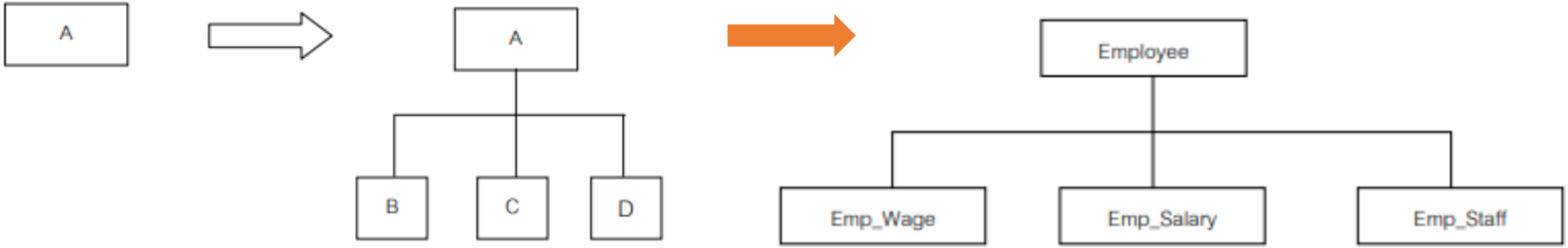
A department has employees



Top-Down Strategy (cont.)

B. Considering decompose an entity to *subclass* (Generalization)

- Entity → subclass

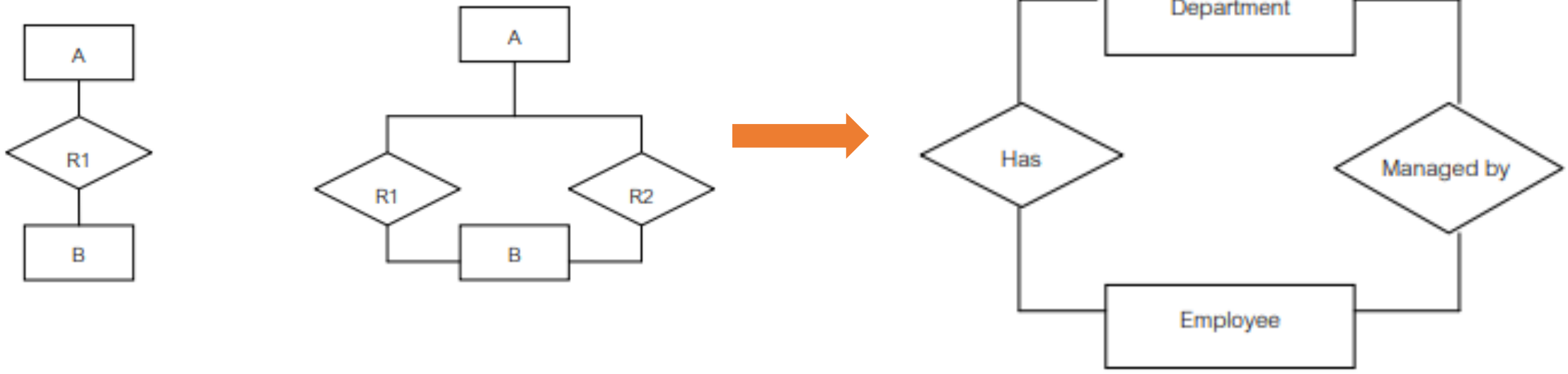


Top-Down Strategy (cont.)

C. Decompose relationship

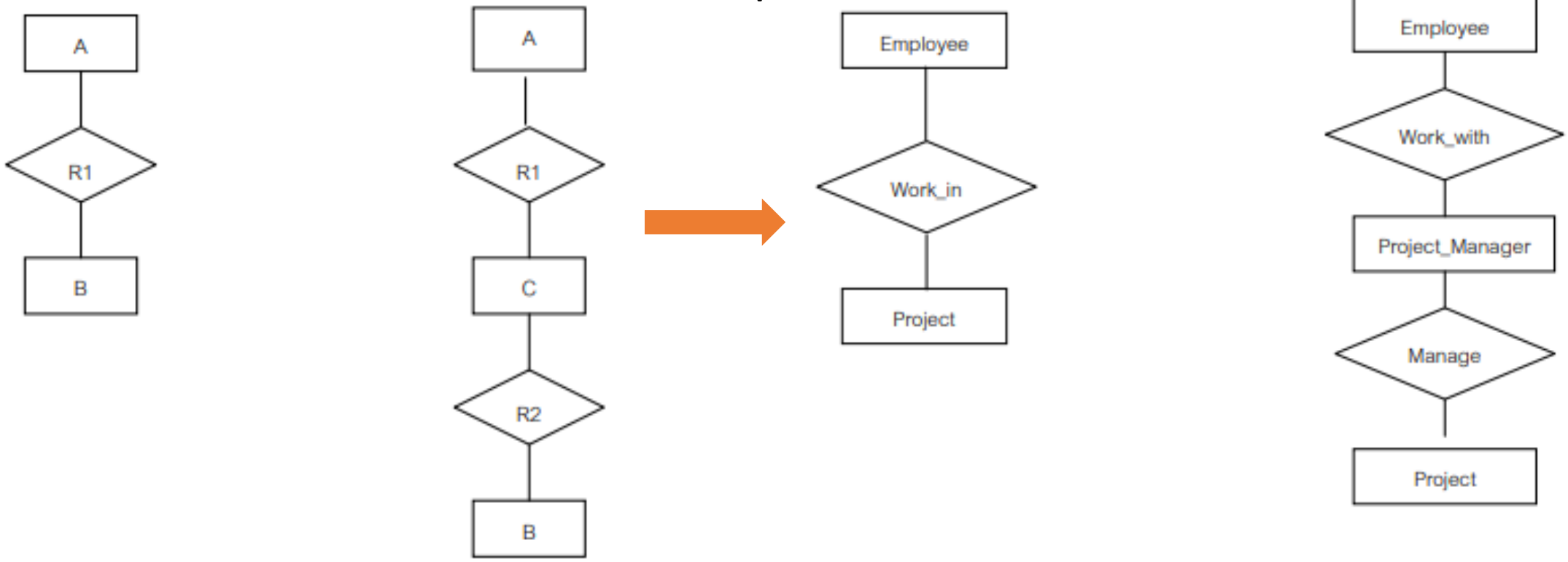
- Relationship → *Parallel Relationship*

R1 : Has
R2: Managed by



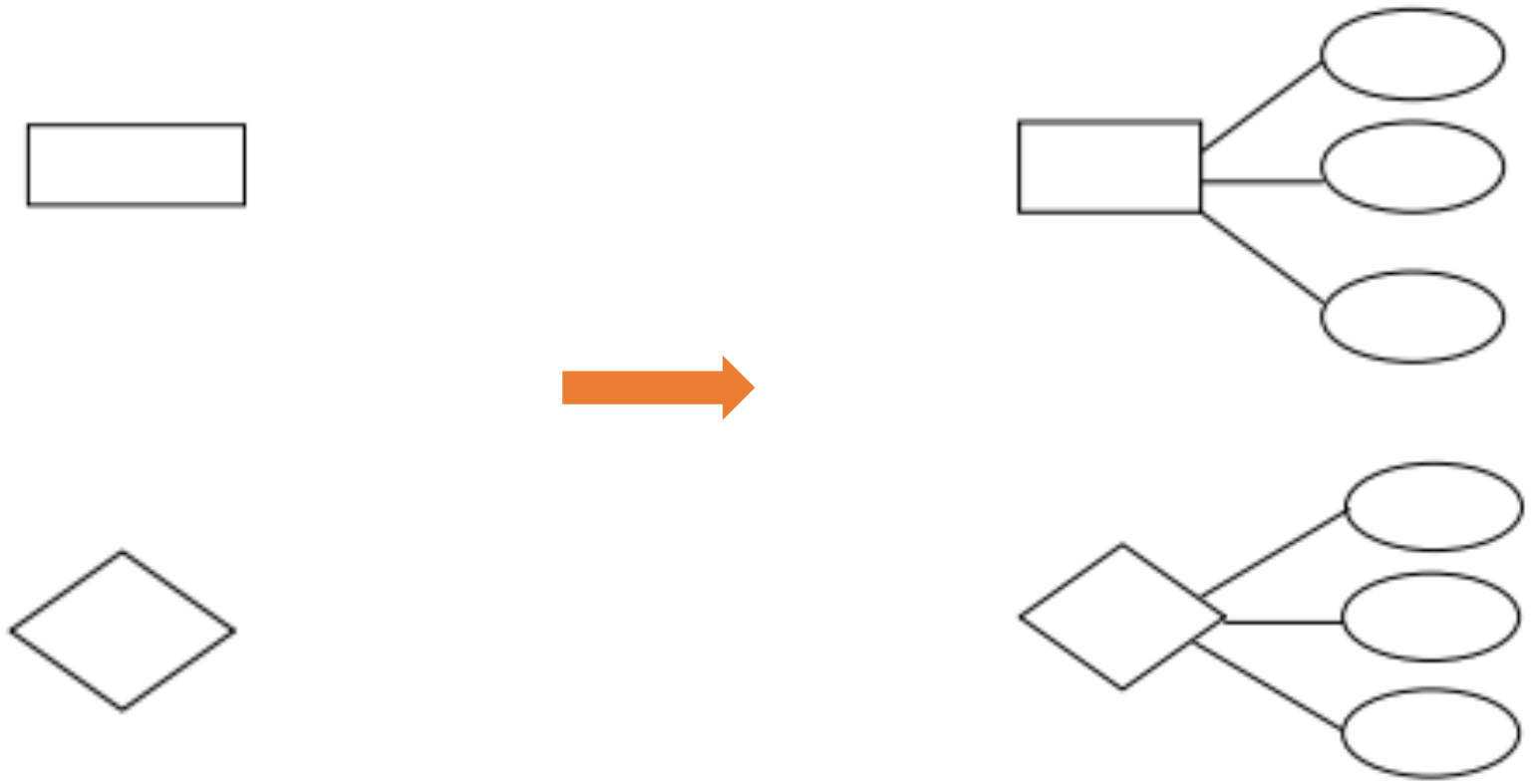
Top-Down Strategy (cont.)

- D. Transform relationship between 2 entities to new entity with new relationship
- Relationship → Entity with relationship



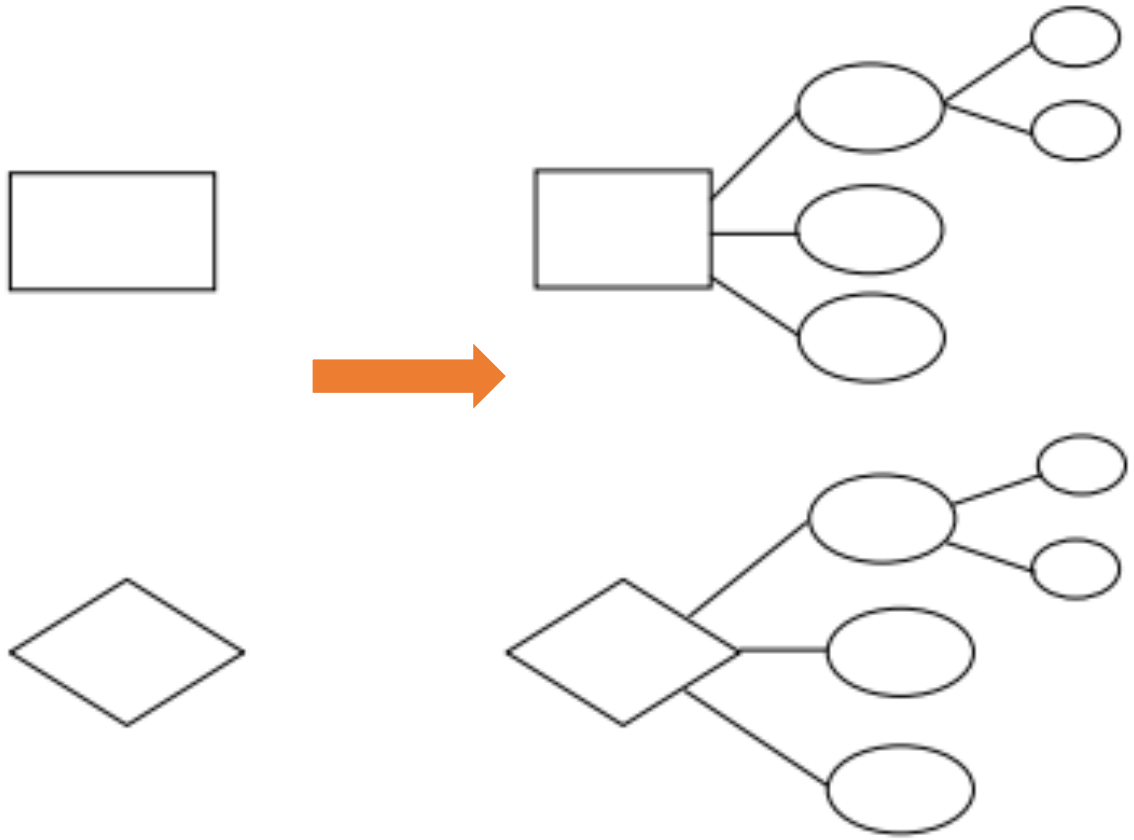
Top-Down Strategy (cont.)

- E. Attribute development for entity and relation



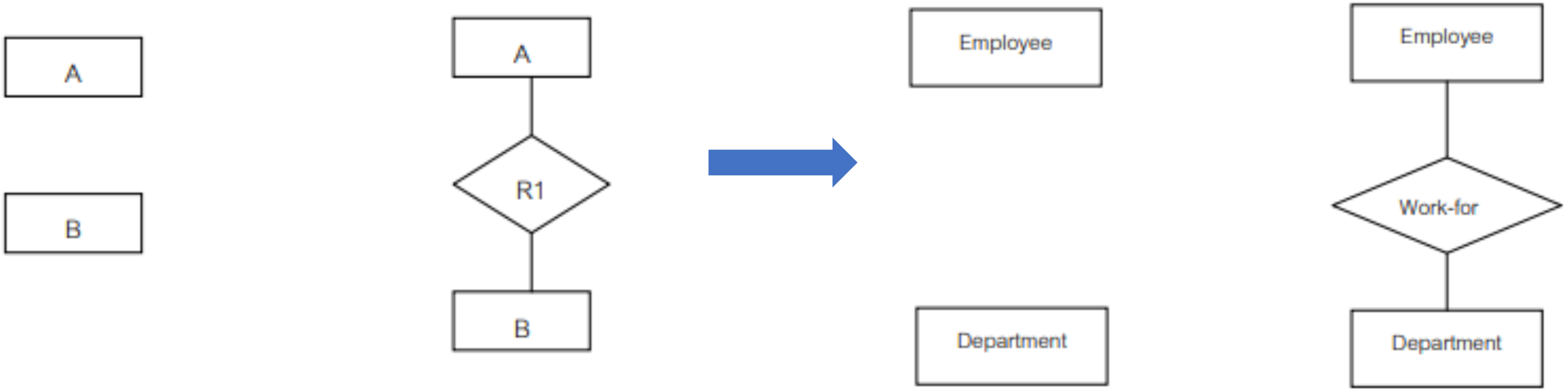
Top-Down Strategy (cont.)

- F. Define Composite attribute for entity or relationship



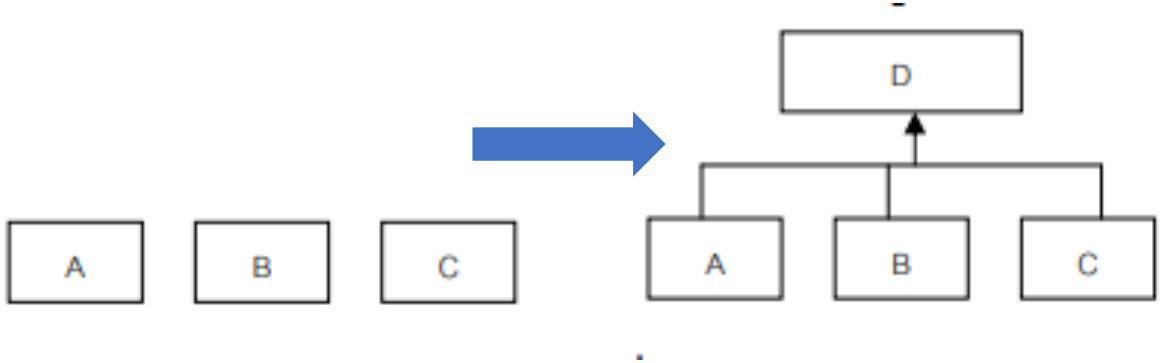
Bottom-Up Strategy

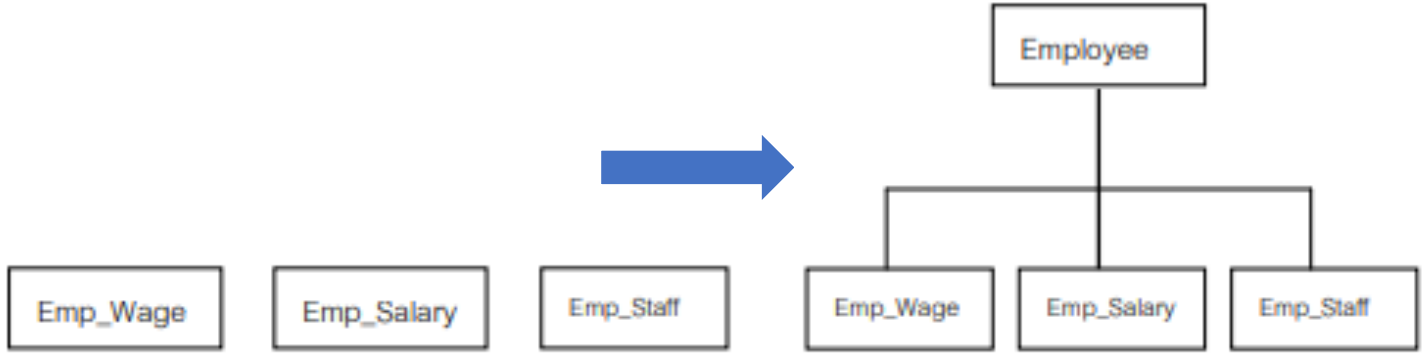
A. Generate relationship between 2 entity



Bottom-Up Strategy (cont.)

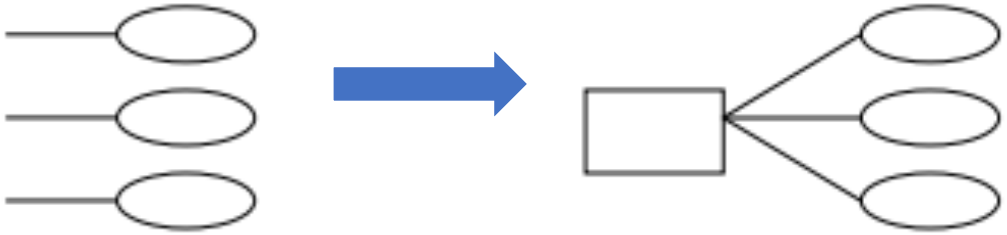
B. Create Superclass

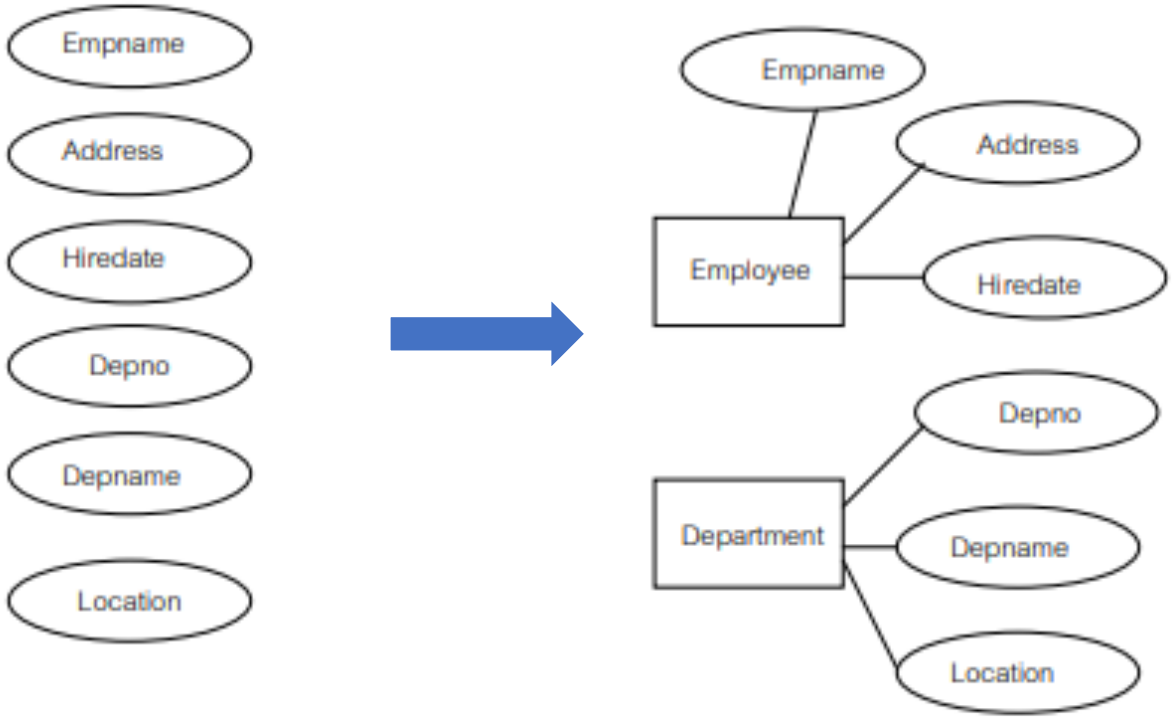




Bottom-Up Strategy (cont.)

- C. Attribute Aggregation





Bottom-Up Strategy (cont.)

D. Composite Attribute Aggregation