INFO20003 Database Systems

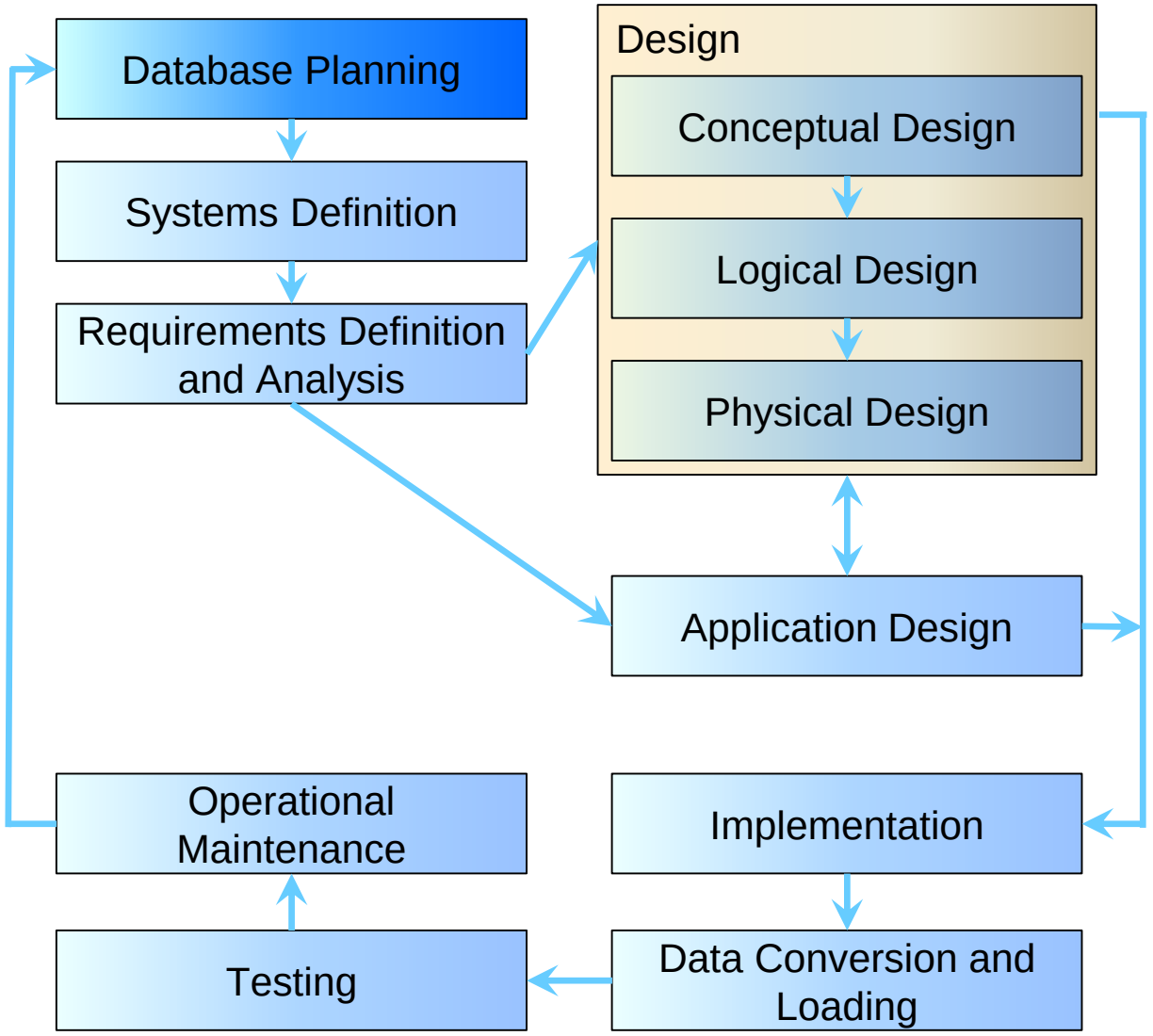
Dr Renata Borovica-Gajic

Lecture 03

Introduction to Data Modelling (ER)

Week 2

Database Development Lifecycle: Review



© University of Melbourne

The Entity-Relationship Model

- Basic ER modeling concepts
- Constraints
- Conceptual Design

Readings: Chapter 2, Ramakrishnan & Gehrke, Database Systems

© University of Melbourne

Conceptual Design: Objectives

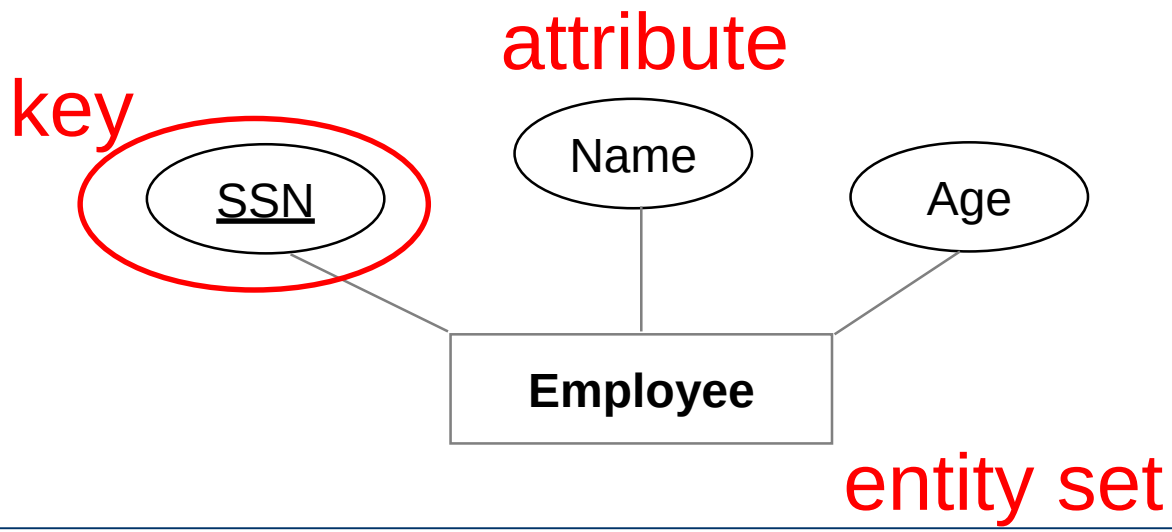
- What are the *entities* and *relationships* in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* that hold?

© University of Melbourne

ER Model: Entity & its attributes

- ***Entity***: Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes.*

- ***Entity Set***: A collection of entities of the same type (e.g. *all employees*)
  - All entities in an entity set have the same set of attributes
  - Each entity has a *key (underlined)*

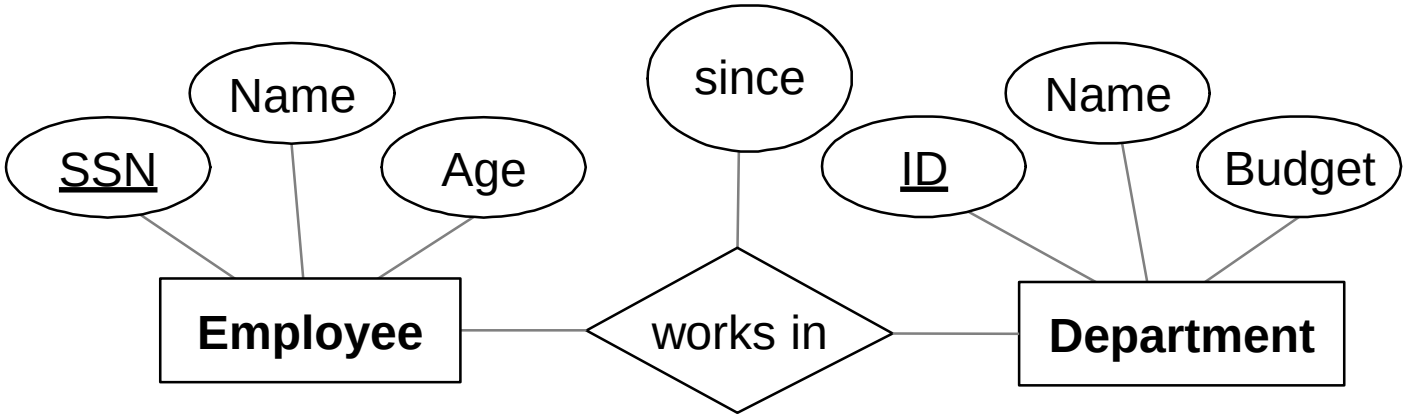


© University of Melbourne

ER Model: Relationship

- ***Relationship***: Association among two or more entities. Relationships can have their own attributes.
  - Example: Fred *works in* the Pharmacy department.

- ***Relationship Set***: Collection of relationships of the same type.
  - Example: Employees *work in* departments.



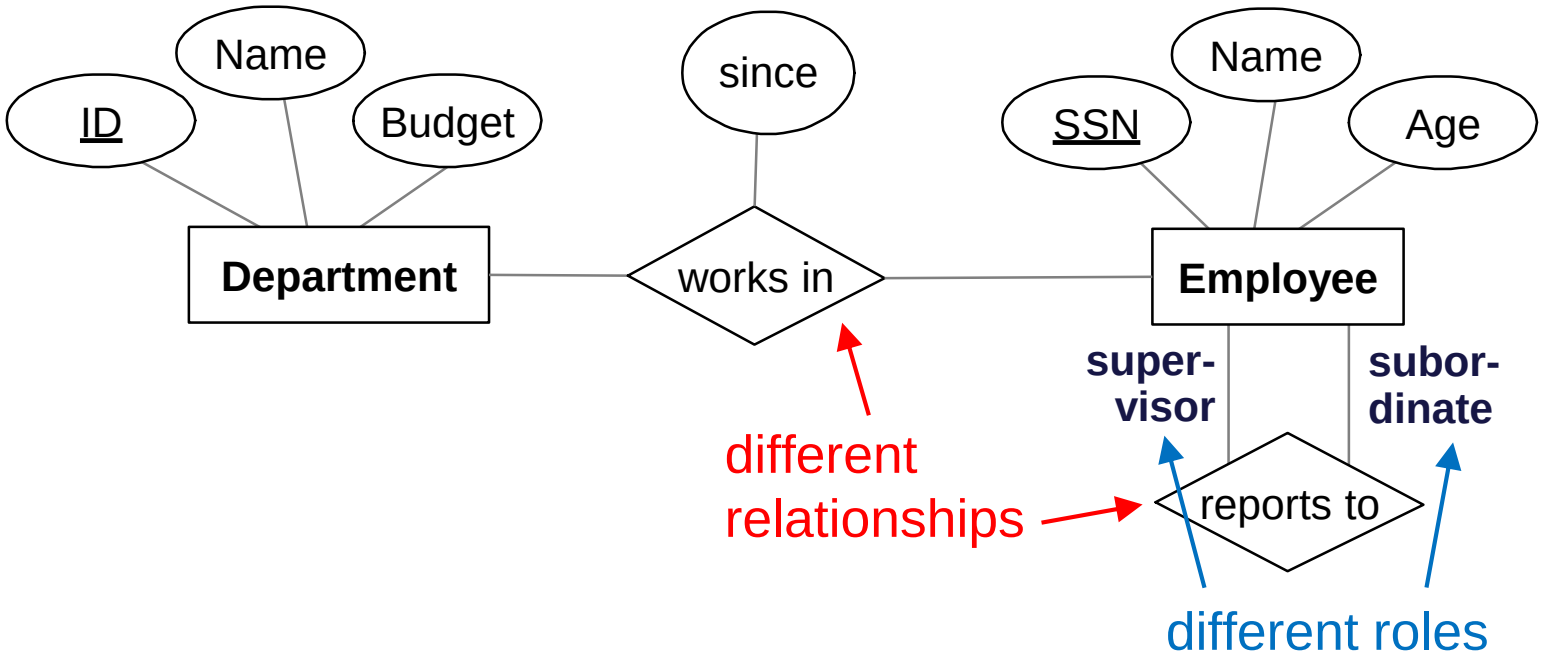
relationship set
(with a descriptive attribute)

© University of Melbourne

ER Model: Relationship roles

Same entity set can participate in:

- *different* relationship sets, or even
- *different "roles"* in the same set



© University of Melbourne

The Entity-Relationship Model

- Basic ER modeling concepts
- Constraints
- Conceptual Design

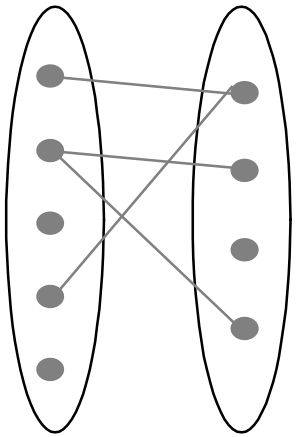
Readings: Chapter 2, Ramakrishnan & Gehrke, Database Systems

© University of Melbourne

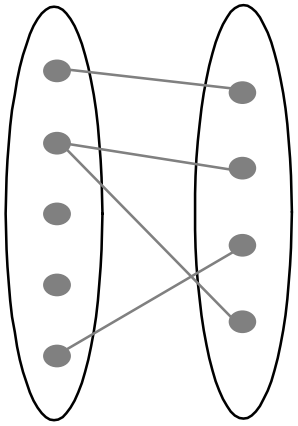
Key Constraints: Types

Key constraints determine the number of objects taking part in the relationship set (how many from each side)

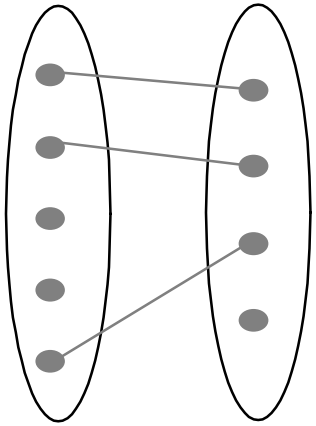
Types of key constraints:



Many-to-Many



One-to-Many (Many-to-One)



One-to-One

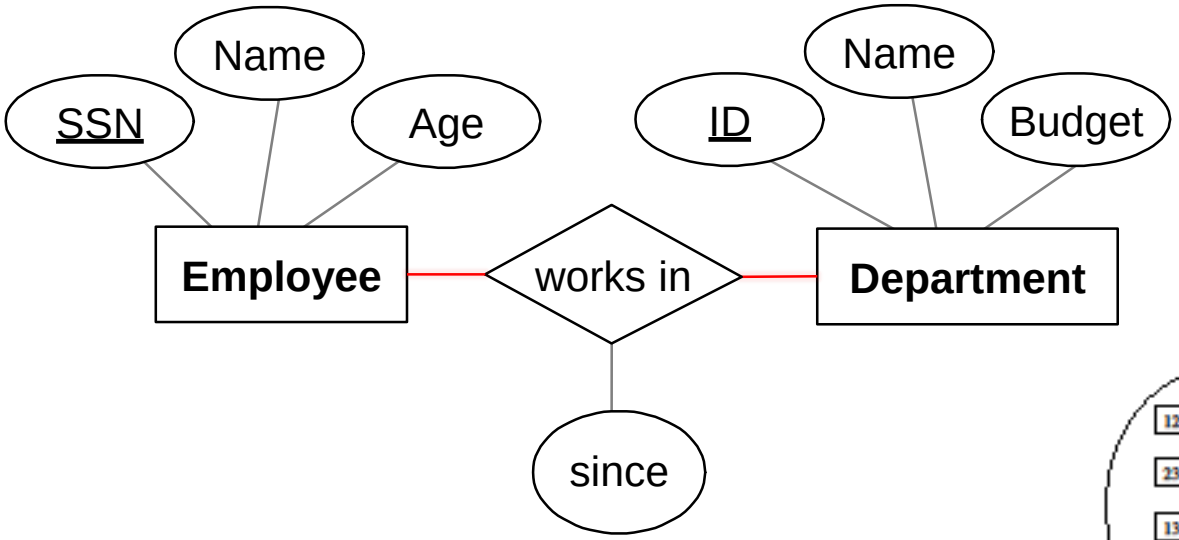
© University of Melbourne

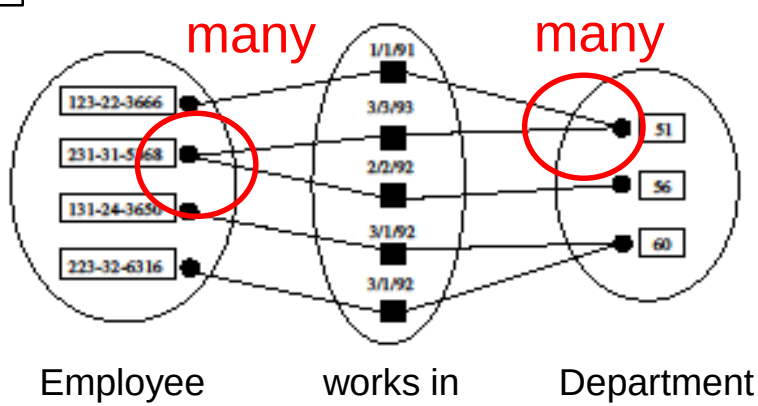
Key Constraints: Many-to-Many

Example:

An employee can work in many departments; a department can have many employees.

Many is represented by a line *(red is here just to emphasize it – no need to color).*





© University of Melbourne

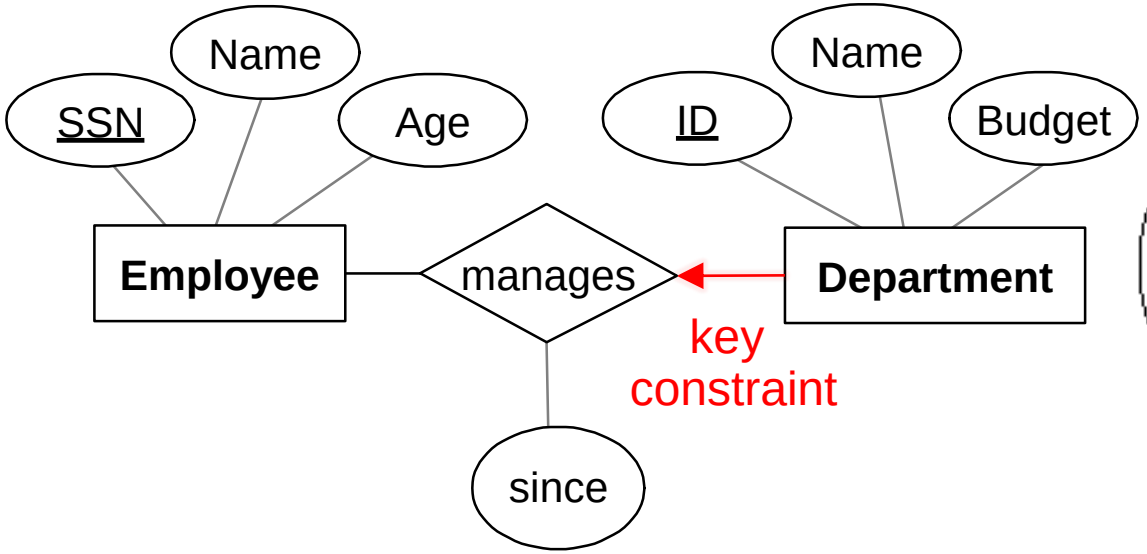
Key Constraints: One-to-Many

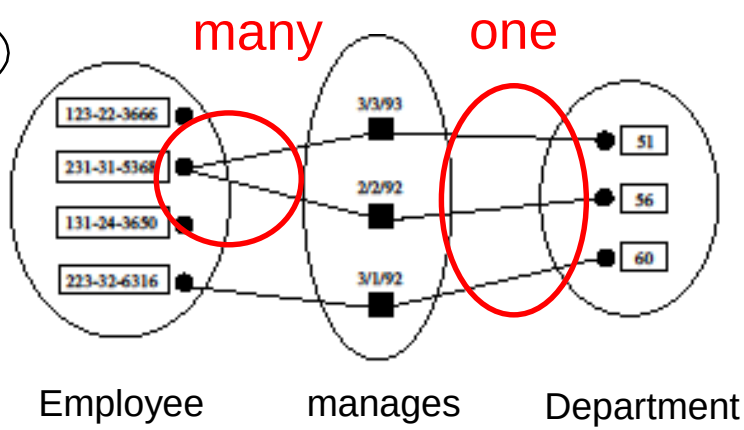
One-to-many constrains one entity set to have a *single* entity per a relationship. An entity of that set can never participate in two relationships of the same relationship set. This is called a **key constraint** and is represented by an arrow.

Example:

Each department has at most one manager.

This is the key constraint on Manages.



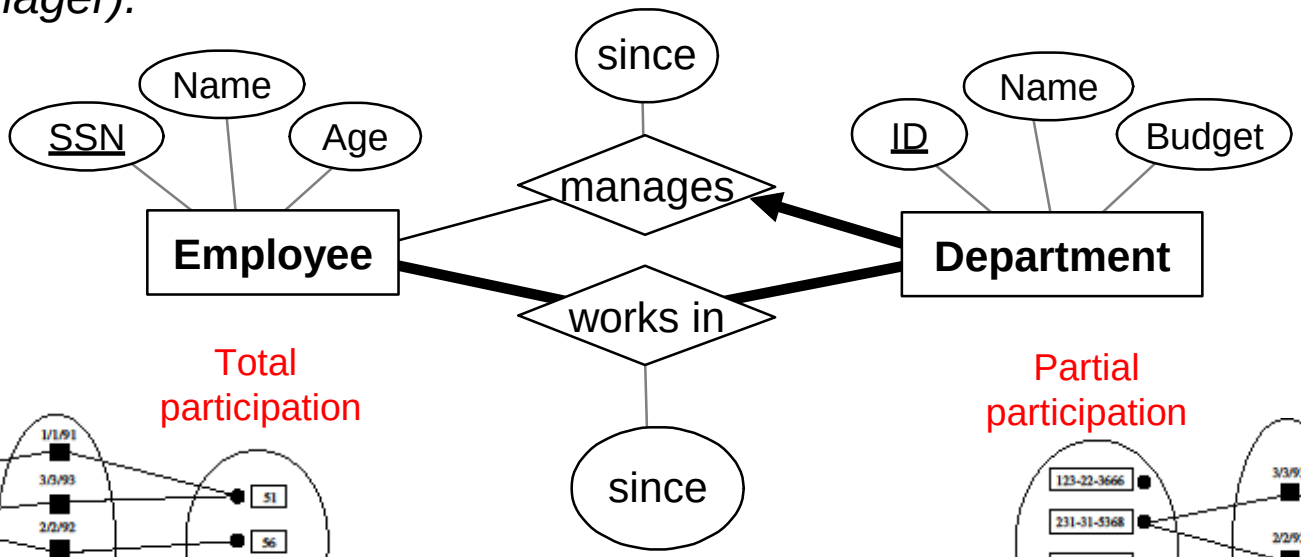


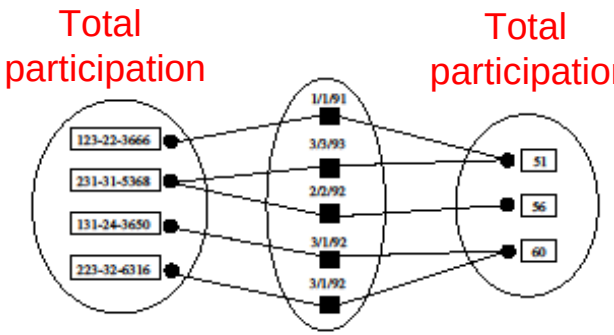
© University of Melbourne

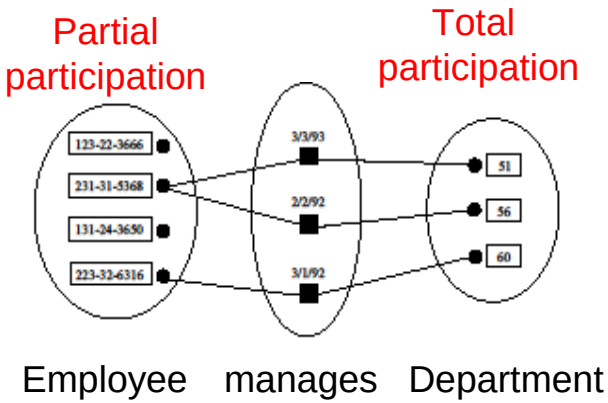
Participation Constraints

Participation constraint explores whether all entities of one entity set take part in a relationship. If yes this is a **total** participation, otherwise it is **partial**. Total participation says that each entity takes part in "**at least one**" relationship, and is represented by a bold line.

Example: *Every employee must work in a department. Each department has at least one employee. Each department has to have a manager (but not everyone is a manager).*





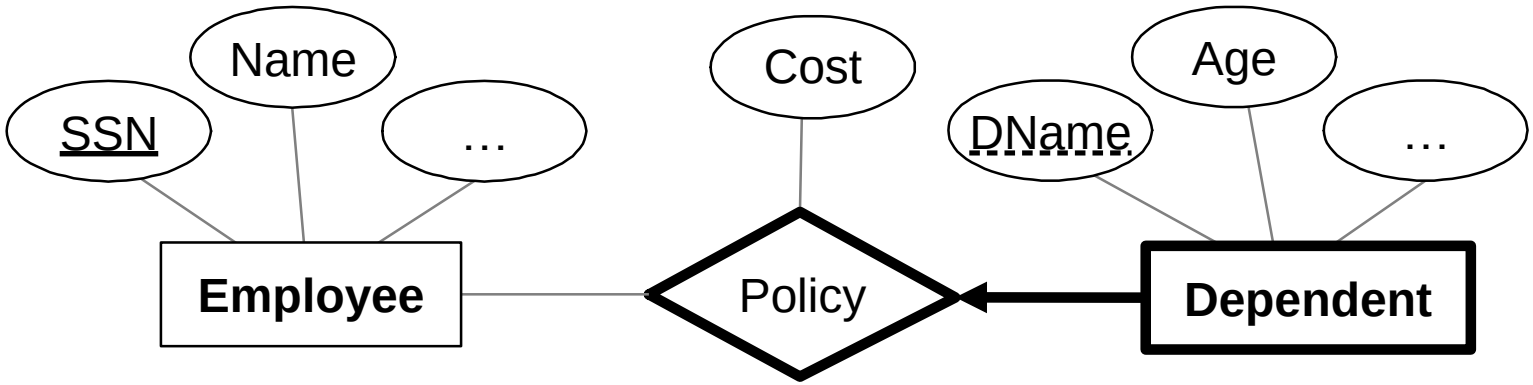


© University of Melbourne

Weak Entities

A ***weak entity*** can be identified uniquely only by considering (the primary key of) another (*owner*) entity. They are represented as a "bold" rectangle.

- Owner entity set and weak entity set must participate in a relationship where each weak entity has one and only one strong entity to depend on (key constraint)
- Weak entity set must have total participation in this relationship set. Such relationship is called *identifying* and is represented as "bold".

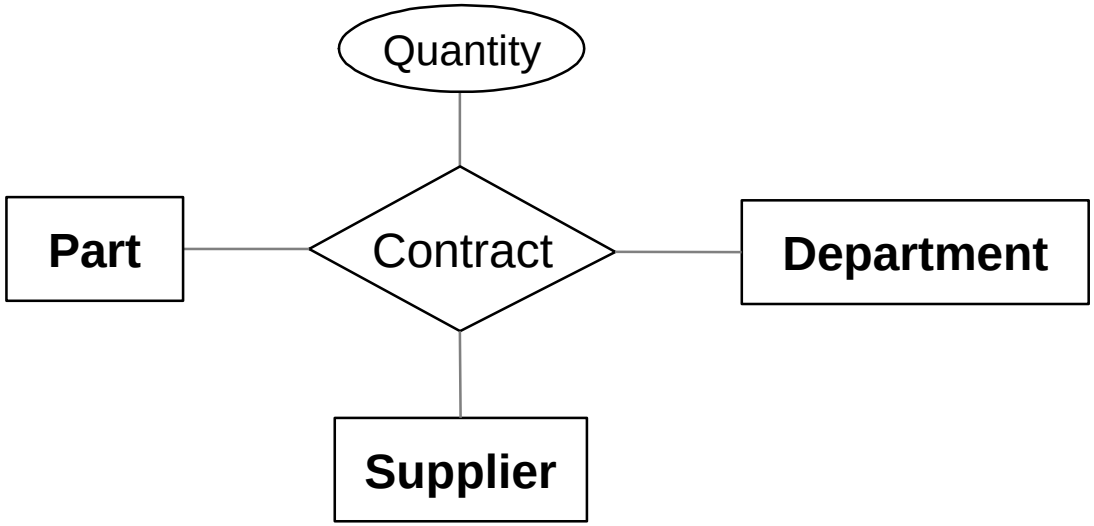


Weak entities have only a "partial key" (dashed underline) and they are identified uniquely only when considering the primary key of the owner entity

© University of Melbourne

Ternary Relationships

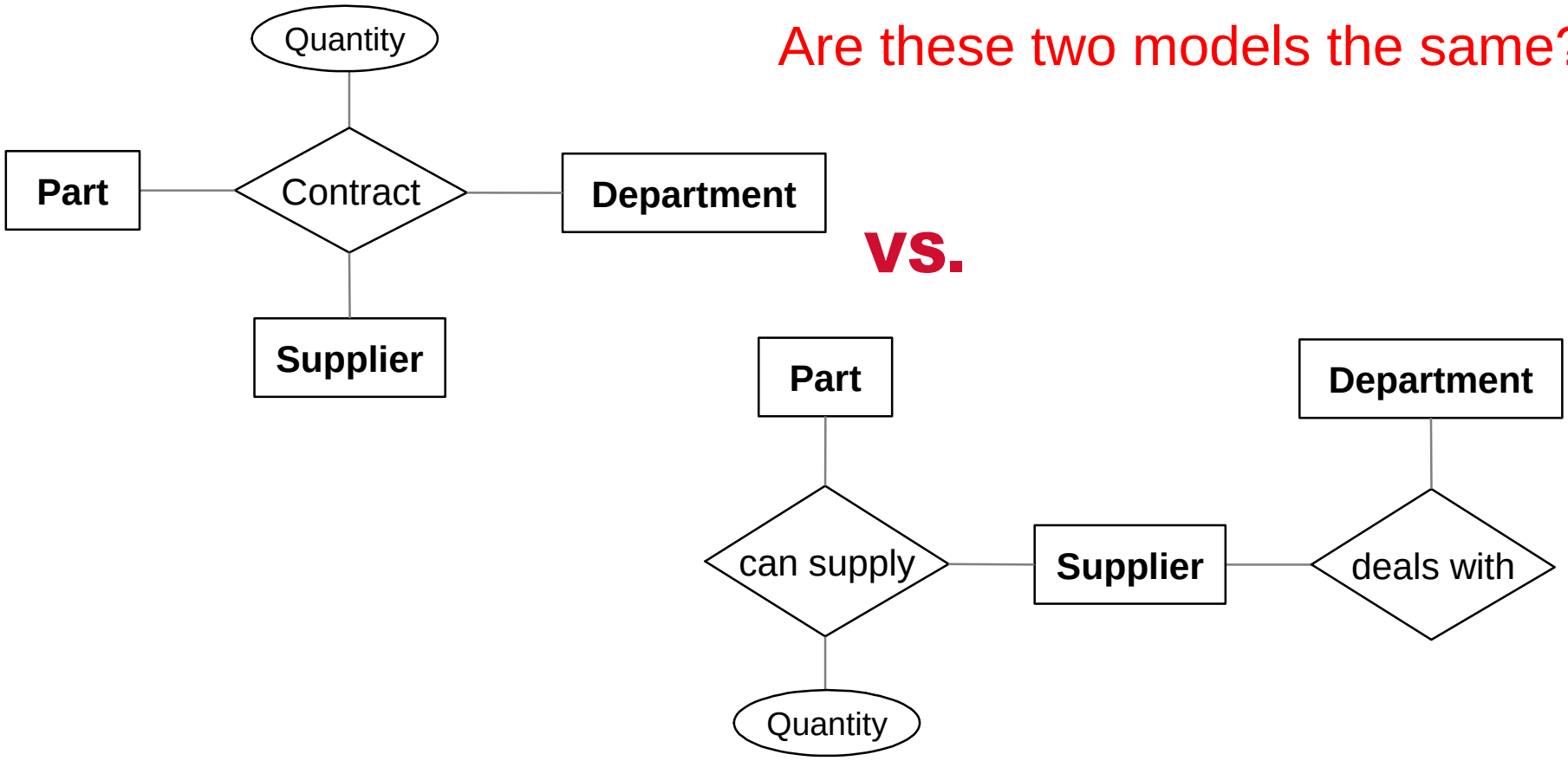
In general, we can have **n**-ary relationships, and relationships can have attributes



This is a ternary relationship
with one relationship attribute

© University of Melbourne

Ternary vs. Binary Relationships



Second model:

- S “can supply” P, D “needs” P, and D “deals with” S does not imply that D has agreed to buy P from S. Not the same!

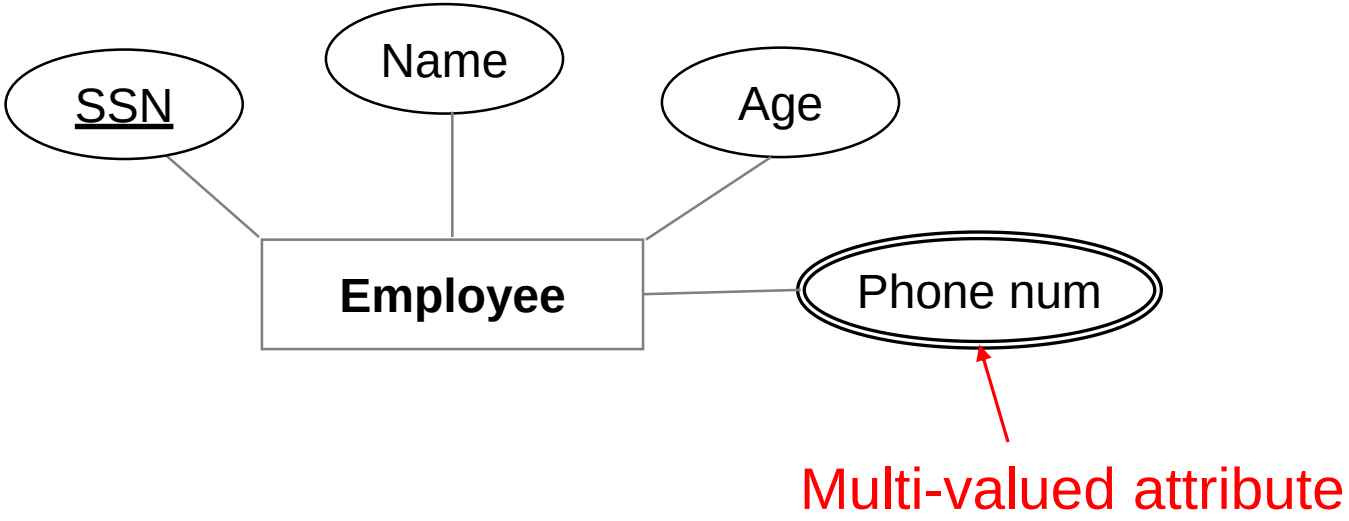
© University of Melbourne

Special attribute type: Multi-valued attributes

- Multi-valued attributes can have multiple (finite set of) values of the same type.

Example:

For employees we need to capture their home phone number and work phone number.

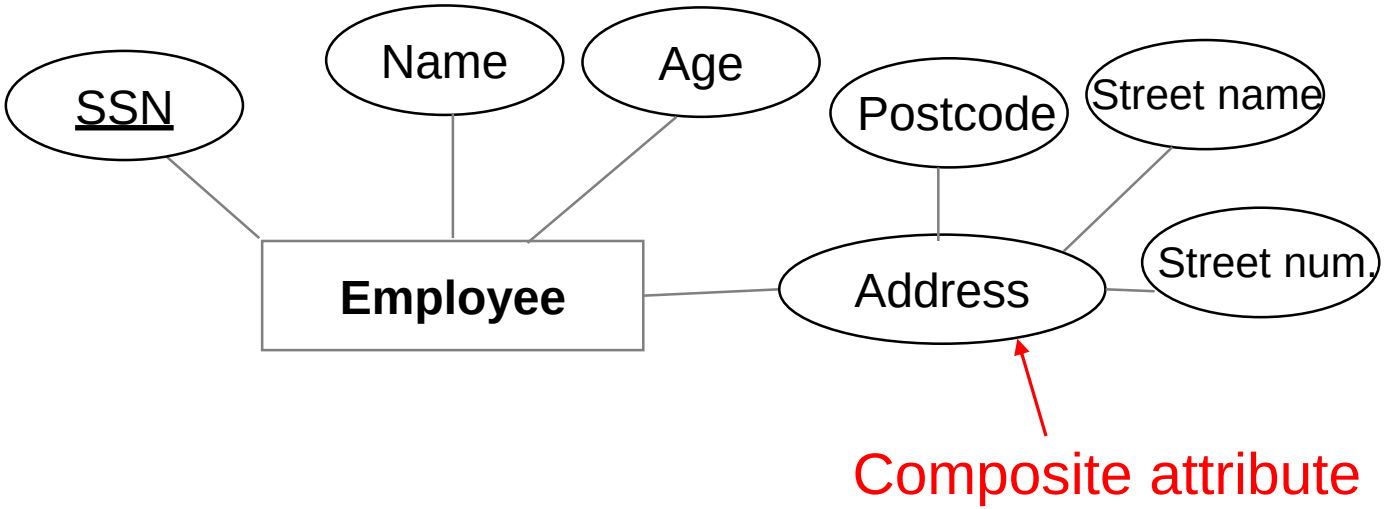


Special attribute type: Composite attributes

- Composite attributes have a structure hidden inside (each element can be of different type).

Example:

For employees we need to capture an address consisting of a postcode, street name and number.



Now you try

University database schema:

- *Entities:* Courses, Professors
- Each course has id, title, time

- Make up suitable attributes for professors

© University of Melbourne

Now you try

1. Every professor must teach some course.

2. Every professor teaches exactly one course (no more, no less).

3. Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor.

© University of Melbourne

The Entity-Relationship Model

- Basic ER modeling concepts
- Constraints
- Conceptual Design

Readings: Chapter 2, Ramakrishnan & Gehrke, Database Systems

© University of Melbourne

Conceptual Design Using the ER Model

- **Design choices:**
  - Should a concept be modelled as an entity or an attribute?
  - Should a concept be modelled as an entity or a relationship?
  - Should we model relationships as binary, ternary, n-ary?

- **Constraints in the ER Model:**
  - A lot of data semantics can (and should) be captured

© University of Melbourne

Entity vs. Attribute

Example:

Should "*address*" be an attribute of Employees or an entity (related to Employees)?

Answer:

- *Depends* upon how we want to use address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity

© University of Melbourne

Notes on the ER design

- ER design is *subjective*. There are often many ways to model a given scenario.
- Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship.
- There is no standard/notation (we will cover two notations, today we learned **Chen's** notation)

© University of Melbourne

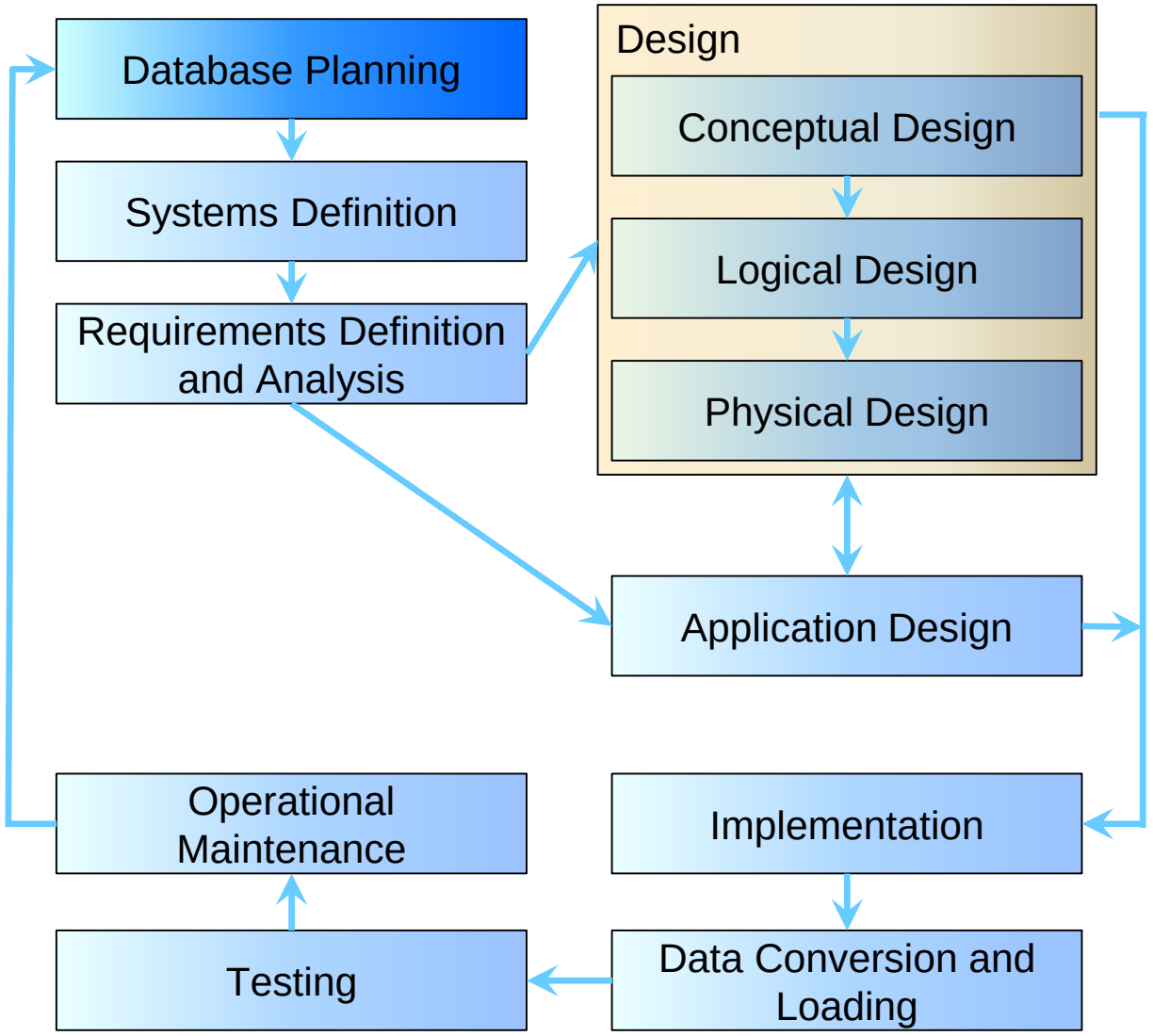
Summary of Conceptual Design

- Conceptual design follows requirements analysis
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications
  - Originally proposed by Peter Chen, 1976

    Note: there are many variations on ER model
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships)
- Some additional constructs: *weak entities*

© University of Melbourne

Database Development Lifecycle: Review



© University of Melbourne

What's Examinable

- Need to be able to draw conceptual diagrams on your own
  - Given a problem, *determine entities, attributes, relationships*
  - What is key constraint and participation constraint, weak entity?
  - Determine constraints for the given entities & their relationships

© University of Melbourne

Next Lecture

- Continue exploring modelling
  - From conceptual through to physical
  - Introducing **relational model**

© University of Melbourne