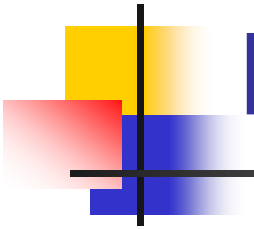
COMP1140: Database and Information Management

Lecture Note – Week 03

Dr Suhuai Luo

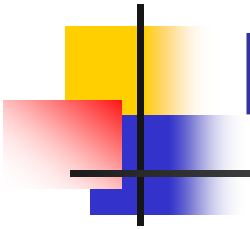
School of SEEC

University of Newcastle



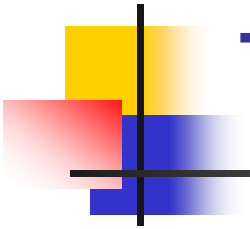
Notice

- You are supposed to be able to download now and use at home Microsoft packages through Microsoft Imagine.
- Make sure to catch up Week 2's lab



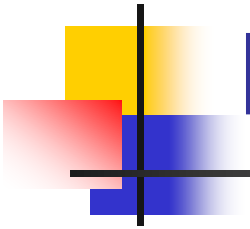
Last lecture

- Database and DBMS architectures
  - **Three-level database architecture**
  - **Multi-User DBMS architectures**
- Assignment discussion:
  - Requirements gathering



This lecture

- Conceptual Database Design with Entity-Relationship Diagrams
- Further clarification on A1
- Ref: chapter 12, 13, 16

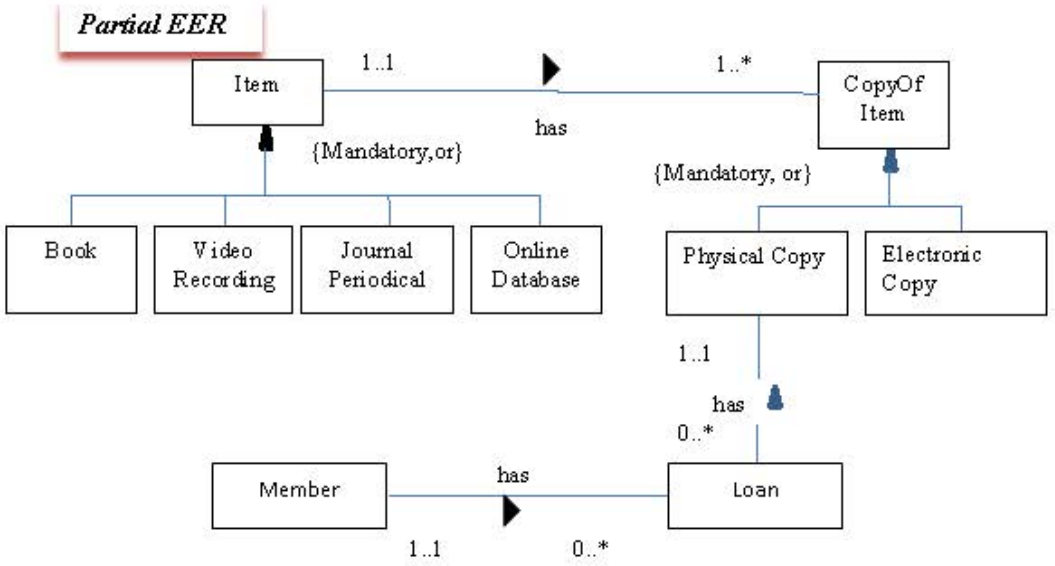


Entity-Relationship Model

- At conceptual database design stage, we represent data requirements gathered in requirement analysis, unambiguously
- Entity-Relationship (ER) Model is a semantic data model, popularly used at the conceptual database design stage

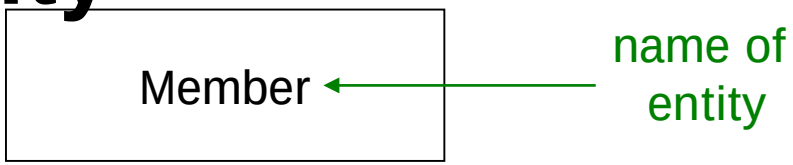
Entity-Relationship Model

- ER model represents concepts graphically
- There are many different graphical notations used. We will use Unified Modeling Language (UML) to draw ER diagrams
- Note: in assignment 1, You MUST use the format and style adopted by the textbook and the course.



Entity

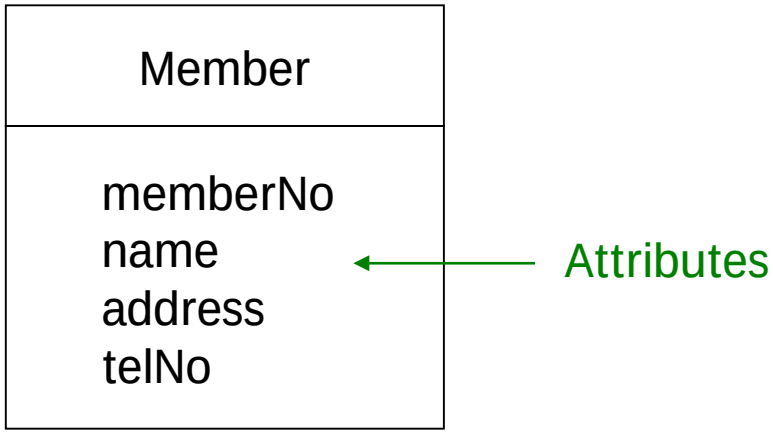
- **entity type:** a group of objects with the same properties. It has an independent existence.
- Also, known as **entity**
- Graphically,



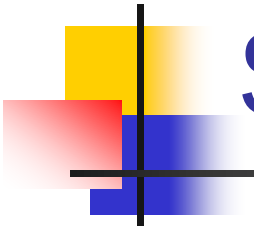
- **entity occurrence:** an uniquely identifiable object of an entity type
- E.g. fruit

Attributes

- An entity occurrence is described by its attributes
- Each attribute has an **attribute domain**: a set of possible values the attribute can have
- Graphically,



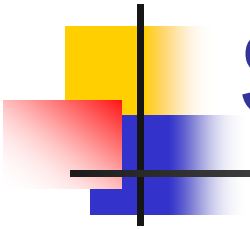
Simple vs. Composite



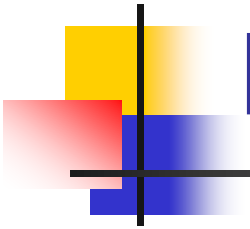
- An attribute composed of a single component is called a **simple attribute**
  - E.g. memberNo

- An attribute may be composed of mulitple components: **composite attribute**
  - For instance, we can model ***address*** as a set of components as ***no***, ***street***, ***city***, and ***zip***.

Single vs. Multi-valued



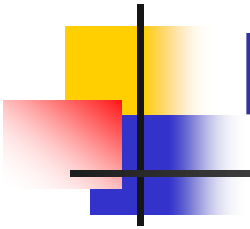
- An attribute that holds a single value for each occurrence is called a **single-valued** attribute
  - E.g. memberNo has a single value for each member
- An attribute that holds multiple values for each occurrence is called a **multi-valued** attribute
  - E.g. There could be multiple telephone number (telNo) for each member.



Derived attribute

- Sometimes, attributes may be derived from other attributes (called a **derived attribute**)

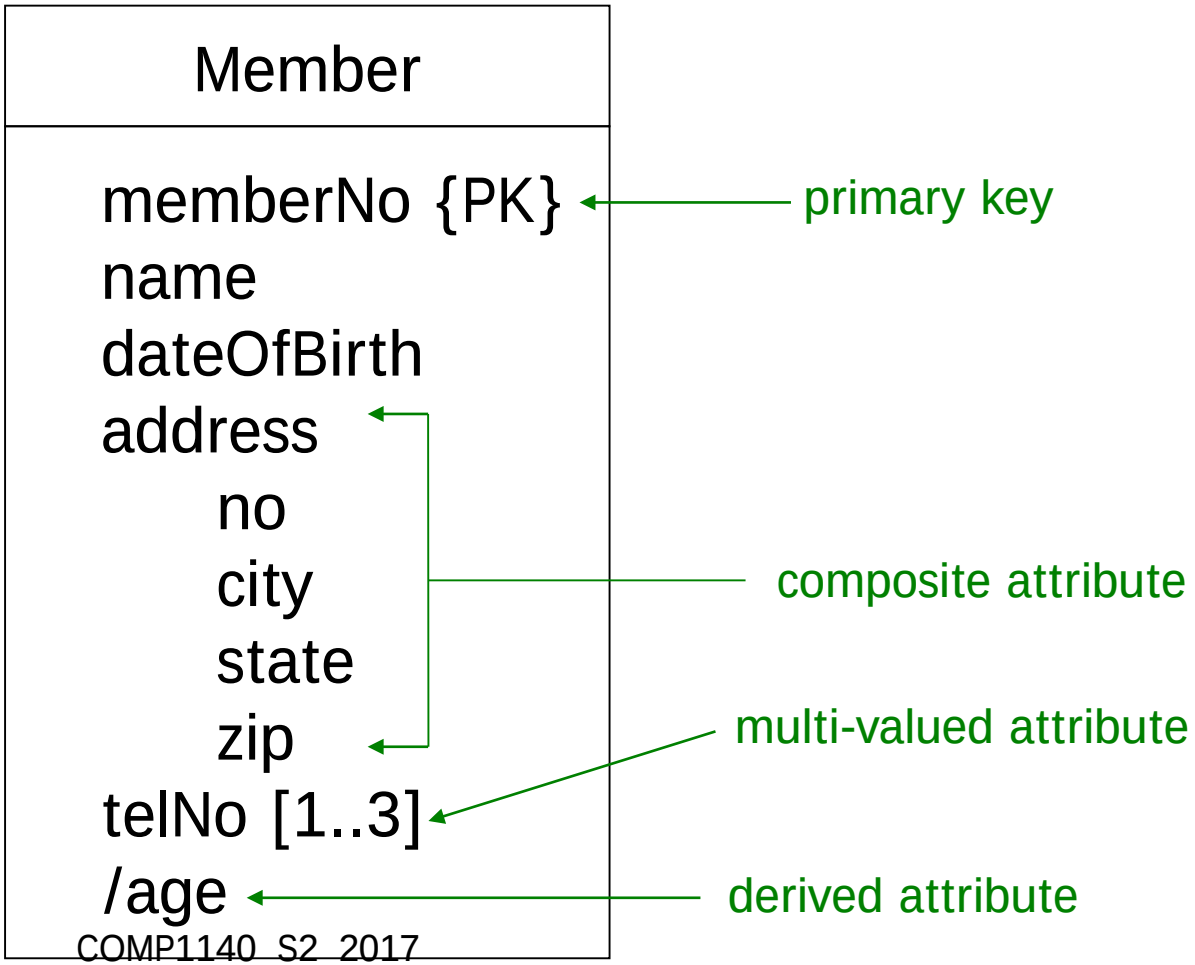
- For instance, **age** is derivable from date of birth and current date

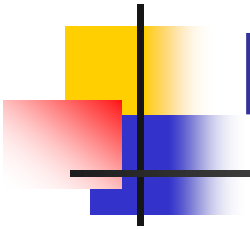


Keys

- A **candidate key** is the minimal number of attributes that uniquely identifies each occurrence of an entity set
- A candidate key is selected and designated as the **primary key** (used to uniquely identify an entity occurrence)
- Sometimes, two or more attributes make a candidate key, which is referred to as **composite key**

Graphical Representation



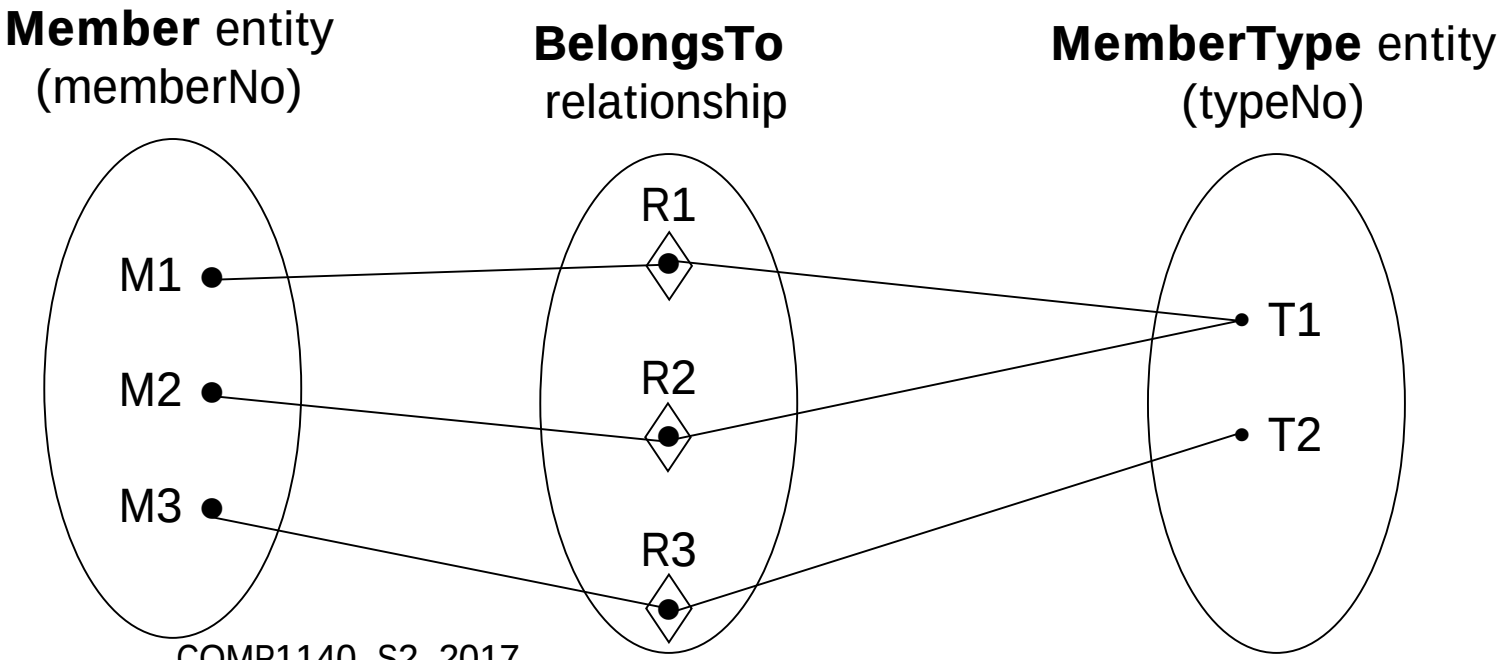


Relationships

- A **relationship** is an association among two or more entities
- A collection of similar relationships is called a **relationship type**
- **relationship occurrence**: a uniquely identifiable association that includes one occurrence from each participating entity type.

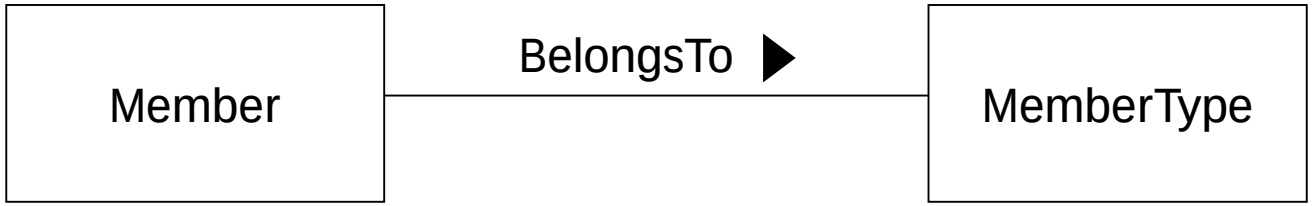
Relationships (contd.)

- Individual relationship occurrences can be viewed using a semantic net.



Relationships (contd.)

- Graphically,



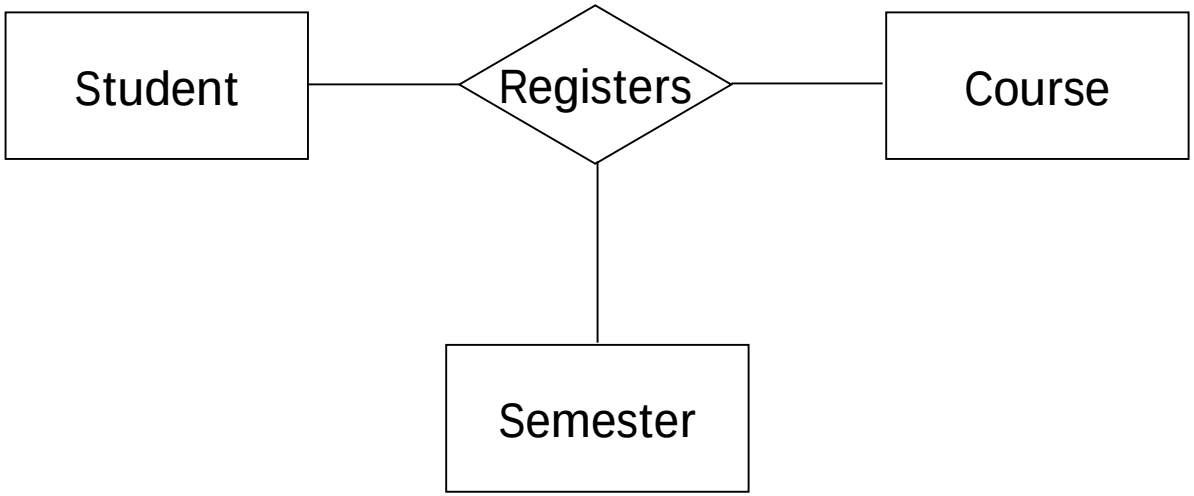
'Member belongs to a MemberType'

Degree of a relationship set

- The number of participating entity types in a relationship is called the **degree of the relationship**
  - A relationship of degree two is called a **binary** relationship
  - A relationship of degree three is called a **ternary** relationship
  - A relationship of degree greater than 3 is called an **n-ary** relationship
  - A relationship of degree one is called a **unary / recursive** relationship

Degree of a relationship (contd.)

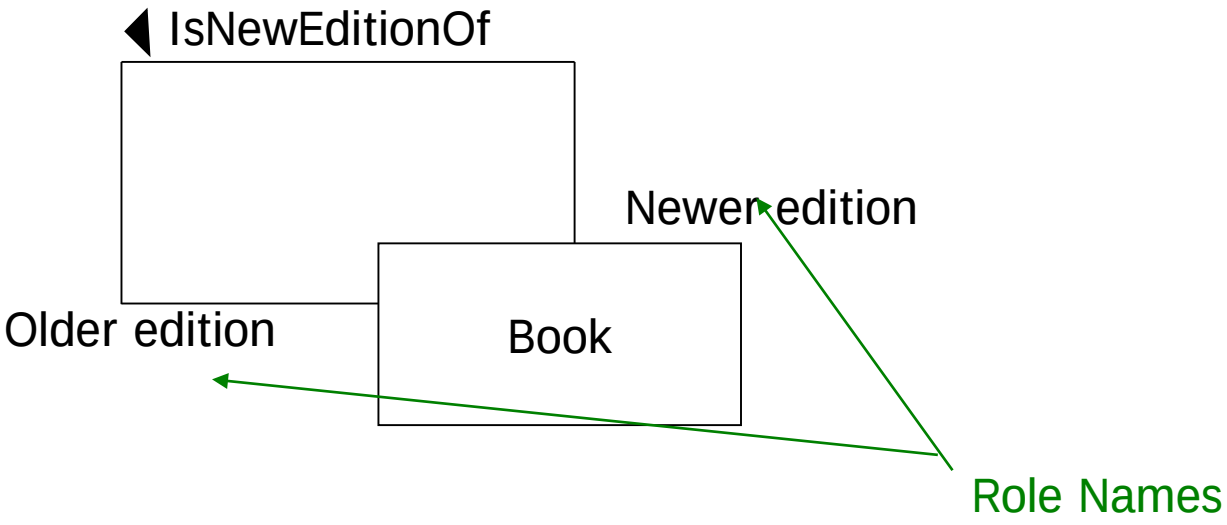
- Graphically,



Relationships with degree more than 2 is represented using a diamond

Recursive Relationship

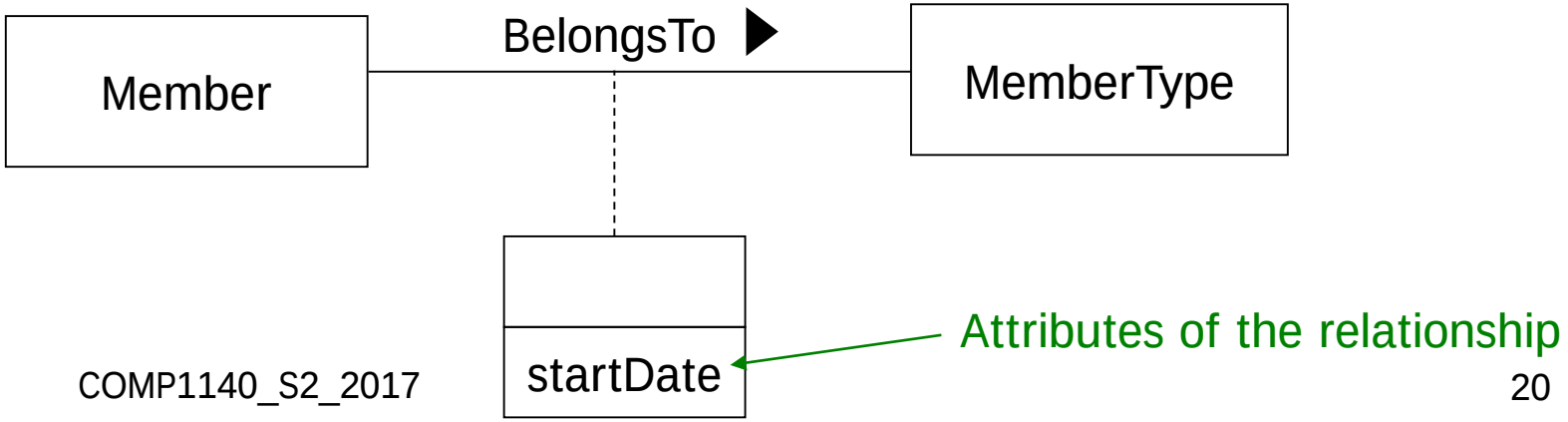
- The entities that the relationship relates need not be distinct:
- **RR**: a relationship type where the same entity type participates more than once in different roles



Attributes on relationships

- Relationships may contain attributes (also known as **descriptive attributes**)

- Graphically,

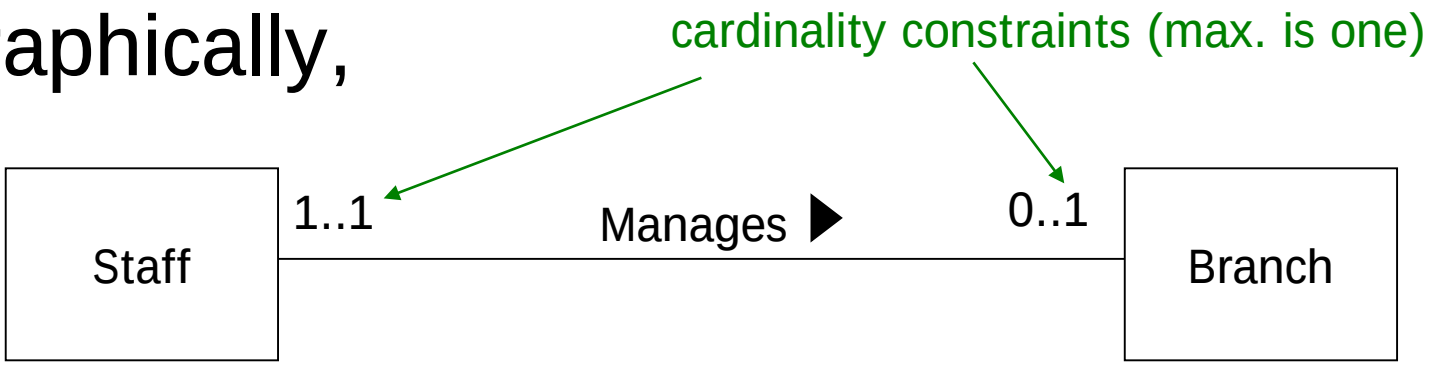


Cardinality constraints

- **Constraints on relationship - Multiplicity (**Cardinality & participation)
- **Cardinality constraints** specifies the maximum number of possible occurrences of an entity type in a particular relationship
- For binary relationships, there are three types of cardinality constraints:
  - One-to-one
  - One-to-many (or Many-to-one)
  - Many-to-many

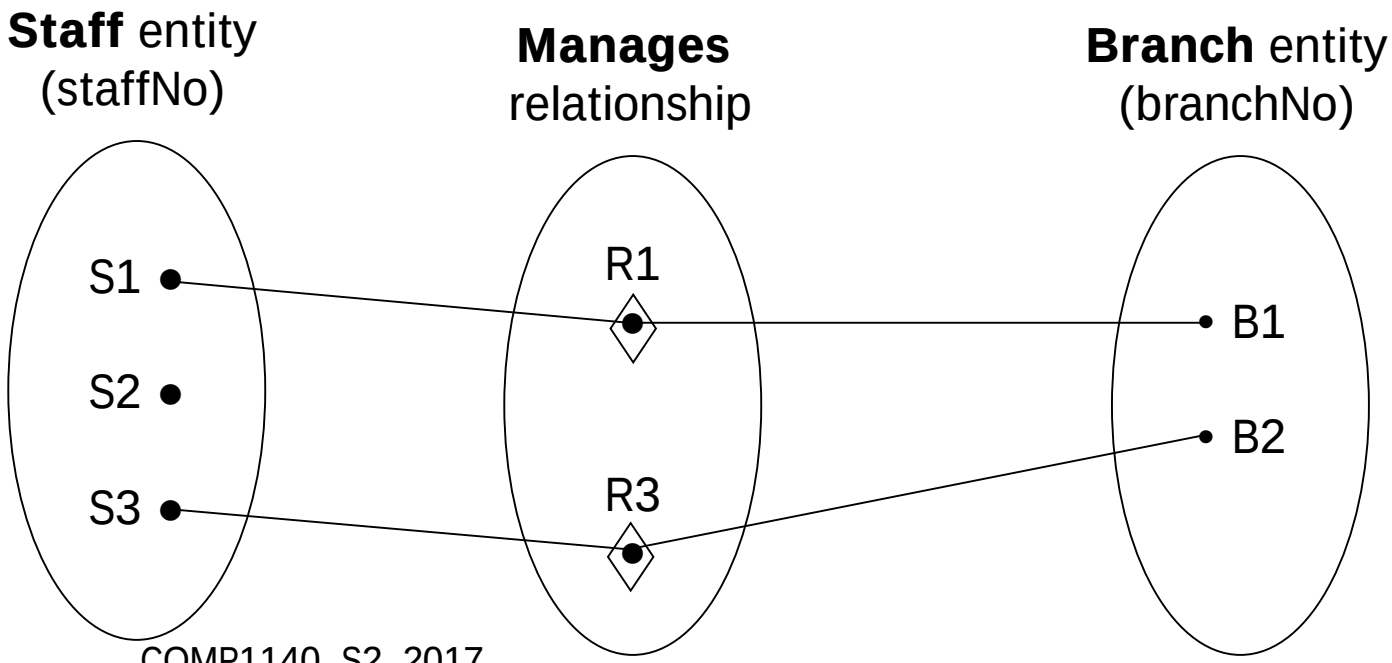
One-to-one (1:1) relationship

- Example:
  - A staff member can manage at most one branch at a time
  - One branch is managed by one staff (at most)
- Graphically,



One-to-one (1:1) relationship (contd.)

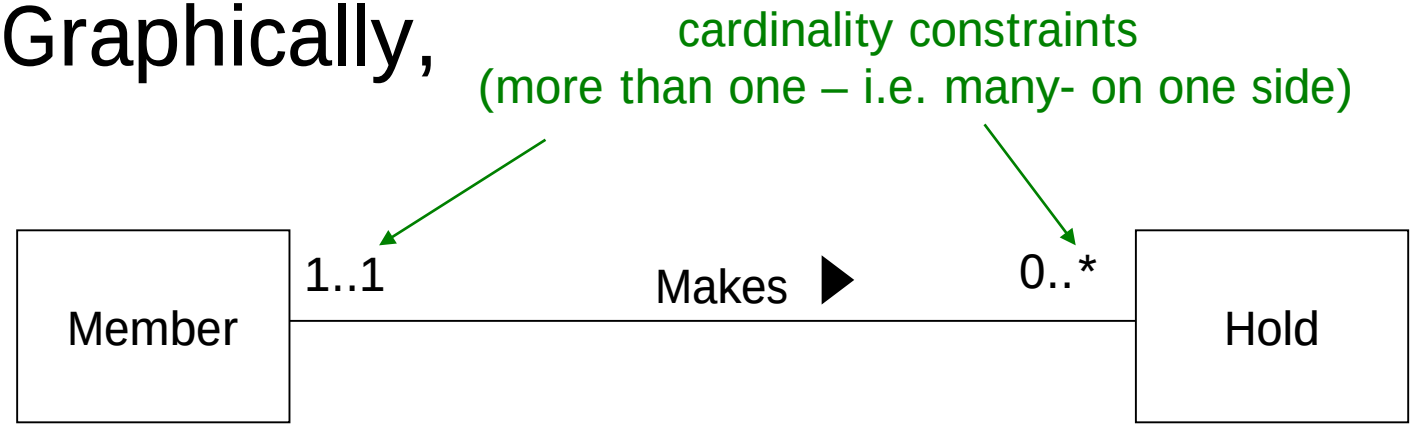
- Semantic Net Example



One-to-many (1:*) relationship

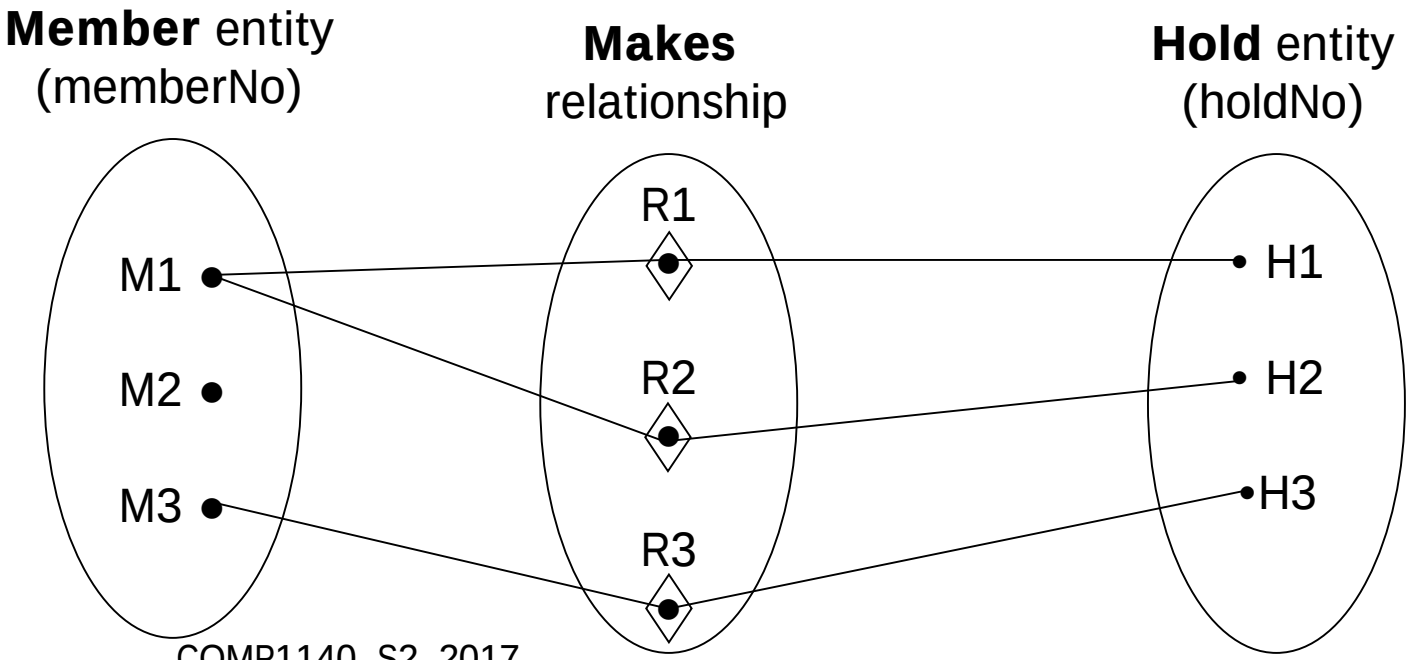
- Example
  - A member can make many hold requests
  - A hold request is made by a member

- Graphically,



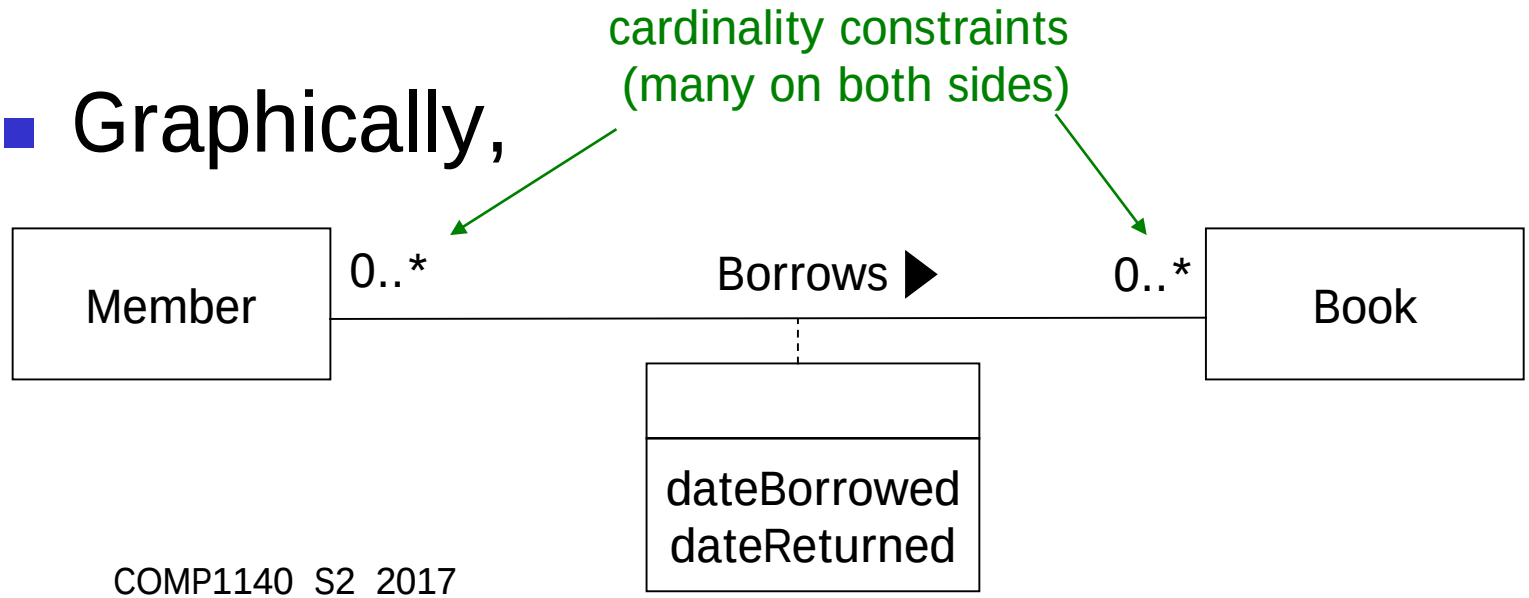
One-to-many (1:*) relationship (contd.)

- Semantic Net Example



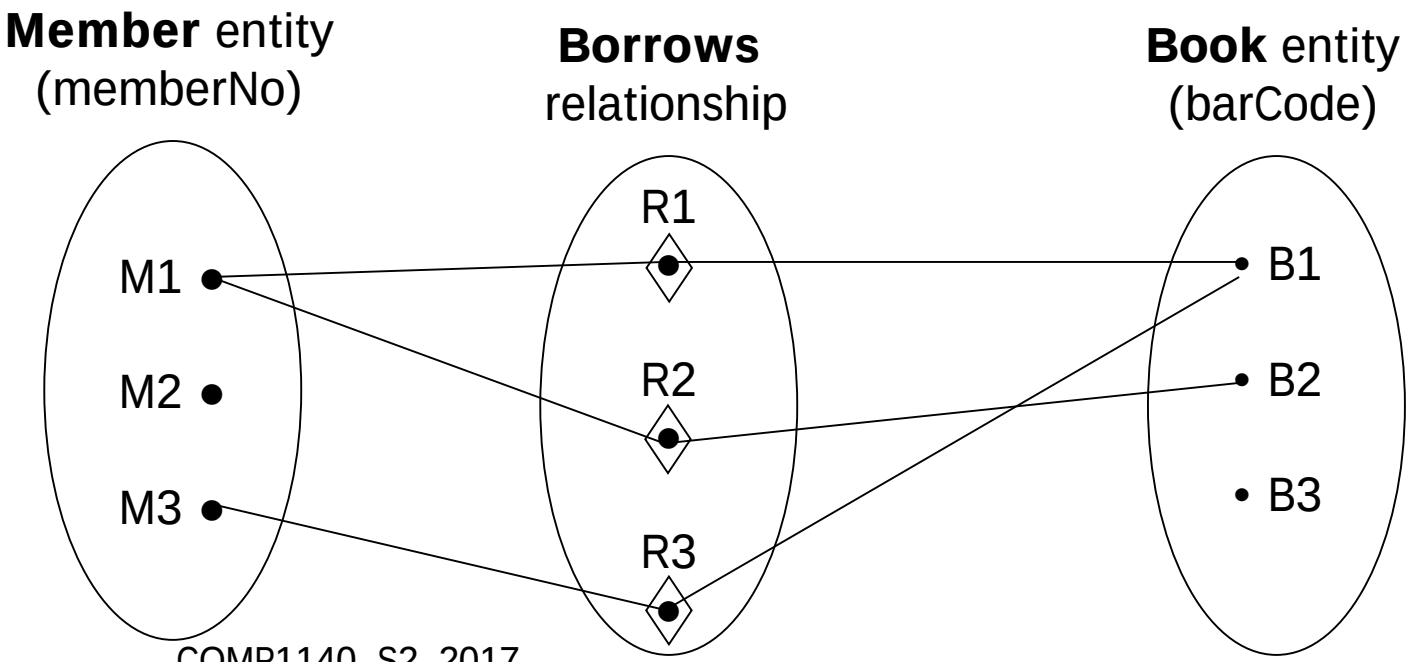
Many-to-many (*:*) relationship

- Example:
  - A book is borrowed by many members at different times
- Graphically,



Many-to-many (*:*) relationship

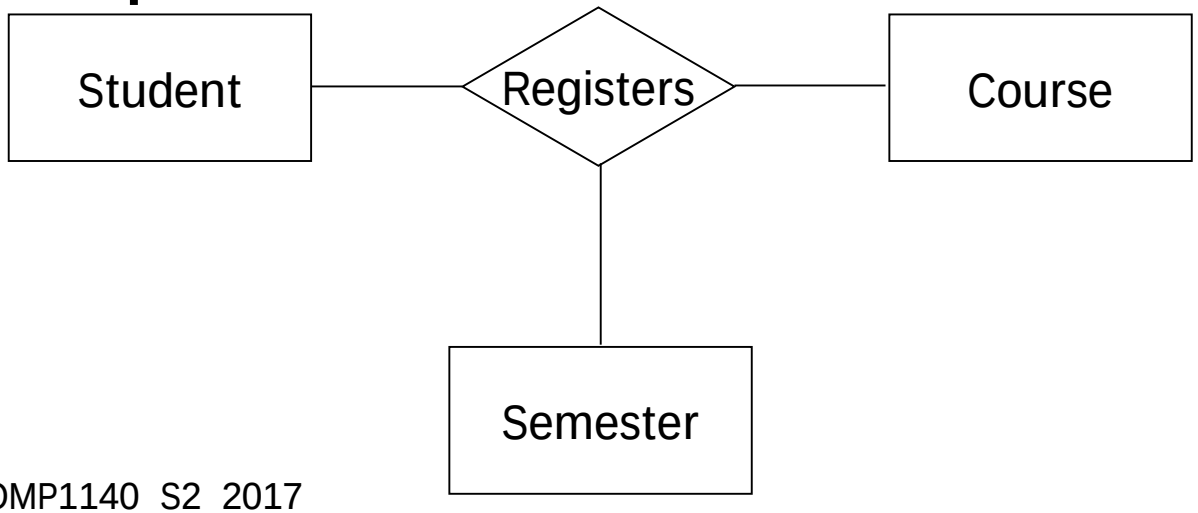
- Semantic Net Example



Cardinality constraints for degree > 2

- In n-ary relationships, we fix n-1 values to determine the cardinality

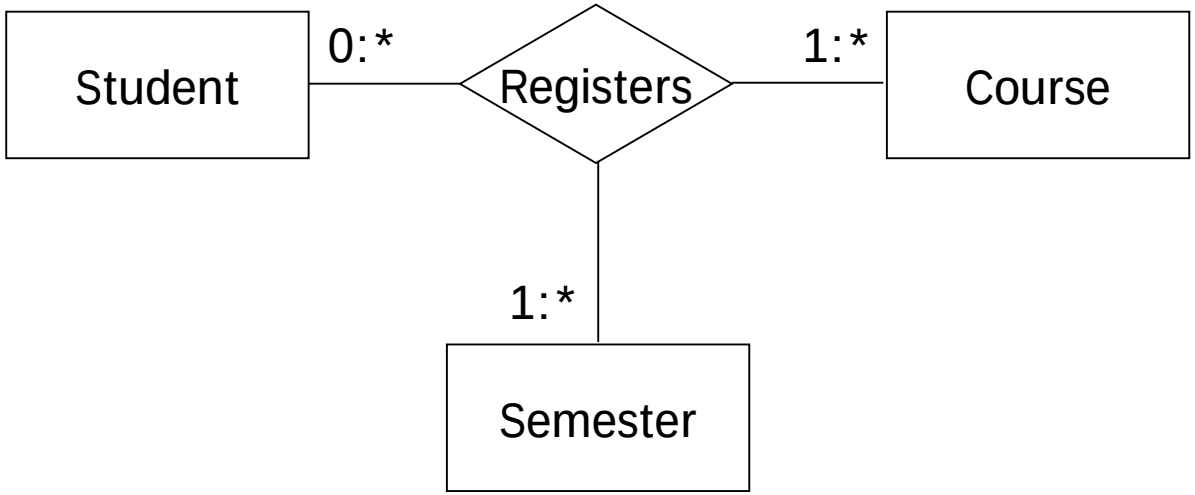
- Example

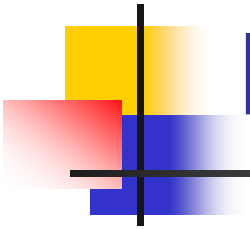


Cardinality constraints for degree > 2 (contd.)

- When student/semester entities are fixed, at least 1 course and there are many courses a student may register to → many on Course side (1:*)
- When semester/course entities are fixed, 0 or more students may be registered → many on Student side (0:*)
- When student/course entities are fixed, 1 or more semester entities may be registered to → many on Semester side (1:*)

Cardinality constraints for degree > 2 (contd.)





Participation constraints

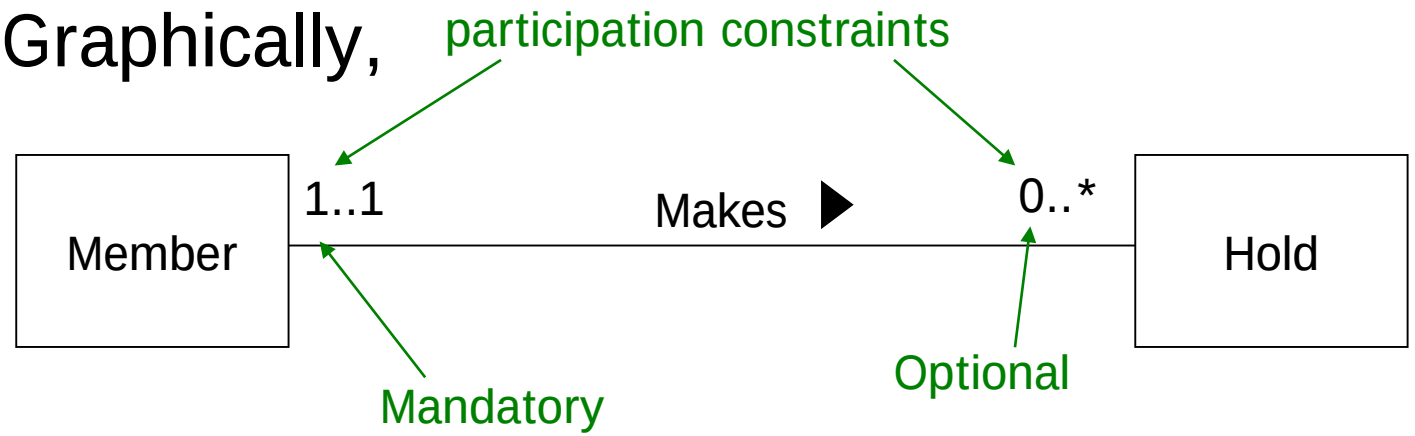
- Participation constraints determine whether an entity occurrence must participate in a relationship

- That is, if all entity occurrences must participate in a relationship, then the minimum multiplicity is more than zero (**mandatory**). Otherwise the minimum multiplicity is zero (**optional**)

Participation constraints (contd.)

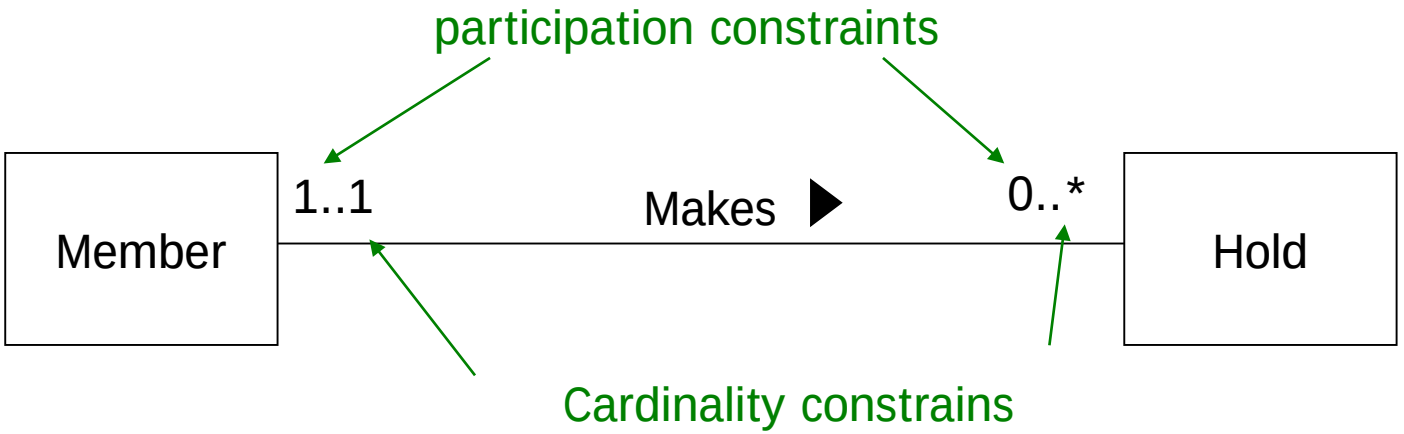
- Example
  - A hold must have a member who makes the hold (i.e. **mandatory**)
  - A book may be held by a member (i.e. **optional**)

- Graphically,



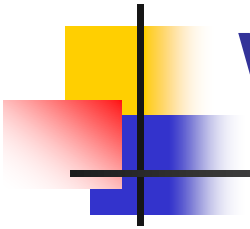
Multiplicity

- Is the number (or *range*) of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship
- is made up of two types of restrictions on relationships: *cardinality* and *participation*.



Multiplicity - summary

Alternative ways to represent multiplicity constraints	Meaning
0..1	Zero or one entity occurrence
1..1 (or just 1)	Exactly one entity occurrence
0..* (or just *)	Zero or many entity occurrences
1..*	One or many entity occurrences
5..10	Minimum of 5 up to a maximum of 10 entity occurrences
0, 3, 6–8	Zero or three or six, seven, or eight entity occurrences

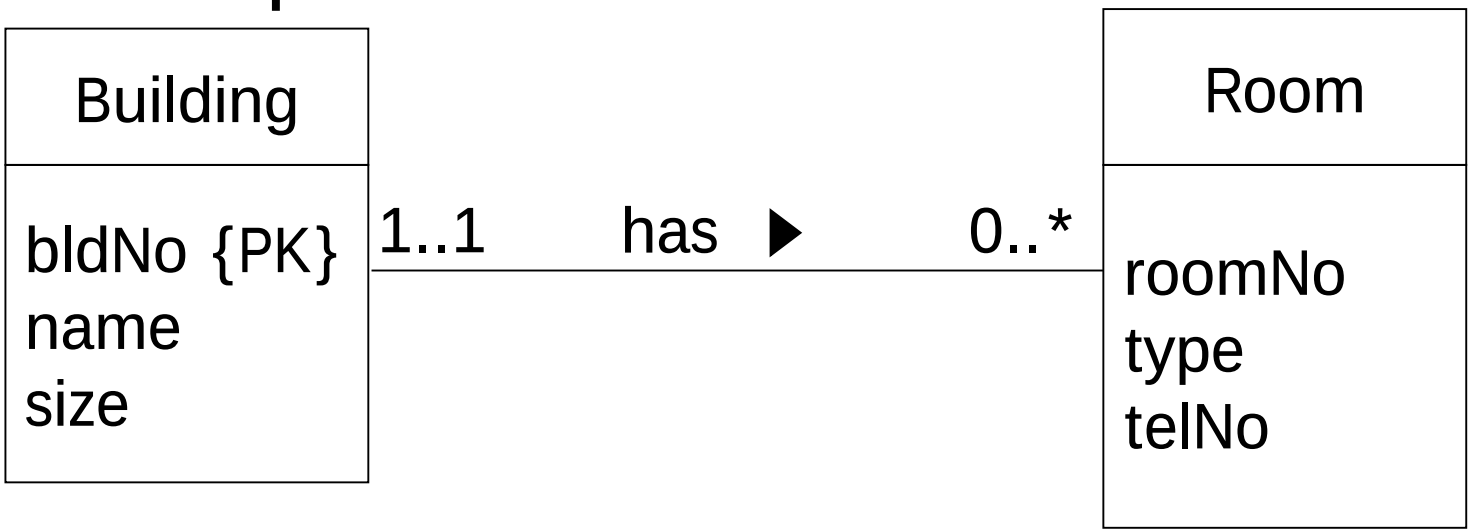


Weak entity

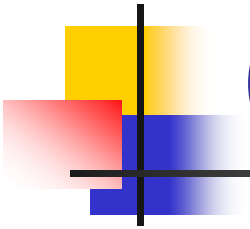
- A weak entity is an entity that depends on another entity (called the **owner entity**) for its existence
- Characteristic: each entity occurrence cannot be uniquely identified using only the attributes associated with that entity
- A weak entity does not have a key
- A weak entity creates a key for itself by combing a set of attributes (known as the **partial key**) with the primary key of the owner entity

Weak entity (contd.)

- Example



Enhanced Entity-Relationship (EER) Model



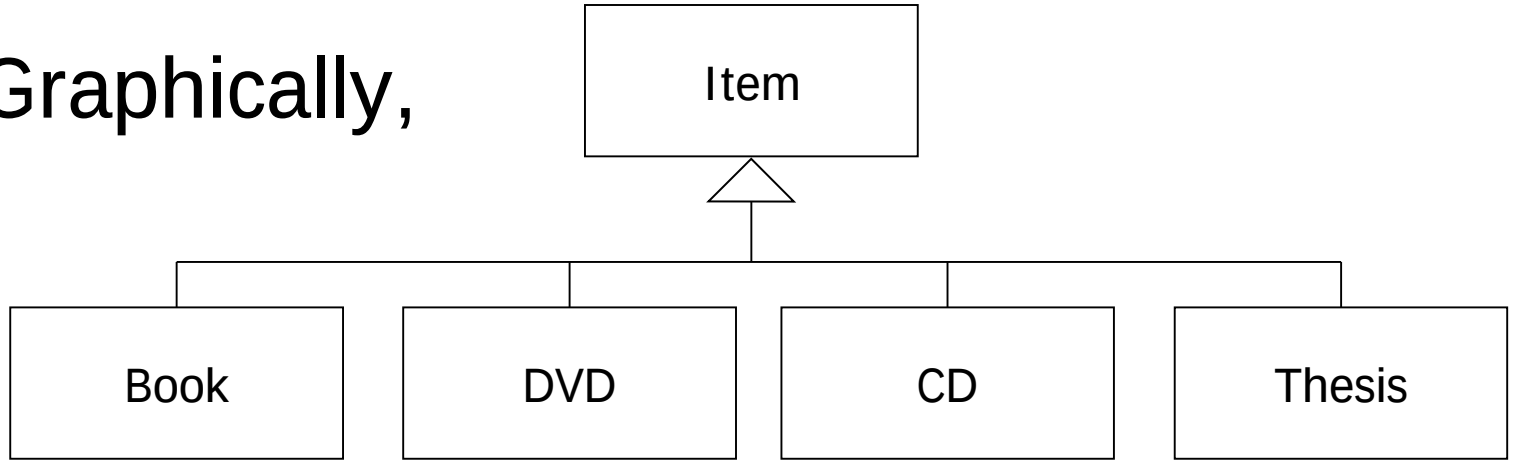
- Important enhancements were proposed to ER model to handle complex semantics

- Superclass/subclass relationship is one such modelling construct

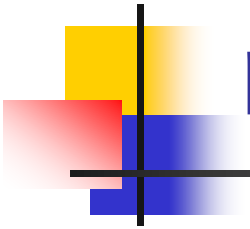
Superclass/subclass relationship

- Entity types may be classified into hierarchies based on superclass/subclass relationship

- Graphically,



Superclass/subclass relationship (contd.)



- *Item* entity set is the superclass. *Book*, *DVD*, *CD* and *Thesis* are subclasses
- Every member in each subclass is also a member of its superclass. However, not vice-versa
- All attributes of superclasses are inherited by its subclasses (also, known as **attribute inheritance**)

Specialization/Generalization Process

- There are two main approaches to defining superclass and related subclass:
  - **Specialization**: Top-down approach of defining superclasses and related subclasses
  - **Generalization**: Bottom-up approach of identifying common attributes and relationships from entities to create superclasses

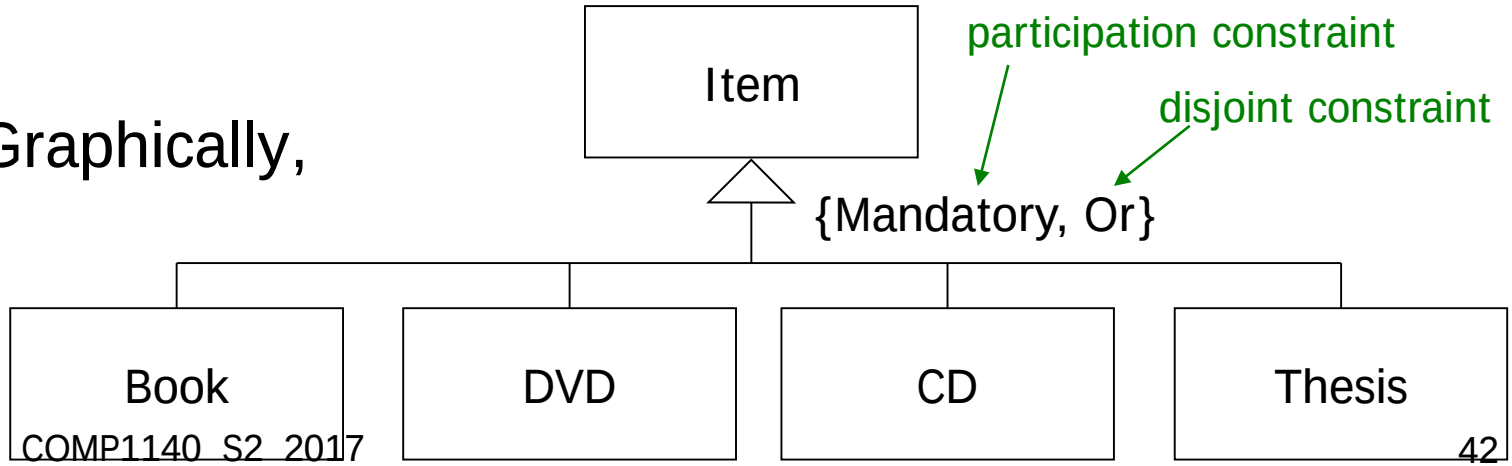
Constraints on superclass/subclass relationships

- Two types of constraints can be specified on superclass/subclass relationship:
  - **Participation constraint**
  - **Disjoint constraint**

- **Participation constraint** determines whether every member of superclass must participate as a member of a subclass
  - **Mandatory**: Every superclass member MUST be a member of a subclass (graphically – “mandatory” keyword)
  - **Optional**: Otherwise (graphically – “optional” keyword)

Constraints... (contd.)

- **Disjoint constraint**: determines whether it is possible for a member of a superclass to be a member of multiple subclasses:
  - **Disjoint**: If a superclass member cannot be a member of multiple subclasses (graphically, “Or” keyword)
  - **Nondisjoint**: Otherwise (graphically, “And” keyword)
- Graphically,



Conceptual Database Design – 9 steps

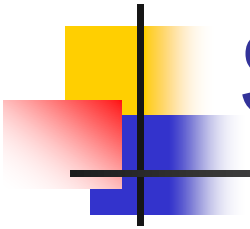
- Identify entities
  - To identify the required entity types
  - *Hint*: Look for nouns (e.g. Book, Member…)
- Identify relationships
  - To identify the important relationships that exist between the entity types
  - Also need to determine multiplicity constraints
  - *Hint*: Look for verbs (e.g. Member *belongs to* a MemberType)
- Identify and associate attributes with entity or relationship types
  - To associate attributes with the appropriate entity or relationship types and document the details of each attribute
  - Simple/composite, single/multi-value, derived

Conceptual Database Design (contd.)

- Determine attribute domains
  - To determine domains for the attributes in the data model and document the details of each domain
- Identify keys
  - Candidate, primary and alternate keys
- Consider use Enhanced ER concepts
  - Superclass/subclass
- Check for redundancy
  - Re-examine 1:1 relationships
  - remove redundant relationships
  - Consider time domains
- Validate model against user transactions
- Review, validate and iterate!

Assignment 1 (Specification)

- **Part 1: Requirements document**
  - Data Requirements – outlining the major data items
  - Transaction requirements – outlining the data manipulation and queries
  - Business Rules
  - E.g., Data Requirements for Library
- **Part 2: EER Model**
  - EER Model
  - Documentation – Data dictionary details (description of entities, relationships and attributes)
  - E.g., EER for Library

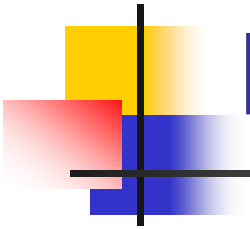


Summary

- Entity relationship model
  - Entities (Strong)
  - Attributes
    - Simple/Composite
    - Single/Multivalued
    - Derived
    - Keys – primary, candidate, composite
  - Relationships
    - Degree (binary, ternary, n-ary)
    - Recursive
    - Cardinality constraints
    - Participation constraints

Summary (contd.)

- Weak Entities
- Superclass/subclass relationship
  - Specialization/Generalization
  - Constraints
    - Participation
    - Disjoint
- Further clarification on Assignment 1
  - Can use Visio to draw EER
  - Brief marking guide is on BB
  - Q?



Review

- What are the 9 steps of conceptual database design?
- Be able to create an EER for an application